Interaction of Feature Selection Methods and Linear Classification Models

Janez Brank JANEZ.BRANK@IJS.SI
Marko Grobelnik MARKO.GROBELNIK@IJS.SI
Jožef Stefan Institute, Jamova 39, 1000 Ljubljana, Slovenia

Nataša Milic-Frayling NATASAMF@MICROSOFT.COM
Microsoft Research, 7 J J Thomson Ave., Cambridge CB3 0FB, UK

Dunja Mladenic DUNJA.MLADENIC@IJS.SI
Jožef Stefan Institute, Jamova 39, 1000 Ljubljana, Slovenia

Abstract

In this paper we explore effects of various feature selection algorithms on document classification performance. We propose to use two, possibly distinct linear classifiers: one used exclusively for feature selection in order to obtain the feature space for training the second classifier, using possibly a different training set. The resulting classifier is used to classify new documents. Experiments show that feature selection based on the linear SVM algorithm combines well with different types of classifiers. Based on the experimental results we make a conjecture that it is the level of sophistication at which the scoring method takes into account information about features, rather than its compatibility with the classifier in terms of its design, that makes the feature selection method more or less successful.

1. Introduction

Feature selection as a mechanism for ‘cleaning up’ the set of features for representing data and increasing the performance of the classifiers has been successfully applied in the past. For example, in the study of Naïve Bayes, feature selection based on odds ratio scores has consistently resulted in statistically significant improvements in classification performance over the use of the full feature set (Mladenic & Grobelnik, 1999).

Similar feature selection methods have been used in information retrieval, in search systems that support pseudo (automatic) feedback or a proper user feedback where a number of highly scoring documents or those marked relevant by the user are considered as a source of additional terminology for query expansion (Evans & Lefferts, 1995, Evans et al. 1996, Robertson et al. 1995).

In these scenarios a fundamental question arises: how can one best match a feature selection algorithm with a classification or a search algorithm. It is tempting to conjecture that each feature selection method will work best when combined with a learning algorithm with which it is most closely related by design. For instance, odds ratio is closely related to Naïve Bayes. However, experiments presented in this paper show that a rather different conjecture is more likely to be valid.

Recently we have compared classification performance of the linear SVM classifier when used in conjunction with various feature selection algorithms (Brank et al., 2002). There we introduced iterative training of classifiers that is appropriate in scenarios where a trade-off between the size of the feature space and the number of training data has to be made.

The idea is to first obtain an initial formulation of the classifier in the full feature space and then retrain the classifier using only a subset of features but possibly an expanded training set. The key point is that the initial classifier produces a ranking of features that could be used for feature selection. This is particularly intuitive in the case of linear classifiers like linear SVM where the normal on the separating hyper-plane can be explicitly determined. However, this may also be applicable to non-linear classifiers (Sindhwani et al., 2001).

For simplicity, in this study we restrict our attention to linear classifiers. We use two linear classifiers. The first classifier, trained over the full feature space, is used only for feature selection. The second linear classifier is then trained using the selected feature subset but with possibly a larger set of training data. The resulting classifier is used to categorize new documents. This type of experiments allows us to monitor the interaction of the feature selection and classification.

In the following sections we describe learning algorithms and feature selection algorithms that we explored in our experiments. We describe in detail the experiment setup and discuss the results we obtained. We conclude with suggestions for future explorations.

2. Learning algorithms

We compare classification performance of different feature selection methods in combination with the following learning algorithms:

Naïve Bayes. We use the multinomial model as described by McCallum and Nigam (1998). The predicted class for document d is the one that maximizes the posterior probability $P(c|d)$, which is proportional to $P(c)\Pi_w P(t|c)^{\mathrm{TF}(t,d)}$, where $P(c)$ is the prior probability that a document belongs to class c, $P(t|c)$ is the probability that a word w, chosen randomly in a document from class c equals t, and $\mathrm{TF}(t, d)$ is the "term frequency", or the number of occurrences of word t in a 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 $\mathrm{TF}(w, d)$. 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.

Perceptron (Rosenblatt, 1958). This algorithm trains a linear classifier as a neural unit using an additive update rule. The prediction for a document represented by the vector $\mathbf{x}$ is $\mathrm{sgn}(\mathbf{w}^{\mathrm{T}}\mathbf{x})$ where $\mathbf{w}$ is a vector of weights obtained during training. Computation starts with $\mathbf{w} = 0$, 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: $\mathbf{w} \leftarrow \mathbf{w} + y_i \mathbf{x}_i$ where y_i is the correct class label of the document $\mathbf{x}_i$ ($y_i = +1$ for a positive document, $y_i = -1$ for a negative one).

Several passes may need to be made through the training set. If the problem is linearly separable this algorithm is known to converge to some $\mathbf{w}$ which correctly classifies all the training examples. Otherwise, one may stop training after a few passes through the training data if the performance of the classifier stops improving. Various modifications and improvements of the original perceptron algorithms have been made, such as introducing a threshold or a margin (Krauth and Mézard, 1987).

Support Vector Machine (SVM) (Cortes & Vapnik, 1995). This algorithm trains a linear classifier of the form $\mathrm{sgn}(\mathbf{w}^{\mathrm{T}}\mathbf{x} + b)$. Learning is posed as an optimization problem with the goal of maximizing the *margin*, i.e., the distance between the separating hyperplane $\mathbf{w}^{\mathrm{T}}\mathbf{x}+b=0$ and the nearest training vectors. 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 v.3.5 program by Joachims (1999) to train the SVM models. This program is particularly suitable for our purposes because of its optimizations for working with linear kernels.

Both the perceptron and the SVM can be used for learning nonlinear models by mapping the training vectors from their original d-dimensional real space R^d into a higher-dimensional "feature space" F. If this mapping (say φ) is nonlinear, a hyper-plane in the feature space will correspond to a nonlinear decision surface in the original space. In addition, it is not necessary to represent images $\varphi(\mathbf{x})$ explicitly, as long as the inner products between them can be computed. A function $K(\mathbf{x},\mathbf{z}) = \langle \varphi(\mathbf{x}), \varphi(\mathbf{z}) \rangle_F$ used for this purpose is called a *kernel*. The linear kernel, $K(\mathbf{x},\mathbf{z}) = \mathbf{x}^{\mathrm{T}}\mathbf{z}$, corresponds to working in the original space.

We work with linear kernels throughout these experiments because the existing literature on text categorization indicates that for this problem nonlinear kernels gain very little in terms of categorization performance.

3. Feature selection methods

The feature selection methods we consider in this paper are all based on assigning a score to each feature that suggests how important or valuable the feature is likely to be for training and categorization. Features are then ranked based on this score and the top ranked features are kept while others are discarded.

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

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

Obviously, this scoring measure favors features that are representative of positive examples. 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 documents will be ranked at the top of the feature list. Odds ratio is known to work well in combination with Naïve Bayes (Mladenic & Grobelnik, 1999).

Information gain. Here class membership is seen as a random variable C with two values, positive and negative, and a word as a random variable T with two values, *present* and *absent*. Then, using the information-theoretic definition of mutual information, we may define $IG(t) = H(C) - H(C|T)$. In effect, this is the amount of information about C (the class label) gained by knowing T (the presence or absence of a given word).

Feature selection based on linear classifiers. A linear classifier outputs predictions of the form

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

Thus a feature j with the weight w_j close to 0 will have a smaller effect on the prediction that features with large absolute values of w_j. The weight vector $\mathbf{w}$ can also be seen as the normal to a hyperplane used by the classifier to separate positive from negative instances.

One speculates that since features with small $|w_j|$ are not important for categorization they may also not be important for learning and are therefore good candidates for removal. The effect of this type of feature selection on the classification performance, in particular when the linear SVM is used for training and feature ranking, has been explored in detail in Brank et al. (2002).

4. Experiments

4.1 The experimental setting

The experiments presented in this section are all based on the Reuters Corpus, Volume 1 (released in 2000), which contains approximately 810,000 news articles from the period 20 August 1996 through 19 August 1997. We divided the corpus into a "training period" covering the 504,468 articles dated 14 April 1997 or earlier, and a "test period" covering the remaining 302,323 documents. Because it would be computationally very demanding to train on all the documents from the training period, we selected a sample of 118,294 documents as a training set (the remaining documents from the training period were not used in our experiments). The entire test period was used as a test set.

In this corpus, each document may belong to several of 103 categories available in the Reuters' classification scheme. Because running experiments on all categories would be too time-consuming, we conducted a preliminary experiment to choose a smaller set of categories for the work in this study. Two criteria were used in choosing categories: the size of the category (i.e., the number of documents containing the class label in the whole corpus), and its "difficulty" of classification, measured by the precision-recall break-even point achieved by the linear SVM classifier that was trained on a smaller training set having documents from all the categories.

From the histogram of category size and performance statistics we chose the following 16 categories (listed here in order of their increasing size) that approximately follow the statistical distribution of the full set of 103 tags: E142, GOBIT, E132, C313, E121, GODD, GHEA, E13, C183, M143, GSPO, C13, E21, GPOL, M14, C15. To verify whether our choice of categories is in some sense biased, we later also ran an experiment for all 103 categories, with the same training and test sets as those described below, with no feature selection and with linear SVM as the training method. The global macroaveraged F_1 over all 103 categories is 0.6640, while the macroaverage over our 16 categories equals 0.6837. Additionaly, we computed macroaverages for 1000 random choices of 16 categories; in 352 of these cases, the distance from the global macroaverage was less than in the case of our 16 categories listed above, while on average the distance was 1.8 times as large as for our sample. This suggests that our choice of categories is not unreasonably biased.

Our document representation is based on the bag-of-words model. We convert each document into lowercase, remove stop-words (from a standard set of 523 stop-words) and words occurring in less than 4 training documents. We represent each document as a vector of TF-IDF weights, normalized to the unit length, except when training with Naïve Bayes when plain TF vectors are used.

4.2 Sparsity of the documents after feature selection

In order to compare classification performance of several feature selection methods, it seems natural to observe how the performance changes as we vary the number of features retained before the final training phase. However, taking the number of features as the independent variable puts different feature selection methods on a very different footing. For example, it is well known that odds ratio gives high scores to many rare features (i.e., features occurring in a small number of positive documents only), and that consequently one must keep many best-scoring features in order to avoid having many zero-length training vectors. On the other hand, information gain rewards features whose presence or absence tends to match well with the document's membership of one class or the other, and features that occur in very few documents do not score well according to this criterion.

For the particular data set in this study, keeping 100 best features according to the information gain criterion leaves an average of 4.1 terms per document, while keeping 100 best features according to the odds ratio values yields an average of 0.01 terms per document (i.e., the vast majority of the document vectors is empty). On the other hand, the results of the SVM- and perceptron-based feature selection fall between these two extremes: on average 2.3 and 1.8 terms per document, respectively. For the purpose of this discussion we shall use the term "sparsity" to indicate the average number of terms per document (or, in more technical terms, the average number of nonzero components in the sparse vectors by which documents are represented).

While a relatively large number of features required by some methods (in particular, the odds ratio) to adequately represent the training documents might seem a significant disadvantage, this is not the case with algorithms such as SVM and the perceptron, both of which can easily take as input a large number of features. However, the greater sparsity of document vectors after feature selection is an important parameter: it directly influences both the memory requirement for storing the sparse vectors and the time needed to perform calculations such as computing dot products (required for SVM and perceptron) or $P(c|d)$ probabilities required by Naïve Bayes. Note that, while reducing the total number of features is expected to increase sparsity, the rate at which this is achieved depends on the distribution characteristics of features that are removed. Also, retaining rare features generally incurs a low cost to data storage and calculations.

Consequently, to avoid an unreasonably strong bias against feature selection methods that favor rare features, we use, as the independent variable in our experiments, the sparsity of the document representation rather than the number of features retained after feature selection.

4.3 Results

Throughout these experiments, we use the macroaveraged F_1 measure to measure the quality of different classifiers, as this evaluation measure is commonly used in document classification. Each category can be viewed as a two-class problem, with members of that category being positive examples and all other documents being negative examples. Precision, p, is the proportion of documents predicted positive that are actually positive. Recall, r, is defined as the proportion of positive documents that are predicted positive. The F_1 measure is then defined as $F_1 = 2pr/(p+r)$. The average of the F_1 measure over all categories is known as the macro average. We also investigated the break-even point performance measure; however, because of space limitations, we will only present detailed results for the F_1 measure.

The charts on Figure 1 show the macroaveraged F_1 performance for various combinations of feature selection measures and training algorithms. The graphs are plotted putting the various levels of sparsity on the horizontal axes: 2, 2.5, 5, 10, 20, 40, and 80 terms per document as well as the full document vectors which have about 88.3 terms per document on average.

In addition to two standard feature selection methods, odds ratio and information gain, we report classification results using (1) feature sets that are obtained from the normal vectors of the linear SVM classifiers learned over three different sets of training data: the full training set, a randomly chosen subset containing $^1/_4$ as many documents, and one containing $^1/_{16}$ as many documents (these variants are denoted *normal-1, normal-*$^1/_4$, and *normal-*$^1/_{16}$ in the charts), and (2) feature sets obtained from the weight vectors of perceptrons trained over the same subsets of training data (feature sets denoted by *perceptron-1, perceptron-*$^1/_4$*, perceptron-*$^1/_{16}$).

These eight sets of features are used as input to the Naïve Bayes, and for retraining the perceptron and the linear SVM classifier over the full training set, keeping a subset of features. The resulting classifiers are then applied to the test data. The following sections comment on the experiment results presented in Figure 1.

4.3.1 Naïve Bayes

The experiments with Naïve Bayes confirm the well-known fact that Naïve Bayes benefits greatly from appropriate feature selection. This is particularly true of the performance on smaller categories, while on the largest few categories there is little or no improvement from feature selection. Unexpectedly, odds ratio works well when documents are made moderately sparse (around 10 to 20 terms per document, which requires on average approximately 4000 to 8000 of the best features to be retained) but not extremely sparse. This is in contrast to earlier research, where odds ratio was found to work well even if only a few hundred features were kept (Mladenic and Grobelnik, 1999). It would be worth exploring whether sparsity is a more reliable predictor of the 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).

Still higher performance can be achieved with feature selection based on the weighting obtained from the perceptron and the linear SVM models, even when the latter are only trained on smaller subset of the document set. *Table 1* gives the performance results for the tested feature selection measures.

Feature selection/ranking method	Sparsity which yielded best performance	Macroaveraged F_1 at that sparsity
Odds ratio	20	0.4683
Information gain	2.5	0.4648
Normal-$^1/_{16}$	5	**0.5208**
Normal-$^1/_4$	5	**0.5356**
Normal-1	5	**0.5421**
Perceptron-$^1/_{16}$	10	0.4715
Perceptron-$^1/_4$	10	0.4887
Perceptron-1	10	0.5005

Table 1. This table shows, for each feature selection method, the best performance achieved by this method, and the sparsity where it was achieved. In all cases, Naïve Bayes was used as the training method after feature selection.

Comparing these results with the other two classifiers, we observe that even with the best of the tried feature selection methods, Naïve Bayes cannot match the performance of the perceptron and the linear SVM classifiers.

4.3.2 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*. When all features in the training set are considered the data is represented with 88 terms per document on average. The reduction of 8 features on average per document is in fact facilitated by retaining around 32,500 out of 75,839 original features. The increase in F_1, although statistically significant, is marginal (0.8 %), and this should probably be seen more as a data cleansing operation than as feature selection. Otherwise, performance drops quickly and considerably as more and more features are discarded.

Using perceptron-based feature rankings works better here than using SVM-based rankings, although the differences 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 *normal-1* at the same sparsity. The differences between *perceptron-*$^{1}/_{4}$ and *normal-*$^{1}/_{4}$ and between *perceptron-*$^{1}/_{16}$ and *normal-*$^{1}/_{16}$ are smaller and often insignificant. Thus in scenarios 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 normal-based rankings of features achieve lower precision and lower F_1 values they tend to have higher recall and break-even point for the same subset of training documents.

Feature ranking proposed by the information gain criterion usually performs slightly but significantly worse than the rankings of linear classifiers. On the other hand, the ranking that results from odds ratio scores performs much worse (interestingly, odds ratio 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 odds ratio to favor features characteristic of positive documents, in particular those that are rare and not present in negative documents. It would be interesting to see whether perceptron models with margins are more robust in that respect.

4.3.3 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 the point where training documents are represented by 10 terms 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 still very small) improvements to the F_1 performance (in comparison to the full feature set) are only achieved when using *normal-1* or *perceptron-1* feature sets to increase the sparsity from 88.3 to about 80 terms per document.

Odds ratio works well in combination with the linear SVM classifier, particularly when very sparse documents are required. At sparsity of 20 or more terms per document, odds ratio performance does not differ significantly from that of information gain, but it is significantly better than information gain for sparser document vectors. There it also outperforms *normal-*$^{1}/_{16}$ and, at more extreme levels of sparsity, eventually *normal-*$^{1}/_{4}$ and even *normal-1* (for sparsity of less than 2.5 terms per document on average). Thus odds ratio is a good and robust feature selection measure for use with SVM.

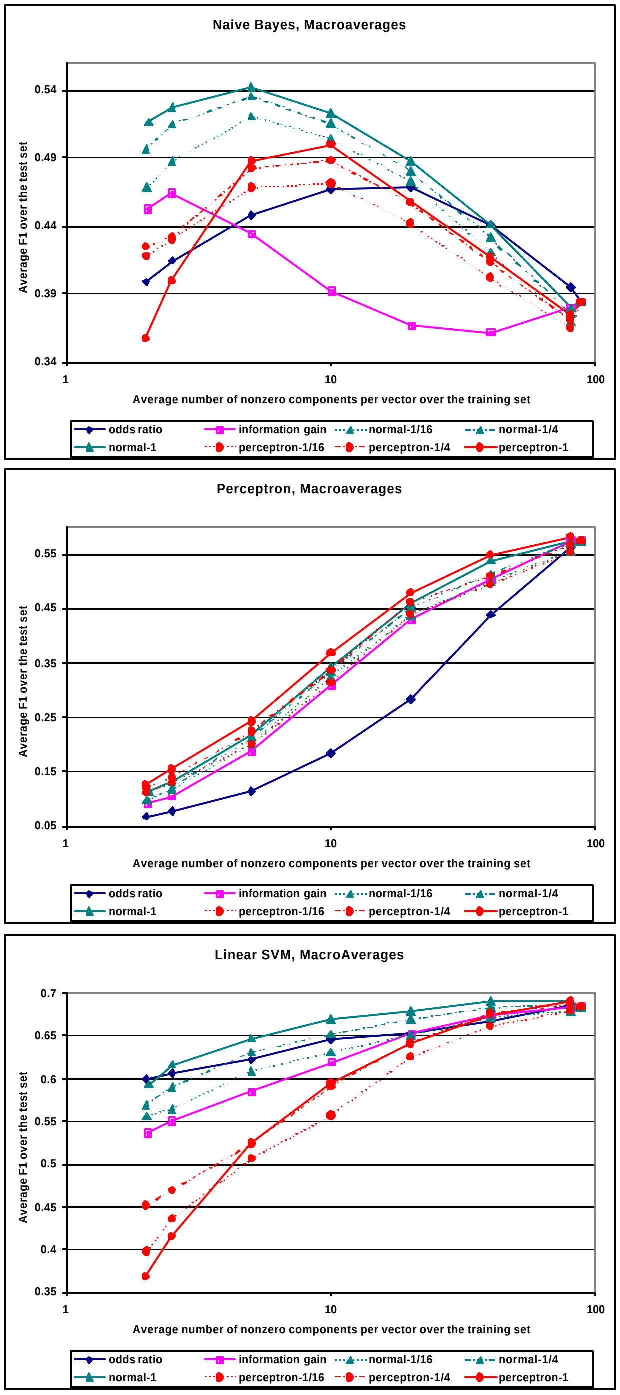

Figure 1. Macroaveraged F_1 performance of different feature selection methods in combination with different machine learning algorithms. Note the different scales on the vertical axes. The horizontal axes refer to sparsity, i.e., the average number of nonzero components per training vector (that is equivalent to the average number of terms per training document after terms rejected by feature selection have been removed).

An interesting occurrence here is that the perceptron-based feature weightings perform much worse than the normal-based weightings. In addition, *perceptron-1* is generally not significantly better than *perceptron-$^1/_4$*, and for extremely sparse documents it is in fact significantly worse.

Looking at precision and recall separately shows that perceptron-based feature rankings have particularly poor recall. It turns out that the perceptron-based feature weighting has greater preference for rare features than normal-based weighting (though not nearly so great as odds ratio), especially for features typical of positive documents.

Indeed, if we plot sparsity curves which show how the density of document vectors increases with the number of features retained in the feature set, the curve for *perceptron-x* is rising more slowly than that for *normal-x* (for the same value of *x*) for the set of positive documents. For negative documents the difference is much smaller. We speculate that this preference for features that are present in positive documents but rather uncommon may be the cause of poor recall (and consequently poor F_1).

Interestingly, information gain tends to produce high precision but low recall, whereas the opposite holds for odds ratio. The normal-based rankings are good at both precision and recall, particularly the latter.

5. Conclusions

Our experiments show that feature scoring and selection based on the normal obtained from the linear SVM combines very well with all the classifiers considered in the study.

While feature selection based on odds ratio shows its beneficial effect on the Naïve Bayes classifier, the selection based on SVM normals seems far more effective for all levels of sparsity below 40 features per vector. Use of feature scores obtained from the perceptron algorithm also produces much better feature rankings than odds ratio.

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 the odds ratio, which closely follows the feature scoring used within Naïve Bayes, should provide the best selection for that classifier. However, the method like SVM that tunes the normal by taking into account all the data points simultaneously seems to be most successful in scoring features.

Use of SVM normal as the feature selection method combines relatively well with the perceptron classifier considering a rather dramatic negative effect of feature selection on the perceptron compared to other results. One could argue that, because the *perceptron-1* is the best performing feature ranking with the perceptron classifier, the compatibility of score hypothesis takes precedence over the above conjecture. However, for the higher performance levels (F_1>0.45) of the perceptron classifier the impact of the SVM-based and perceptron-based feature selection is almost identical when smaller subsets of training data (such as $^1/_4$ of the training set) are used. This makes us believe that our conjecture is in the right direction.

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.

Further explorations should expand the study to additional classifiers, linear and non-linear, and diversify the feature scoring algorithms to include those that possibly include information of feature dependencies or similar characteristics, leading to more sophisticated data modeling.

References

Brank, J., Grobelnik, M., Milic-Frayling, N., & Mladenic, D. *Feature selection using linear SVM*. Microsoft Research Tech. Rept. MSR-TR-2002-63, June 2002.

Evans, D. A. & Lefferts, R.G. CLARIT-TREC experiments. *Inf. Processing and Mgmt., 1995, Vol. 31, 385-395.*

Evans, D. A., Milic-Frayling, N., & Lefferts, R.G. CLARIT TREC-4 Experiments. In: *The Fourth Text Retrieval Conference (TREC-4)*. Harman, D. K. (Ed.). NIST, 1996.

Joachims, T. (1999). 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.

Krauth, W., & Mézard, M. (1987). Learning algorithms with optimal stability in neural networks. *Journal of Physics A 20*, L745–L752.

McCallum, A., & Nigam, K. (1998). A comparison of event models for Naïve Bayes text categorization. *Learning for Text Categorization: Papers from the AAAI Workshop* (pp. 41–48). AAAI Press.

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

Robertson, S.E., Walker, S., Jones, S., & Hancock-Beaulieu, M. M. Okapi at TREC-3. *In: Third Text Retrieval Conference (TREC-3)*. Harman, D. K., (Ed.). NIST 1995.

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

Sindhwani, V., Bhattacharya, P., & Rakshit, S. (2001). Information theoretic feature crediting in multiclass Support Vector Machines. *1st SIAM Int. Conf. on Data Mining*. Philadelphia: SIAM.