Comparison of SVM and Some Older Classification Algorithms in Text Classification Tasks

Fabrice COLAS[1] and Pavel BRAZDIL[2]

[1] LIACS, Leiden University, THE NETHERLANDS, fcolas@liacs.nl
[2] LIACC-NIAAD, University of Porto, PORTUGAL, pbrazdil@liacc.up.pt

Summary. Document classification has already been widely studied. In fact, some studies compared feature selection techniques or feature space transformation whereas some others compared the performance of different algorithms. Recently, following the rising interest towards the Support Vector Machine, various studies showed that SVM outperforms other classification algorithms. So should we just not bother about other classification algorithms and opt always for SVM ?

We have decided to investigate this issue and compared SVM to kNN and naive Bayes on binary classification tasks. An important issue is to compare optimized versions of these algorithms, which is what we have done. Our results show all the classifiers achieved comparable performance on most problems. One surprising result is that SVM was not a clear winner, despite quite good overall performance. If a suitable preprocessing is used with kNN, this algorithm continues to achieve very good results and scales up well with the number of documents, which is not the case for SVM. As for naive Bayes, it also achieved good performance.

1 Introduction

The aim of using artificial intelligence techniques in text categorization is to build systems which are able to automatically classify documents into categories. But as the feature space, based on the set of unique words in the documents, is typically of very high dimension, document classification is not trivial. Various feature space reduction techniques were suggested and compared in [13, 9]. A large number of adaptive learning techniques have also been applied to text categorization. Among them, the k nearest neighbors and the naive Bayes are two examples of commonly used algorithms (see for instance [7] for details). JOACHIMS applied the Support Vector Machine to document classification [4]. Numerous classifier comparisons were done in the past [12, 14, 4, 2].

Some algorithms like the SVM are by default binary classifiers. Therefore, if we have a problem with more than two classes, we need to construct as

Please use the following format when citing this chapter:

Colas, F., Brazdil, P., 2006, in IFIP International Federation for Information Processing, Volume 217, Artificial Intelligence in Theory and Practice, ed. M. Bramer, (Boston: Springer), pp. 169–178.

many classifiers as there are classes (*one versus all* strategy). However, it is not fair to compare a single multi-class naive Bayes (or kNN) classifier to n SVM classifiers (for n classes). This is why we have decided to focus on *one against one* classification tasks. Moreover, FÜRNKRANZ [3] showed that a *round robin* approach using the set of *one against one* classifiers, performs at least as well as a *one versus all* approach. These binary problems involve also smaller amounts of data, which means that the classifiers are faster to train. The properties of the train set have much influence on the classifier learning abilities. Therefore, focusing on binary classification tasks allows one to carefully control the nature of train sets. Finally, directly studying multi-class classification tasks tends to obscure the particular behaviors of the classifiers on some classes which may be of interest.

We seek answers to the following questions. *Should we still consider old classification algorithms in text categorization or opt systematically for SVM classifiers ? What are the strength and weaknesses of the SVM, naive Bayes and kNN algorithms in text categorization on a set of simple binary problems ? Are there some parameter optimization results transferable from one problem to another ?* Before giving the answers, our experimental settings and evaluation methodology are described. Then, our results regarding the parameter optimization are presented. The optimized versions are then used in further comparative studies, which are used to answer the above questions.

2 Document Collection, Algorithms and Evaluation Methodology

2.1 Document Collection

For our experiments we used the well known `20newsgroups` dataset composed of 20000 newsgroup emails (removed email headers and no stemming). We chose to study the set of *one against one* binary classification tasks of this dataset. Thus, $\frac{20(20-1)}{2} = 190$ classification tasks were examined. Given the large dimensions of the problem, sub sampling techniques were applied to observe the classifier learning abilities for an increasing train set size. We also used the *Information Gain* to impose an ordering on a set of attributes. We chose this heuristic for its simplicity and its good performance [13, 9].

2.2 Algorithms

In this paper, two well known classifiers are compared to the Support Vector Machine namely the kNN and the naive Bayes. These two classifiers were chosen because of their simplicity and their generally good performance reported in document classification. With respect to the SVM, the SMO implementation of PLATT, available in the `libbow` [6] library, has been used.

Let us consider the classification function Φ of the data points $\boldsymbol{x}_i$ $(i = 1...l)$ into a class $y_i \in C = \{+1, -1\}$. Let d be the dimension of the feature space. The three classification algorithms are presented in the following subsections.

Support Vector Machine.

The SVM problem (*primal*) is to find the decision surface that maximizes the margin between the data points of the two classes. Following our results and previously published studies in document classification [12, 14], we limit our discussion to *linear* SVM. The *dual* form of the *linear* SVM optimisation problem is to maximize :

$$\begin{aligned} \boldsymbol{\alpha}^* = \text{maximise}_{\boldsymbol{\alpha}} \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i=1}^{l} \sum_{j=1}^{l} y_i y_j \alpha_i \alpha_j \langle \boldsymbol{x}_i, \boldsymbol{x}_j \rangle, \\ \text{subject to } \sum_{i=1}^{l} y_i \alpha_i = 0 \\ 0 \leq \alpha_i \leq C, i = 1...l \end{aligned} \tag{1}$$

with α_i the weight of the examples and C the relative importance of the complexity of the model and the error. The class prediction $\hat{\Phi}(\boldsymbol{x}')$ of the point $\boldsymbol{x}'$ is given by :

$$\hat{\Phi}(\boldsymbol{x}') = sign(\sum_{i=1}^{l} \alpha_i y_i \langle \boldsymbol{x}_i, \boldsymbol{x}' \rangle + b) = sign(\langle \boldsymbol{w}^*, \boldsymbol{x}' \rangle + b) \tag{2}$$

where $\boldsymbol{w}^* = \sum_{i=1}^{l} \alpha_i y_i \boldsymbol{x}_i$.

k Nearest Neighbors.

Given a test point, a predefined similarity metric is used to find the k most similar points from the train set. For each class y_i, we sum the similarity of the neighbors of the same class. Then, the class y_i with the highest score is assigned to the data point $\boldsymbol{x}'$ by the k nearest neighbors algorithm.

$$\hat{\Phi}(\boldsymbol{x}') = argmax_{y_j \in C} \sum_{i=1}^{k} \delta(y_j, \Phi(\boldsymbol{x}_i)) sim(\boldsymbol{x}_i, \boldsymbol{x}') \tag{3}$$

Naive Bayes

Let $P(y_i)$ be the prior probability of the class y_i and $P(a'_j|y_i)$ be the conditional probability to observe attribute value a'_j given the class y_i. Then, a naive Bayes classifier assign to a data point $\boldsymbol{x}'$ with attributes $(a'_1...a'_d)$ the class $\hat{\Phi}(\boldsymbol{x}')$ maximizing :

$$\hat{\Phi}(\boldsymbol{x}') = argmax_{y_i \in C} P(y_i) \prod_{j=1}^{d} P(a'_j|y_i) \quad (4)$$

2.3 Evaluation Methodology

A classical 10-fold cross validation was used to estimate classifier performance. We chose the macro averaged F_1 measure $MF_1 = \frac{2 \times MPrecision \times MRecall}{MPecision + MRecall}$ [10], where the $MPrecision$ and the $MRecall$ measures are the averages of the precision and the recall computed on the basis of the two confusion matrices (in one, a class is considered positive and the other negative ; in the other the assignment is interchanged). Finally, we recorded the global processing time in seconds (the sum of the training and the testing time). As the size of the test set is nearly the same for each experiment, this processing time reflects mostly the train time of the classifiers.

3 Experimental Results

3.1 Parameter Optimization Results

We ran some preliminary experiments on `20newsgroups` to find the best parameter values. These experiments were restricted to three binary classification tasks[3]. Our results are presented in the following sections for SVM and kNN.

Support Vector Machines.

Various parameters of SVM were considered in the attempt to optimize the performance of this algorithm. The parameter C was varied and various kernel functions were tried as well. None of those lead to interesting improvements in terms of performance (MF_1) or processing time. So, the default value $C = 200$ and a *linear* kernel are used.

We have also varied the ϵ parameter controlling the accepted error. We have found that ϵ had no influence on MF_1 as long as its value was smaller or equal to 0.1. However, ϵ did affect the training time. Indeed the time could be reduced by a factor of 4 in the best case (see Fig. 1 (A) with 500 features), when the largest value of ϵ (0.1) was used. Our hypothesis is that the precision of the optimisation problem is simplified when an acceptable optimal hyper plane is bounded by a larger error ϵ. Therefore, it seems that no high precision is needed to train SVM on these binary classification tasks. Fig. 1 (A) and (B) portray the training time of SVM for various values of ϵ when the size of the feature space is varied and when the number of documents in the train set is increased.

[3] `alt.atheism` vs. `talk.religion.misc`, `comp.sys.ibm.pc.hardware` vs. `comp.sys.mac.hardware`, `talk.politics.guns` vs. `talk.politics.misc`

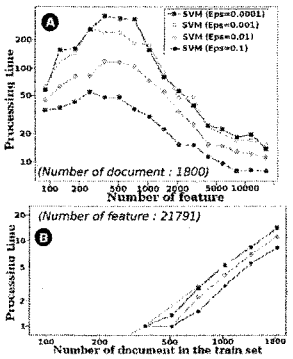

Fig. 1. *Processing time of the SVM classifier on* `alt.atheism` *vs.* `talk.religion.misc`*, for several values of* ϵ*, given an increasing number of features* ***(A)*** *and an increasing number of documents in the train set* ***(B)****.*

k Nearest Neighbors.

Two parameters were considered to optimize the performance of the kNN, the *number* k *of nearest neighbor* and the *feature space transformation.* Indeed, to achieve good performance with kNN, the feature space should be transformed to a new one. Common transformation in text mining are based on the number of occurences of the i^{th} term tf_i, the inverse document frequency which is defined as the ratio between the total number of documents N and the number of documents containing the term df_i, a normalization constant κ.

$$\begin{aligned}
\Phi_{\texttt{atc}}(x_i) &= \frac{(\frac{1}{2} + \frac{tf_i}{2tf_{max}})\ log(\frac{N}{df_i})}{\kappa} \\
\Phi_{\texttt{ntn}}(x_i) &= tf_i\ log(\frac{N}{df_i}) \\
\Phi_{\texttt{lnc}}(x_i) &= \frac{log(tf_i)}{\kappa}
\end{aligned} \tag{5}$$

Thirty measures (3 problems, 10-fold cross validation) caracterized the experimental results for each parameter setting. To compare these results, a simple heuristic based on a pairwise t-test (95% confidence interval) was used. When a significative difference was observed regards the results of one parameter setting to one other, a victory point was attributed to the best setting. In case of tie, no point was given. Train sets with the maximum number of document[4] were used whereas the feature space was composed of *all* the possible attributes.

Number of Nearest Neighbors.

We observed that large k values lead to relatively good performance. Indeed, the contribution towards the class score of the neighbors is weighted by their

[4] A binary task involves 2 × 1000 documents. Considering that 10-fold cross validation is used, each training set includes 1800 documents and the test set 200.

similarity to the test point. Therefore, the farthest neighbors have little effect on the class score. However, the best number of nearest neighbors is $k = 49$. This optimal k value (49) is interestingly quite close to the one of YANG (45) in [12] with completely different experimental settings (`Reuters-21570`, classification task seen as a single multi-class problem). As a result, it seems that k values between 45 and 50 are well suited for text classification tasks.

We first ran all our experiments with $k = 5$. Therefore the kNN results could be slightly improved in the following comparative study if we used the optimized value for k.

Feature Space Transformation.

About 400 transformations were evaluated. Our observation is that any term frequency is suitable, but not the binary transformation (value 1 or 0), depending whether a particular word is (or is not) present. This is coherent to the previous study of MCCALLUM et al. [5]. In the same way, the inverse document frequency should be systematically applied because, as it is well known, it decreases the importance of common words occurring in numerous documents. The normalization did not affect the performance. In the coming comparative study, the transformations[5] presented in formulas 5 are used.

3.2 Some Classifier Comparisons

The aim of our experiments was to examine the classifier learning abilities for an increasing number of documents in the train set (learning curves), and also, how the performance is affected by the number of attributes of the feature space. In the study involving learning curves, *all the features* were selected. Similarly, when the behaviors for an increasing number of features were studied, the train set was composed of its *maximum size*, containing as many documents of both classes.

First of all, we have observed that architectural variables (sample selection, algorithm parameters, feature subset selection, working feature space) had often a *larger* impact on the performance than the choice of individual classifiers. In fact, if suitable architectural variables are chosen and if the parameter settings of the classifiers get correctly optimized, then the differences between the algorithms are not very large.

Moreover, the behaviors of the classifiers are very similar across the classification tasks. This is illustrated in Fig. 2 (A) and (B) which shows the performance of the three algorithms on two *typical* binary classification tasks among the 190. The figure shows how the performance depends on the number of documents in the train set. Fig. 2 (A) shows that naive Bayes is slightly

[5] A weighting scheme is composed of two parts, for example `ntn.lnc` or `atc.atc` (SMART Information Retrieval System notations). The first group of three letters word describes the feature space transformation for the documents in the train set, whereas the second group describes the feature space for the test documents.

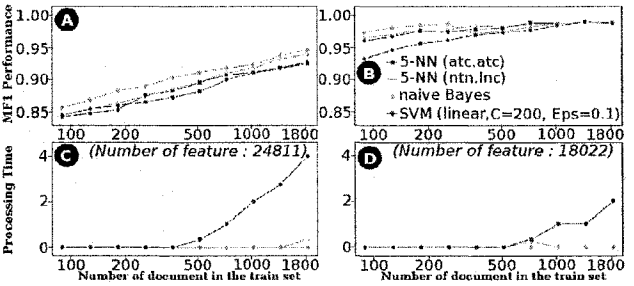

Fig. 2. *Performance (A), (B) and processing time (C), (D) of kNN, SVM and naive Bayes for two problems :* `comp.graphics` *vs.* `comp.sys.ibm.pc.hardware` *(A), (C) and* `comp.os.ms-windows.misc` *vs.* `talk.politics.misc` *(B), (D) for an increasing number of documents in the train set.*

better than the other algorithms for all train sizes. However, the difference from the worst performer is not very large (about 2 or 3%).

Fig. 2 (B) shows that naive Bayes starts with an advantage when a small number of documents are used in the train set, but then as the number of documents increases, the difference diminishes. When 1800 documents are used, the performance is virtually identical to the other classifiers. SVM is however in a disadvantage, when we consider the processing times. These are not only much higher than for the other algorithms, but also, the processing time tends to grow quadratically with the number of documents in the train set (see Fig. 2 (C), (D) and Fig. 4 (D)).

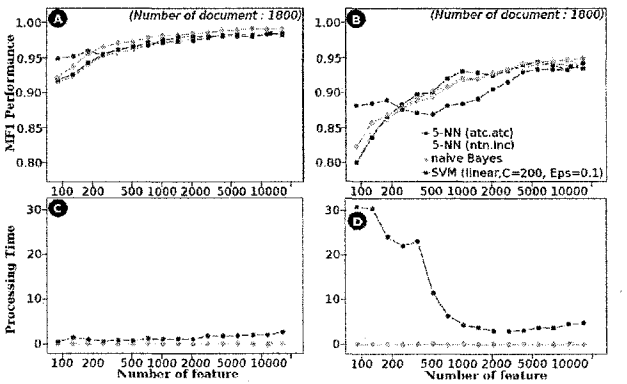

Fig. 3. *Performance (A), (B) and processing time (C), (D) of kNN, SVM and naive Bayes for two problems :* `rec.motorcycles` *vs.* `sci.med` *(A), (C) and* `comp.sys.ibm.pc.hardware` *vs.*`sci.electronics` *(B), (D) for an increasing number of attribute in the feature space.*

Regards the number of features, all three classifiers tend to achieve better performance on large feature set (see Fig. 3 (A) and (B)). However, the SVM processing time can be particularly high if the number of features is small (see Fig. 3 (D) and Fig. 4 (C)). Besides, regards performance of SVM an interesting pattern can be observed on some tasks (see Fig. 3 (B) and Fig. 4 (A)). First, a maximum is reached for a relatively small feature set. Then, the performance decreases until it reverses its tendency again.

On the problem involving `alt.atheism` and `talk.religion.misc` (Fig. 4), both three classifiers achieved relatively poor performance when compared to other classification tasks. In fact, as the two newsgroups are closely related, it is difficult to determine to which category the documents belong. In this task, SVM outperforms naive Bayes and kNN for small feature spaces (see Fig. 4 (A), 100-200) whereas it performs poorly on large feature spaces (500-20000). Although this behavior is specific to this task, it is still a surprising result. Indeed, it is often said that SVM deals well with large number of features. It appears that naive Bayes and kNN do this better here. However, it could be taken into consideration when constructing the learning curves. For instance, the learning curve of SVM shown in Fig. 4 (B) which uses *all* the possible features (21791), could be pushed up if a smaller feature set was used (200).

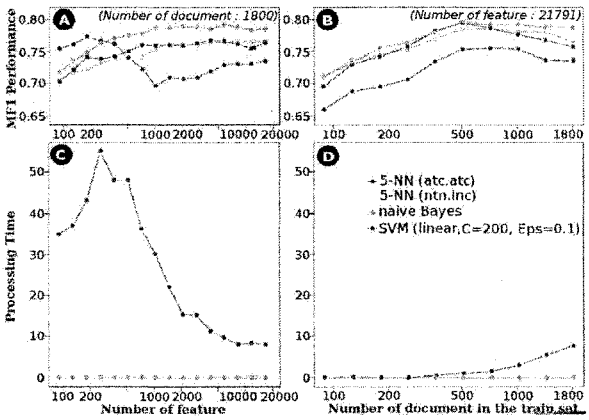

Fig. 4. *Performance (A), (B) and processing time (C), (D) of naive Bayes, kNN and SVM on* `alt.atheism` *versus* `talk.religion.misc`*, given an increasing number of features (A), (C) and an increasing number of documents in the train set (B), (D). right.*

On most of the classification tasks, the training time of SVM increases linearly with the number of features (see Fig. 3 (C)). However, the search for the optimal hyper plane of SVM may require very large training time. For

example, the largest training times among the 190 classification tasks occur on the problem presented in Fig. 4. Indeed, correlating the above-mentioned pattern, SVM training time is higher for small feature spaces than for large ones (Fig. 4 (C) and Fig. 3 (D)). Therefore, training SVM on a extended feature space tends to be faster on these particular tasks.

Discussion.

As explained earlier, comparing a single multi-class naive Bayes (or kNN) to n SVM classifiers (n the number of categories) is definitively not fair for naive Bayes (or kNN). However, this is the approach followed in some published comparative studies [2, 12, 14].

In [2], the SVM SMO version of PLATT was claimed to outperform naive Bayes and other learning methods. However, the optimal number of features was not investigated for each classifier. Indeed, 300 features were selected for SVM which may not be far from the optimal setting. But only 50 were used for naive Bayes. According to our results naive Bayes performs much better with large number of features. Also, the Mutual Information (MI) was used to select features which may not be the best option according to [13]. Finally, they studied the set of *one-against-all* classifiers for each type of algorithm. However, this approach tends to obscur the particular behavior of the classifiers on the various classification tasks.

Recently, a study [1] showed that the architectural parameters often have a more significant impact on performance than the choice of individual learning technique. The work presented here also confirms this. This is why we have decided not to do simple classifier comparisons and present tables with performance results. We preferred to compare the general tendencies of different classifiers when certain parameters are varied.

4 Conclusion

Firstly, we showed that kNN and naive Bayes are still worth considering. Both classifiers achieved good overall performance and are much faster than SVM to use. Indeed, the cost to train SVM for large train set is a clear drawback.

Secondly, compared to SVM, both kNN and naive Bayes are very simple and well understood. SVM is however, more appealing theoretically and in practice, its strength is its power to adress non-linear classification taskw. Unfortunately, most of the tasks examined here were not like that. The simplest SVM based on a *linear* kernel and a large error ϵ were found to be sufficient.

We also observed that results highly depend of the adopted methodology. We have focused here on simple binary classification tasks. Regards kNN, the optimal number k of nearest neighbors is interestingly close to the ones used in other comparative studies carried out on different problems.

As our primary objective is to arrive at general conclusions, transferable from one domain to another, we need to validate our results on other document classification tasks. For this purpose, new experiments are actually being carried out. Moreover, if we are interested to recommend a classifier with suitable parameter settings, we should have a good way of characterizing the given documents and develop a good meta-learning strategy.

Acknowledgements. The first author wishes to thank P. BRAZDIL of the LIACC-NIAAD, and also A.-M. KEMPF and F. POULET of the ESIEA Recherche Institute, for having him introduced to his research. We wish to express our gratitude to K. R. PATIL and C. SOARES for their relecture and to all colleagues from LIACC-NIAAD, University of Porto for their encouragement and help. Finally, the Portuguese Pluri-annual support provided by FCT is gratefully acknowledged.

References

1. W. Daelemans, V. Hoste, F. D. Meulder, and B. Naudts. Combined optimization of feature selection and algorithm parameters in machine learning of language. In *Proceedings of the European Conference of Machine Learning,* pages 84–95, 2003.
2. S. Dumais, J. Platt, D. Heckerman, and M. Sahami. Inductive learning algorithms and representations for text categorization. In *Proceedings of the 7th International Conference on Information and Knowledge Management*, pages 148–155, 1998.
3. J. Fürnkranz. Pairwise classification as an ensemble technique. In *Proceedings of the 13th European Conference on Machine Learning*, pages 97–110, 2002.
4. T. Joachims. Making large-scale support vector machine learning practical. In *Advances in Kernel Methods: Support Vector Machines.* 1998.
5. A. McCallum and K. Nigam. A comparison of event models for naive bayes text classification. *AAAI-98 Workshop on Learning for Text Categorization*, 1998.
6. A. K. McCallum. Bow: A toolkit for statistical language modeling, text retrieval, classification and clustering. http://www.cs.cmu.edu/~mccallum/bow, 1996.
7. T. M. Mitchell. *Machine Learning.* McGraw-Hill, 1997.
8. J. Platt. Sequential minimal optimization: A fast algorithm for training support vector machines. Technical Report 98-14, Microsoft Research, 1998.
9. M. Rogati and Y. Yang. High-performing feature selection for text classification. In *Proceedings of the 11th International Conference on Information and Knowledge Management*, pages 659–661, 2002.
10. Y. Yang. An evaluation of statistical approaches to text categorization. *Information Retrieval*, pages 69–90, 1999.
11. Y. Yang. A scalability analysis of classifiers in text categorization. In *Proceedings 26th ACM International Conference on Research and Development in Information Retrieval*, 2003.
12. Y. Yang and X. Liu. A re-examination of text categorization methods. In *Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 42–49, 1999.
13. Y. Yang and J. O. Pedersen. A comparative study on feature selection in text categorization. In *Proceedings of the 14th International Conference on Machine Learning*, pages 412–420, 1997.
14. T. Zhang and F. J. Oles. Text categorization based on regularized linear classification methods. *Information Retrieval*, pages 5–31, 2001.