Text Categorisation with Support Vector Machines and Feature Reduction*

Irene Díaz[1], Elías F. Combarro[2], Elena Montanés[1], José Ranilla[1]

[1] Artificial Intelligence Center, University of Oviedo. Campus de Viesques, Gijón (Asturias) Spain
[2] Computer Science Department, University of Oviedo. Campus de Viesques, Gijón (Asturias) Spain

Abstract. The automatic classification of texts into predefined categories has became crucial due to the availability of documents in electronic form. The most modern approach to classification text is Machine Learning, where Support Vector Machines (SVM) have proven to be an efficient technique. This paper explores the use of Support Vector Machines for learning text classifiers from examples. It analyses the influence of different methods of feature reduction when SVM are used to classify the Reuters-21578 document set and compares them according to precision/recall parameters.

1 Introduction

Text categorisation can be defined as the action of labelling natural language text with respect to a fixed set of categories. Text categorisation is an important task in the management and retrieval of information.

From a formal point of view, text categorisation consists of assigning a boolean value to each pair $< d_j, c_i > \in \mathcal{D} \times \mathcal{C}$, where $\mathcal{D}$ is a domain of documents and $\mathcal{C} = \{c_1, ..., c_n\}$ is a set of n predefined categories [15]. Two main processes must be carried out in order to categorise a document set. The first one is feature extraction and the second one is the labelling assignment of the documents according to the previously extracted features.

Within feature extraction it is possible to distinguish two steps. The first one is to transform documents into a representation suitable for the categorisation task. This text transformation usually involves the removing of HTML, XML or any other tags, the removing of stop-words and the stemming of the words forming the text. The second step is the document indexing, that is the representation of a certain document according to some words describing the content of a document. The main issue of indexing is to reduce the size of the representation of a document. This task is the main objective of this paper.

Since the 60's some different approaches to text categorisation have been developed. These approaches come mainly from statistics (regression models,

* The research reported in this paper has been supported in part under MCyT and Feder grant TIC2001-3579 and FICYT grant BP01-114.

nearest neighbor classifiers, Bayesian classifiers) or machine learning (symbolic learning algorithms, relevance feedback, neural networks and support vector machines).

The aim of this work is to study the behavior of support vector machines when performing some different feature reductions.

This paper is organized as follows. In Section 2 we discuss previous work. In Section 3 we describe the different feature reduction techniques that will be applied. We also briefly describe the support vector machine algorithm as well as describing the experiment. In Section 4 we present the results of the experiments and finally, in Section 5 some conclusions are presented.

2 Previous work

Many algorithms can be used to categorise a corpus. Most of the research in text categorisation have been focused on *binary problems*, where a document is classified as either *relevant* or *non relevant* with respect to a predefined query. However, there are many problems where a *multi-label* categorisation is needed. The most common method to cope with a n multi-class problem is to break it into n disjoint binary categorisation problems. Each binary problem separates one class from the rest following the *one-against-all* method ([14]) In the following section we briefly show some of the most well-known algorithms that carry out text categorisation (TC) task.

2.1 Algorithms for classifying a document set

One classic categorisation algorithm is Rocchio's one. This algorithm works by computing the distance between each class prototype vector c_i and the document vector d ([1]).

K-nearest neighbor classifies a document vector by ranking the document's neighbors among the training document vectors, and predicting the class of the input documents by using the labels of the k most similar neighbors.

Until the late 80's, the most common approach to TC was *knowledge engineering*. This approach involves the manual definition of a set of rules encoding the expert knowledge about how to classify documents according to the predefined categories ([15]).

Since the early 90's *machine learning* has emerged as the most useful tool to classify text. The *machine learning* approach needs a set of preclassified documents $\mathcal{D} = \{d_1, ..., d_s\}$ (called *train set*) under $\mathcal{C} = \{c_1, ..., c_n\}$ to usually build a decision tree. When the user needs to classify a new document, the document d_* is matched against the decision tree previously built. Well-known machine learning algorithms can be applied to TC C4.5 [11], ID3[3] or ARNI [12].

In the 00's *Support vector machines* [5] have been proved to be a good classifier in the TC environment (and in most classification problems). The SVM integrates both dimension reduction and classification tasks. It is only applicable to binary classification tasks. Support vector machines are based on the

Structural Risk Minimization principle from computational learning theory [18]. SVMs are universal learners able to find out linear or non-linear threshold functions to separate the examples of a certain category from the rest. One important property of SVM theory is that their are supposed to be independent of the dimensionality of the feature space. This characteristic is studied in this paper in order to check if a well chosen dimensionality reduction improves the efficiency of SVM.

2.2 Feature selection

Before classifying a document set, it is necessary to represent its documents in a suitable way; the most commonly used document representation is the vector space model [13] where documents are represented by vectors of words. Thus, a document set is represented by matrix A where each element a_{ij} of A represents the weight of word i in document j.

There are several ways of determining the weight a_{ij}. The simplest approach is to assign a weight 1 if a word occurs in a document and 0 otherwise. Another simple approach is to use the absolute frequency of the word in the document. Just another approach is the *tf-idf-weighting* [13] which assigns the weight a_{ij} in proportion to the number of occurrences of the word in the document, and in inverse proportion to the number of documents of the collection in which the words occurs at least once. Other measures of this kind are *tfc-weighting*, *Itc-weighting* or *Entropy weighting* [1].

The selection of the subset of features to be used is a key step in text classification in order to reduce the dimensionality of the space and to remove words without meaning (usually known as *stop words*). The simplest feature selection is to select as representative features the words whose absolute frequency in a document belongs to a fixed interval. There are a lot of more complex measures as *Information gain* [11] which counts the number of bits of information for category prediction knowing the presence (or absence) of a word in a document, *Expected cross entropy* [7], *Mutual information* [19], *Weight of evidence for text* [8] and *Odds ratio* [17].

3 Description of the experiments

3.1 The corpus

We have conducted all our experiments on the Reuters-21578 corpus. This a collection of short economy-related news published by Reuters in 1987. It is publicly available[3] and has been extensively used in previous research about TC.

The stories on the collection were manually classified into at least one of the 135 fixed categories, resulting in an unbalanced distribution. Some categories

[3] The files containing the Reuters-21578 collection can be reached at http://www.research.att.com/~lewis/reuters21578.html

have several hundreds (and even thousands) of documents assigned to them, while others have none.

The collection of documents has been splitted on train and test sets by Lewis and later on by Apte ([2, 9]). We have chosen to use this latter splitting, which should give us 9603 train documents and 3299 test documents after eliminating those documents not assigned to any category. However, as it has already been noticed ([1]), there are some irregularities among the documents selected by Apte and Lewis. Namely, there exist documents which are said to be classified but they are really not, and documents which have no body text. Clearly, these documents are not useful for the learning task, so we have decided to drop them. Then, we finally work with 7063 train documents and 2742 test documents splitted on 90 categories. There is a difference with previous works ([1]) on the number of selected test documents, but these figures are easily checked to be correct using, for instance, an utility like *grep*[4].

3.2 Representation of the documents and filtering

We have used the usual representation of documents as *bags of words* ([13]). It simply consists of representing a document by the set of the words occurring in it. This is the most used representation in TC ([13]).

To better study the influence of the dimensionality reduction we have chosen two different initial vocabularies as feature spaces. The first one consists of those words occurring in any train document. The second one is particular for each category and consists only of those words occurring in the train documents assigned to the category. This latter approach, which seems very natural since it is more likely that the relevant words for a category occur *within* the category itself, involves an extra reduction of the dimensionality of the feature space, so we will have another parameter to test the effects of filtering words.

In both cases, before selecting the vocabularies, we eliminated a list of words with empty meaning (also known as *stop words*) including articles, common verbs, adverbs, etc. We also reduced the words to their *stems* applying the Porter stemming algorithm ([10]).

To perform the reductions, we have ordered the vocabularies according to the absolute frequency of appearance of words and applied different levels of filtering ranging from 75% to 98% . In each case, we kept the most frequent words and removed the others.

We have also performed experiments in which we made no filtering at all in order to study the impact of the removal of words compared to the case in which all available information is considered.

3.3 SVM and categorisation

We have divided the task of assigning categories to documents in a number of binary decisions. Given a document, for each possible category we decide

[4] *grep* is a computer program usual in Linux and other flavors of Unix, which allows to search files for the appearance of text strings and regular expressions.

whether the document belongs to the category or not. To make this decision we use the model obtained for each category after training SVM with the 7063 train documents.

The training of SVM has been performed with the default parameters for the SVM-light package[5].

4 Results

4.1 Evaluating the performance

For the quantification of the performance of different systems in TC, two measures of effectiveness commonly used in Information Retrieval(IR) have been widely adopted ([15]). These measures are precision and recall. To define these notions, we first need to introduce some auxiliary concepts.

Given a category and a categorisation system, a document is said to be a *true positive* (**TP**) if the system says that it belongs to the category and this is true (because the document was previously labelled as *positive* by an expert). If the system classifies it as belonging to the category but it doesn't, it is called a *false positive* (**FP**). If the system doesn't assign it to the category but it belongs to it, we have a *false negative* (**FN**). Finally, if the system says that the document doesn't belong to the category and, in fact, it doesn't, we have a *true negative* (**TN**).

Now, we define the *precision* of a categorisation system for a category C on a fixed test set as

$$Pr_C = \frac{\#TP}{\#TP + \#FP}$$

where $\#TP$ is the number of true positives in the test set and $\#FP$ is the number of false positives in it. Note that the precision will be maximum when the system assigns to the category only documents really belonging to it.

We define the *recall* of the system for the category C on a fixed test set as

$$Re_C = \frac{\#TP}{\#TP + \#FN}$$

where $\#FN$ is the number of false negatives in the test set and $\#TP$ is as above. The recall will be maximum when the system assigns to the category every document belonging to it.

To compute the global performance of a system over all the categories there exist two different approaches. We could either compute the average precision and recall over all categories (what is called *macroaveraging*) or consider the number of documents in each category and compute the average in proportion to these numbers (this is called *microaveraging*). There has been some controversy about which of these approaches is more adequate ([15]). We won't enter the

[5] SVM-light is an efficient implementation of SVM that can be found at htpp://svmlight.joachims.org

discussion here, but since we are dealing with an unbalanced distribution of documents into categories it seems more natural to take it into account and use microaveraging.

It is well known that there exists a trade off between the precision and the recall of a system ([15]) and then it makes no sense to consider these parameters on their own. Several ways to combine their values have been proposed ([17]), the family of functions F_α being among the most popular. These functions are defined as

$$F_\alpha = \frac{1}{\alpha \frac{1}{Pr} + (1-\alpha)\frac{1}{Re}}$$

where α is a parameter ranging from 0 to 1. The value of α expresses the degree of relevance given to precision and recall. The nearer α is to 1, the greater the importance of precision is. Conversely, the more similar α is to 0, the more important recall becomes.

In our framework, where neither precision nor recall are clearly more important than the other, the most adequate choice for α seems to be 0.5, giving equal relevance to both. Then resultant measure is commonly known as F_1 (instead of the more correct denomination $F_{0.5}$) and defined as

$$F_1 = \frac{1}{0.5\frac{1}{Pr} + 0.5\frac{1}{Re}}$$

We will also use a measure common in Machine Learning ([11]), *accuracy*, which is just the proportion of correct decisions (either of assign or not assign a document to a category) to the total number of documents. In symbols

$$Acc_C = \frac{\#TP + \#TN}{\#TP + \#TN + \#FP + \#FN}$$

Again, we will microaverage the results obtained in all the categories.

4.2 The results

In tables 1 and 2 and in figures 1 to 4 we summarize the values of the measures introduced in the previous section obtained in our experiments.

It is quickly noted (see table 1 and table 2) that the reduction of the feature space seems to have little effect on the accuracy, which remains more or less constant in all experiments, both when the vocabulary is considered category by category or in all the corpus. However, the accuracy decreases when the filtering is more aggressive (95% and 98%).

As we can see in figure 1, when the vocabulary is taken from all the documents, a moderate filtering (75%) results on an slight improvement in recall when comparing with the case in which we do not filter at all (0% of filtering). However, with greater reductions we obtain a considerable loss. This loss is a constant in precision (figure 2), where there exists only an isolated case of improvement (92%). In both cases the more aggressive the filtering is, the greater

Parameters	MICROAVERAGE (total)							
	0%	75%	80%	85%	90%	92%	95%	98%
accuracy	99,60%	99,60%	99,60%	99,60%	99,60%	99,60%	99,59%	99,51%
precision	92,98%	92,89%	92,84%	92,77%	92,93%	93,07%	92,88%	92,15%
recall	77,30%	77,38%	77,21%	77,12%	77,15%	76,86%	76,50%	70,96%
F1	84,41%	84,43%	84,30%	84,22%	84,31%	84,19%	83,90%	80,18%

Table 1. Microaveraging with total filtering

Parameters	MICROAVERAGE (filtering by categories)							
	0%	75%	80%	85%	90%	92%	95%	98%
accuracy	99,60%	99,58%	99,85%	99,53%	99,52%	99,47%	99,45%	99,08%
precision	92,98%	92,29%	92,30%	92,19%	92,39%	91,61%	91,61%	91,54%
recall	77,30%	78,70%	78,81%	78,63%	79,18%	78,94%	78,98%	76,95%
F1	84,41%	84,95%	85,03%	84,87%	85,28%	84,80%	84,83%	83,61%

Table 2. Microaveraging with filtering by categories

the loss becomes, specially in recall. This is natural, since when we consider all documents in the corpus, those categories with more examples will have a bigger influence than those with only a few. Then, those words particular to the biggest categories will be the most frequent, while words representative of others (and necessary to correctly classify their documents) could be removed when aggressive reductions are made.

However, the aggregated performance of the system (as measured by F_1) is more or less constant, and is even improved when a moderate filtering is performed (see figure 3 and table 1).

Let's now turn to analyze the results in the cases where the vocabulary is chosen category by category. In table 2 we observe that precision is always smaller than in case where no reduction is made. Again, it becomes worse with the level of filtering.

The situation with recall is quite different, as shown in figure 1. There is always a significant improvement (excepting the highest level of filtering) and it reaches a maximum with a level of filtering of 95% .

This improvement in recall results in an overall improvement and the value of F_1 (figure 3) is always bigger than that obtained with no filtering at all (again, excepting the too aggressive filtering of 98%).

5 Conclusions and future work

It is a widely accepted fact in IR ([17]) that some words are more relevant than others to understand or classify a document and even that some infrequent (or too common) words can introduce noise and misguide the algorithms that perform these tasks.

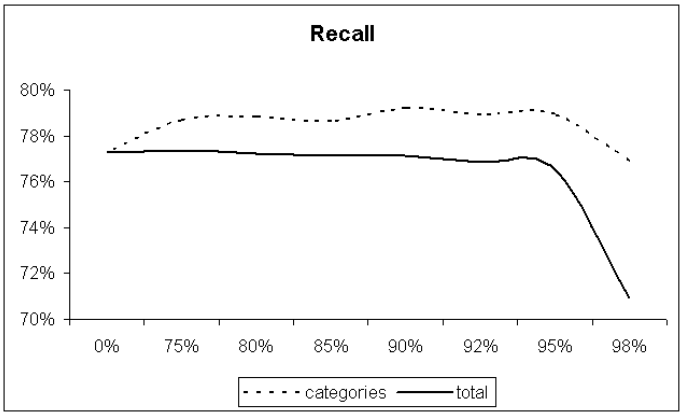

Fig. 1. Recall

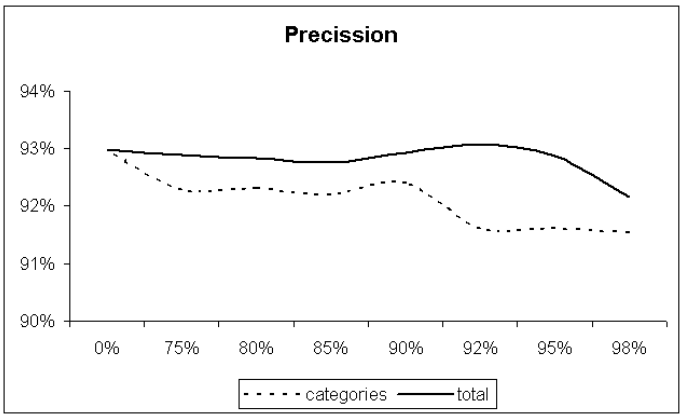

Fig. 2. Precision

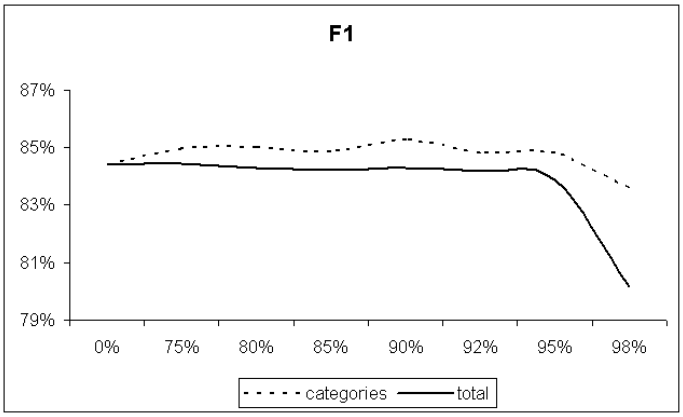

Fig. 3. F1

However, it had been argued ([5, 6]) that the special properties of the SVM learning model (namely, the capacity of dealing with a huge number of features) could overcome the necessity of feature space reduction in TC.

The experiments shown here suggest that this may not be true in all the cases. At least in the special circunstances considered here, the simple removing of non-frequent words could help in the automatic classification of documents even if it is performed with Support Vector Machines. In particular, it seems that considering vocabularies local to each category could be a way of improving the overall performance of the classification and, specially, its recall.

Of course, the study presented here is quite preliminar and incomplete and without further investigation it is impossible to know to what extent the results we have got depend on the particular representation of the documents chosen and on the weighting scheme that we have adopted.

It may well be the case that with more informative measures of the relevance of the words of the corpus (for instance information gain or tf-idf, see [19]) all the features are needed to achieve the best classification results.

Then, it seems interesting to repeat these experiments varying several parameters, like document representation, weighting scheme, and internal parameters of the SVM.

It would also be interesting to adopt other measures of evaluation (like the breakeven point or the 11-point average precision [15]) to see if the improvements and their trend remain. Finally, the hypothesis drawn here should be tested on other document collections before it can be accepted that feature reduction can help SVM in the classification task.

References

1. Aas K., Eikvil L.: Text categorisation: A survey. Technical report, Norwegian Computing Center (1999)
2. Apté C., Damerau F., Weiss S.: Automated learning of decision rules for text categorization. ACM Transactions on Information Systems,**12** (1994) 233-251
3. Colin A.: Building decision trees with the ID3 algorithm Dr. Dobbs Journal (1996)
4. Cortes C., Vapnik V.: Support-vector networks. Machine Learning, **20** (1995) 273-297
5. Joachims T.: Text categorization with support vector machines: Learning with many relevant features Proc. 10th European Conference on Machine Learning, Springer Verlag (1998)
6. Joachims T.: Text categorization with support vector machines: Learning with many relevant features. University of Dortmund (1997)
7. Koller D., Sahami M.: Hierarchically classifying documents using very few words. Proc. of the 14rd International Conference on Machine Learning, (1997) 170-178
8. Kononenko I.: On biases estimating multi-valued attributes. Proc. of the 14th International Conference on Artificial Intelligence, (1995) 1034-1040
9. Lewis D.D., Ringuette M.: A comparison of two learning algorithms for text categorization. Proceedings of SDAIR-94, 3rd Annual Symposium on Document Analysis and Information Retrieval, (1994) 81-93
10. Porter M. F.: An algorithm for suffix stripping. Program, **43(5)** (1980) 130-137
11. Quinlan J.R.: Constructing decision tree in C4.5. Programs of Machine Learning, Morgan Kaufman Publishers (1993) 17-26
12. Ranilla J., Bahamonde, A.: Fan: Finding accurate inductions. To appear in International Journal of Human Computer Studies (IJHCS)
13. Salton G., McGill M.J.: An introduction to modern information retrieval. McGraw-Hill (1983)
14. Scholkopf B., Burges C., Vapnik V.: Extracting support data or a given task. Proceedings of the 1st International Conference on Knowledge Discovery and Data Mining, AAAI Press (1995)
15. Sebastiani F.: Machine Learning in Automated Text Categorisation ACM Computing Surveys, **34(1)** (2002)
16. Van Rijsbergen C.J.: Information retrieval (Second edition) Butterworths, London, U.K. (1979)
17. Van Rijsbergen C.J., Harper D.J., Porter M.F.: The selection of good search terms. Information Processing and Management, **17** (1981) 77-91
18. Vapnik V.: The Nature of Statistical Learning Theory. Springer, New York (1995)
19. Yang T., Pedersen J.P.: Feature selection in statistical learning of text categorization. 14th Int. Conf. on Machine Learning, (1997) 412-420