A Scenario for the Comparison of Algorithms in Text Classification

In this chapter, we describe a data mining scenario for the comparison of algorithms in text classification. We start by introducing the problem of automatically classifying text documents into categories. Then, we consider the problem of doing fair classifier’s comparisons. Next, we describe the three algorithms that we compare: the k nearest neighbors classifier, naive Bayes and the Support Vector Machines. Last, we define the settings of our scenario and the data on which we performed our experiments on.

6.1 Introduction

The aim of text categorization is to build systems which are able to automatically classify documents into categories.

To build text classification systems, the bag of words representation is the most often used feature space. Its popularity comes from its wide use in the field of information retrieval and from the simplicity of its implementation. Yet, as in the bag of words representation each dimension corresponds to the number of occurrences of the words in a document, the task of classifying text documents into categories is difficult because the size of the feature space is very high. In typical problems, it commonly exceeds tens of thousands of words. Another aspect that hampers this task is the fact that the number of training documents is several orders of magnitude smaller than the size of the feature space.

Among the algorithms applied to text classification, the most prominent one is linear Support Vector Machines. First introduced to text categorization by Joachims [Joa98], Support Vector Machines (SVM) were systematically included
in subsequent comparative studies [Dum98; Yan99b; Zha01; Yan03]. Their conclusions suggest that SVM is an outstanding method for text classification. In his large scale study, Forman also confirmed SVM as outperforming other techniques for text classification [For03].

So should we just not bother about other classification algorithms and opt always for SVM?

As shown in [Dae03], set-up parameters can have a more important effect on the performance than the individual choice of a particular learning algorithm. Indeed, classification tasks are often highly unbalanced and the way training documents are sampled has a large impact on performance. In fact, in several large studies, SVM did not systematically outperform other classifiers. For example the work of [Liu05] showed the difficulty to extend SVM to large scale taxonomies. Others showed that depending on experimental conditions, the nearest neighbors classifier or naive Bayes can achieve better performance [Dav04; Sch06]. We also reported such results [Col06b]. On top of that, selecting the right parameters of the SVM, e.g. the upper bound of the Lagrange multipliers (C), the kernel, the tolerance of the optimizer (ε) and the right implementation are non-trivial issues which are seldomly investigated thoroughly.

In this thesis, considering globally the task of classifying text documents, we present a data mining scenario (together with its results when applied) to compare text classification algorithms, see Figure 6.1.

6.2 Conducting fair classifier comparisons

Although naive Bayes and the k nearest neighbors classifier are multi-class classifiers, the SVM are by default binary classifiers. Then, to handle multi-class problems, SVM usually relies on a one versus all strategy where as many binary classifiers as there are classes are trained. For instance, in the case of a classification problem with n-classes, n one versus the rest binary classifiers are trained.

Therefore, when running experiments on complex classification tasks involving more than two-classes, we are actually comparing n SVM classifiers (for n classes) to single multi-class naive Bayes or k nearest neighbors classifier. We consider this unfair.

Moreover, Fürnkranz has shown that a round robin approach using a set of one against one binary classifiers performs at least as well as a one versus all approach [Für02]. Therefore, we do not limit the generality of the results by studying only one against one classification problems.

In addition, to observe and compare the behaviors of the classifiers when experimental conditions are varying, these conditions must be controled precisely. Indeed, the properties of the training set can influence largely the learning abilities
of the classifiers, while in multi-class problems, it can be difficult to understand the particular influence of each class on the classifier’s behaviors.

Therefore, in our scenario, we focus on problems with only two-classes. First, it enables us to discard the influence of the multi-class aggregating algorithm in the case of SVM and thus, to compare SVM more fairly with naive Bayes and the $k$ nearest neighbor classifier. Second, it also gives the possibility to control more carefully the properties of the training set. In that regard, in order to give to both classes the same chance to be learned as well, we only studied situations where the number of training instances is the same in each class. Last, as binary problems are smaller than multi-class problems, they are usually easier and faster to learn, thus facilitating the conduction of experiments.

### 6.3 Classification algorithms

Because of their simplicity and their generally good performance reported in text categorization, we compare the SVM with two well known classifiers, namely the $k$ nearest neighbors classifier and naive Bayes. In the following, we first introduce some general notations and then, we introduce the three classifiers formally.

Consider a database of instances $\mathbf{x}_i$ and class membership $y_i$, $i = 1, ..., N$ and $d$ the dimension of the feature space, i.e. the dimension of $\mathbf{x}_i$. Denote by a function $\Phi$ the mapping in the database between each instance and its class mem-
bership such that \( y_i = \Phi(x_i) \). Considering only binary classification problems, this mapping takes values in \( \mathcal{C} = \{-1, +1\} \). Then, a classification algorithm can learn this mapping by training and we denote the estimated classification function by \( \hat{\Phi}(x_i) \).

### 6.3.1 k Nearest Neighbors.

Given a test point \( x' \) and a predefined similarity metric (\( \text{sim} \)) that can order the training points by their similarity to \( x' \), a \( k \) nearest neighbor classification rule will assign to \( x' \) the class having the highest similarity score. These scores are calculated by summing up the similarities of the \( k \) nearest neighbors in each class. The classification rule compares these scores and return the class having the highest, it is defined as

\[
\hat{\Phi}(x') = \arg\max_{y' \in \mathcal{C}} \sum_{k=1}^{K} \delta(y', \Phi(x_k)) \text{sim}(x_k, x'), \tag{6.1}
\]

with \( K \) the number of nearest neighbors and \( \delta(y', \Phi(x_k)) = 1 \) if \( \Phi(x_k) = y' \), 0 otherwise.

### 6.3.2 Naive Bayes

For \( y' \in \mathcal{C} \), let \( P(y') \) be the prior probability of each class. For \( x_{ij} \) (feature \( j \) of an instance \( x_i \)), let \( P(x_{j|y'}) \) be the probability to observe the feature value \( x_j \) conditionally to \( y' \). Then, given a test point \( x' \) whose feature values are \( (x'_1, ..., x'_d) \), the naive Bayes classification function is expressed by

\[
\hat{\Phi}(x') = \arg\max_{y' \in \mathcal{C}} P(y') \prod_{j=1}^{d} P(x'_j|y'). \tag{6.2}
\]

### 6.3.3 Support Vector Machines

The SVM are based on statistical learning theory [Vap95]. Its theoretical foundations together with the results obtained in various fields makes it a popular algorithm in machine learning.

The SVM classification function of a test point \( x' \) is given by

\[
\hat{\Phi}(x') = \text{sign}(\langle w, x' \rangle + b) \tag{6.3}
\]

with \( w \) and the scalar \( b \), the coordinates of the separating hyperplane and the bias to the origin. The particularity of this hyperplane \( w \) is that it is the one separating the points of the two classes with the maximum distance when these
are linearly separable. This concept of maximum separating distance is formalized by the *geometrical margin* which is defined as

$$\gamma = \frac{1}{2||w||_2}.$$  \hfill (6.4)

Therefore, the SVM problem resides in searching the maximum of $\gamma$ or, alternatively, the minimum of $||w||_2$ given the constraints. To identify this $w$, an optimization problem must be solved. Its primal form is expressed by

$$\min_{w,b} \quad \frac{1}{2}\langle w,w \rangle,$$

subject to $y_i(\langle w,x_i \rangle + b) \geq 1, i = 1,...,N,$ \hfill (6.5)

where the $x_i$ are the training instances in the database. Yet, as the number of training documents is typically several orders smaller than the number of features in text classification, it is usually preferred to operate the SVM in the dual (that depends on the number of training documents). The dual form can be obtained by posing and deriving the Lagrangian according to $w$ and $b$:

$$\max_{\alpha} \quad \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} y_i y_j \alpha_i \alpha_j \langle x_i,x_j \rangle,$$

subject to $\sum_{i=1}^{N} y_i \alpha_i = 0$

$$0 \leq \alpha_i \leq C, i = 1,...,N.$$ \hfill (6.6)

In order to limit values of Lagrange coefficients $\alpha_i$, an upper bound $C$ is introduced so that each training instance has a maximum contribution when classes are not linearly separable. This type of SVM is referred to as soft-margin SVM.

Concerning the kernel function, even though problems may not always be separable in text classification, a *linear* kernel is commonly regarded as yielding the same performance as non-linear kernels in our text classification domain [Yan99b]. For this reason, we only considered linear kernels in our scenario.

Next, recall that the hyperplane $w$ is defined by

$$w = \sum_{i=1}^{N} y_i \alpha_i x_i.$$ \hfill (6.7)

Then, upper-bounding the Lagrange multipliers gives the constraints

$$0 \leq \alpha_i < C.$$ \hfill (6.8)

Observe that the norm of the hyperplane tends to vanish as $C$ goes to zero

$$\lim_{C \to 0} ||w||_2 = 0.$$ \hfill (6.9)
This implies that the geometrical margin goes to infinity ($||w||_2 > 0$)

$$\lim_{C \to 0} \gamma = +\infty.$$  \hfill (6.10)

Consequently, lowering $C$ to very small values will eventually lead to a SVM solution where all training instances are within the margin. We further discuss the quality of this SVM solution in coming sections.

**Interpreting the SVM solution** Recall that a set of points is *convex* if the line segment between any two of its points stays within the set [Str86], and consider the smallest of the convex hulls (the smallest convex set). Then the solution of SVM in a binary classification problem is made of the training points on the smallest convex hulls of the two classes. We can regard these particular points as defining the boundary of the two classes.

In the solution of SVM, the Lagrange multipliers $\alpha_i$ quantify the contribution of each training point to the positioning of the hyperplane. The higher the $\alpha_i$, the more force the point exerts on the position of the hyperplane. Thus, the points within the smallest convex hull of their respective classes are set *inactive* with $\alpha_i = 0$. Those other points $x_i$ for which $\alpha_i > 0$ are on the convex hulls. They are considered as *active* and referred to as support vectors (SV).

If the two classes are linearly separable, the solution of linear SVM will be the hyperplane in *force equilibrium* between the two convex hulls. This constraint is further discussed in the following paragraph *Settings of text classification*. However, when classes are not linearly separable, points may exert high pressure on the hyperplane without ever being on the right side of it. Consequently, some multipliers may be very large compared to others or even infinite. In order to limit the individual contribution of the multipliers, the so-called soft margin was introduced. In a soft margin SVM solution, the multiplier values are upper bounded by a parameter $C$, that is the maximal cost that we are ready to pay to classify a training point well. There are four types of training points. We list and characterize them by their distance and their contribution to the hyperplane in Table 6.1.

**Table 6.1:** The different types of training instances composing an SVM solution.

<table>
<thead>
<tr>
<th>Distance</th>
<th>Contribution</th>
<th>Active?</th>
<th>Well classified?</th>
</tr>
</thead>
<tbody>
<tr>
<td>$y_i(\langle w, x_i \rangle + b) \geq 1$</td>
<td>$\alpha_i = 0$</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>$y_i(\langle w, x_i \rangle + b) = 1$</td>
<td>$0 &lt; \alpha_i &lt; C$</td>
<td>yes, in bound</td>
<td>yes</td>
</tr>
<tr>
<td>$0 &lt; y_i(\langle w, x_i \rangle + b) &lt; 1$</td>
<td>$\alpha_i = C$</td>
<td>yes, at bound</td>
<td>yes</td>
</tr>
<tr>
<td>$y_i(\langle w, x_i \rangle + b) &lt; 0$</td>
<td>$\alpha_i = C$</td>
<td>yes, at bound</td>
<td>no</td>
</tr>
</tbody>
</table>

Recall that the concept of sparsity aims at finding the most parsimonious representation for a problem. Then, in a sparse SVM solution, most of the training
points are set inactive (1) and the ones that are active (2,3,4) define the smallest convex hull of the two classes. The active points are expected to represent the “boundaries” of the two classes.

In linearly separable problems, there are only training points of types (1) and (2) (without the bound $C$). However, many problems are not linearly separable, which means that the linear separation surface will misclassify part of the training points. Thus, in soft margin SVM, the more a solution has bounded SV’s (3 and 4), the less linearly separable the problem is. In addition, we remark that only the bounded SV of type (4) are misclassified training points, in contrast to the bounded SV of type (3) that are well classified.

Large proportions of bounded SV are not desirable because it shows the non-linear separability of the problem. If training points of distinct classes are at the same location in the feature space, no surface of any complexity can separate well those overlapping training points. Therefore, using non-linear kernels would not show any improvement.

**Settings of text classification** In addition to large number of features, the *bag of words* feature space exhibits high levels of sparsity. The majority of the word occurrences are zero. As the dimensionality of the problem increases, there will be more training points on the smallest convex hull of the two classes. As an example, more than 75% of the dual variables are non-zero in the SVM solutions of [Rou06] in text classification. We will also illustrate this phenomenon through our experiments.

Next, consider the *force equilibrium* between the two classes formalized by the constraint

$$\sum_{y_i \in \{+1\}} \alpha_i = \sum_{y_j \in \{-1\}} \alpha_j, \quad (6.11)$$

where the sum of the individual training point forces should remain equal for the two classes. Then, a specific SVM solution is the one where all the training points are equally weighted. We refer to it as the *nearest mean classifier* solution.

Our experiments suggest that the best performing SVM solutions in large *bag of words* feature spaces are solutions that are similar to the *nearest mean classifier* because most of the training points have equal weight.

**Setting the parameters of SVM** First, although we tried several values for the parameter $\epsilon$ which controls the tolerance of the stopping criterion of SVM, we selected $\epsilon = 0.1$. In fact, while no effect on the performance was observed for other settings, this setting significantly reduced the training time. Second, concerning the $C$ parameter, it is seldom optimized in text categorization and our scenario will investigate its effect in Chapter 8, both on the performance and on the type of SVM solutions.
6.3.4 Implementation of the algorithms

For naive Bayes and the $k$ nearest neighbors classifier, we used the libbow library [McC96]. With respect to SVM, we used both the Platt’s SMO algorithm in libbow and the libsvm implementation of [Cha01]. Therefore, in this thesis, our conclusions on SVM do not relate to a particular implementation because we could reproduce them for two different implementations.

6.4 Definition of the scenario

In this section we describe our data mining scenario for the comparison of algorithms in text classification.

The remainder of the section structures as follows. First, in order to compare the classifier’s behaviors, we describe our evaluation methodology and its measures. Second, we present the different dimensions of our experimental set-up.

6.4.1 Evaluation methodology and measures

To improve the reliability of our comparative experiments between classification algorithms, we chose an evaluation methodology that, for each experimental condition, repeats the training of the classifiers a number of times. Then, under each experimental condition, we took a set of measurements to picture the classifier’s behaviors and finally, we calculated aggregates of these measurements (e.g. the empirical mean).

In the following, we first describe our evaluation methodology and then, our measures.

**Evaluation methodology** We adopted the 10-fold cross-validation methodology to evaluate the classifier’s behaviors. It proceeds as follows. First, the complete database of instances is separated in 10 folds. Second, by a mechanism similar to a rotation, each fold is successively considered as test set whereas the remaining of the instances composes the set of instances available for a training set. Indeed, we will show in the next section that the set of available training instances will be sub-sampled.

**Measures** We are interested in the ability of the classification model to predict correctly the class of an instance. To assess this, as illustrated in Table 6.2, we group the errors made by the classifiers using a confusion matrix.

From this confusion matrix, we measure the precision of the classification model that is, the accuracy to predict a specific class. Considering the target class as $A$, then the precision is defined by

$$Precision_A = \frac{tp_A}{tp_A + e_{BA}} \quad (6.12)$$
Table 6.2: Confusion matrix in a two class problem.

<table>
<thead>
<tr>
<th>Known class</th>
<th>Predicted class</th>
<th>A</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>tp_A</td>
<td>e_{AB}</td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>e_{BA}</td>
<td>tp_B</td>
<td></td>
</tr>
</tbody>
</table>

The recall is a measure of the ability of a classification model to retrieve the appropriate instances of a certain class in a dataset. Again, considering A as the target class, it is defined by the formula

\[ \text{Recall}_A = \frac{tp_A}{tp_A + e_{AB}} \]  

(6.13)

Similarly to [Yan99b], we adopt for our experiments the macro averaged \( F_1 \) measure which is defined by

\[ maF_1 = \frac{2 \times maPrecision \times maRecall}{maPrecision + maRecall} \]  

(6.14)

where \( maPrecision = (\text{Precision}_A + \text{Precision}_B)/2 \) and \( maRecall = (\text{Recall}_A + \text{Recall}_B)/2 \). In words, the \( maF_1 \) measure relates the precision and the recall computed in two confusion matrices, interchanging the definition of the target class in the two-class problem (either A or B). We calculate the mean of \( maF_1 \) over the ten measures from the cross validation.

Further, we decided to characterize the solutions of SVM in terms of the number of SV within and at bound. For this purpose, we measured these numbers for all experiments whereas, for a selection of experimental conditions, we also measured all the Lagrange multiplier values \( \alpha_i \) in order to compare different types of SVM solutions on their quantiles of \( \alpha_i \). Regarding the number of SV at and within bounds, their empirical mean is estimated.

Last, the global processing time in seconds is recorded in the course of our experiments. This processing time includes both the training and the test time. We used two types of computers in our experiments: Pentium III 1Ghz with 1GB of RAM in Chapter 7 and AMD Opteron dual 2.6Ghz with 4GB of RAM in Chapter 8. All these computers were using the Linux operating system where the graphical interface was disabled.

6.4.2 Dimensions of experimentation

As classifiers are influenced by the number of training documents and by the features chosen, we decided to examine these issues in detail and we compared the classifiers when varying both dimensions.

In the remainder, we first present our strategy to prepare and sub-sample the training sets. Second, we describe how we reduce the size of the feature space.
Then we present our third dimension of experimentation for SVM. Finally, we report which series of values we use to do the experimentations in two (Chapter 7) or three dimensional (Chapter 8) spaces of experimentation.

**Number of training documents** For each classification task, the procedure to prepare the training sets can be structured as follows.

1. The database of instances is separated by class.
2. For each class, the instances are ordered randomly.
3. For each class, the instances are separated into ten folds.
4. For each class and each test fold, the set of instances available for the training set is the merging of the remaining folds.
5. For each class and each test fold, the trainsets are sub-sampled into sets of increasing size.
6. For each sub-sample, the instances from both classes are merged into one of the ten folds.
7. For each sub-sample, the instances are re-ordered randomly.

To study the behaviors of the classifiers when the number of training documents was increasing, the set of available training instances is sub-sampled. The sub-sampling creates training sets of increasing size but with equal number of cases from each class. Thus, both classes were given the same chance to be learned as well. For instance, a training set of size 90 would have 45 instances of both classes. Further, when varying the experimental condition from 90 training instances to 128, we preserved the 90 first and grew the training set of 38 new instances, i.e. 19 from each class.

Following this procedure enabled us to reduce the variability in our experiments, in particular due to our sampling strategy.

**Number of features** To study the influence of varying the size of the feature space on the classifiers, we chose the information gain heuristic as a means to select a subset of features. Our choice stems from the good overall performance of this heuristic as well as its simplicity [Yan97; Rog02]. In the following, we first introduce some notations and then, the information gain.

Denote by \( p_y = P(Y = y) \) the probability to observe an instance from class \( y \in \mathcal{C} \). Recall that \( d \) is the dimension of the bag of words. Denote by \( X = x_{j} \) with \( j = 1, ..., d \) the distribution of the presence / absence of a word \( j \) in a set of instances.

First, a result from information theory states that to classify instances in \( \mathcal{C} \), the optimal algorithm coding the class needs an average number of bits given by

\[
H(Y) = -\sum_{y \in \mathcal{C}} p_y \log_2 p_y.
\]  

(6.15)
This quantity $H(Y)$ is referred to as the entropy of the distribution $Y$. The entropy is high if the distribution of $Y$ is even over all values and low if the distribution is varied.

Second, we can define the average conditional entropy of a distribution $X$ given the one of $Y$ by

$$H(X|Y) = \sum_{y \in \mathcal{C}} P(Y = y) H(X|Y = y). \quad (6.16)$$

This second quantity estimates the average number of bits to code the values of $X$ if the class $Y$ is known. Knowing more about the problem, here the class, may help to identify a more efficient coding scheme that exhibits a lower entropy.

Finally, the information gain is defined as a function of these two quantities by

$$IG(X = x,j|Y) = H(X = x,j) - H(X = x,j|Y). \quad (6.17)$$

It calculates the average number of bits that could be saved when predicting the occurrence of a word $x,j$ if the distribution of $Y$ was known. The higher is the information gain, the more the class $Y$ associates with the occurrence of the word $x,j$.

Therefore, when applying the feature selection heuristic, we search for words in the bag of words feature space that exhibit the highest information gain in the training set. Finally, recall that features are ranked and selected by information gain at the start of each experiment.

**Dimensions of experimentation** We do measurements in two (Chapter 7) or three dimensional (Chapter 8) spaces. These axis are the number of training documents, the number of features and the parameter $C$ of the SVM classifier. The conditions of experimentation are determined by series of exponential values on each axis because the phenomena that we study are not linear.

For both the number of training documents and of features, the values follow the series given by $2^{i+b}$ with $i = 1, 2, 3...$ and $b \in \mathbb{N}$, e.g. the series starts with $\{90, 128, 181, 256, ...\}$ when $b = 6$.

Concerning the values of $C$, they follow the series $10^i$, e.g. $\{0.0001, 0.001, ..., 1000\}$.

### 6.5 Experimental data

This section outlines as follows. First, we describe the text classification datasets used in the comparison of algorithms in Chapter 7. Second, we present the datasets used in the study of Chapter 8 where we investigate SVM’s scale-up in large bag of words feature spaces.
6.5.1 To study the behaviors of the classifiers

For our experiments we used two well known datasets: 20newsgroups and ohsumed-all. The libbow library was used to process the text data [McC96].

1. The 20newsgroups dataset is composed of 20000 newsgroup emails [Het99]. We removed the headers of the emails and no stemming\(^1\) was performed.

2. The ohsumed-all dataset\(^2\) is taken from the Ohsumed corpus which was initially compiled by William Hersh (ftp://medir.ohsu.edu/pub/oshumed/). The dataset is made of 50216 medical abstracts categorized into 23 cardio vascular disease categories. Although Joachims used only the first 10000 medical abstracts for training and the second 10000 for testing [Joa98], in our experiments we use all 50216 documents for the cross validation. To be consistent with our previous processing of the 20newsgroups dataset, we did not perform stemming.

On these datasets, we chose to study the set of one against one binary classification tasks. Thus, the total number of classification tasks on 20newsgroups was

\[
\frac{20(20 - 1)}{2} = 190.
\] (6.18)

For ohsumed-all, because of computing time considerations, we decided to limit our analysis to the first 162 classification tasks taken in alphabetical order out of the

\[
\frac{23(23 - 1)}{2} = 253
\] (6.19)

possible binary problems.

Therefore, we performed experiments on a total of 352 classification tasks.

6.5.2 To study the scale-up of SVM in large bag of words feature spaces

From the previous study which involved 352 tasks, we selected two binary text classification problems where SVM exhibited a performance drop. In order to further validate our conclusions, we also performed additional experiments on the Reuter Corpus Version 1 dataset (RCV1) [Lew04].

This last dataset interests us particularly because it is available in its processed form. The features were extracted and the word frequencies were tfidf-transformed. See [Lew04] for a detailed description of the dataset processing. Because in our previous study, we used the word frequencies to represent the classification tasks, these additional experiments on RCV1 will enable to show the influence of the feature space transformation on the performance and the nature of SVM’s solutions.

\(^1\)Process of reducing words to their root.

\(^2\)http://dit.unitn.it/~moschitt/corpora/ohsumed-all-docs.tar.gz.
6.6. Concluding remarks

In the remaining of this thesis, we will refer to the three classification tasks by the following acronyms:

**20ng** "alt.atheism / talk.religion.misc" from the 20newsgroups dataset.

**C01-C21** "Bacterial Infections and Mycoses / Disorders of Environmental Origin" from the Ohsumed-all dataset.

**RCV1** Reuters Corpus Version 1.

### 6.6 Concluding remarks

We investigate the problem of automatically classifying text documents into categories which relies on standard machine learning algorithms. These algorithms, given a set of training examples, can learn a classification rule in order to further categorize new text documents automatically.

Among the algorithms suggested for use in text classification, the most prominent one is Support Vector Machines and repeatedly, it was shown to outperform other techniques. Yet, we consider that some of the previous comparative experiments of algorithms were not fairly conducted (see the discussion in the section Related work of chapter 7). In fact, other studies [Dav04; Sch06; Col06b] have shown that in some situations, other algorithms like naive Bayes or the $k$ nearest neighbors classifier give better results than SVM.

Therefore, we first introduced the problem of classifying text documents into categories. Next, with respect to previous comparative studies, we discussed fairness issues when comparing algorithms. Then, given this focus, we described our data mining scenario that aims to compare as fairly as possible classification algorithms. It will help us to better understand the problem of classifying text documents into categories.