Soft Comput (2007) 11: 519–530
DOI 10.1007/s00500-006-0080-8

ORIGINAL PAPER

Catarina Silva · Bernardete Ribeiro

On text-based mining with active learning and background knowledge using SVM

Published online: 20 April 2006

Abstract Text mining, intelligent text analysis, text data mining and knowledge-discovery in text are generally used aliases to the process of extracting relevant and non-trivial information from text. Some crucial issues arise when trying to solve this problem, such as document representation and deficit of labeled data. This paper addresses these problems by introducing information from unlabeled documents in the training set, using the support vector machine (SVM) separating margin as the differentiating factor. Besides studying the influence of several pre-processing methods and concluding on their relative significance, we also evaluate the benefits of introducing background knowledge in a SVM text classifier. We further evaluate the possibility of actively learning and propose a method for successfully combining background knowledge and active learning. Experimental results show that the proposed techniques, when used alone or combined, present a considerable improvement in classification performance, even when small labeled training sets are available.

Keywords Text mining · Partially labeled data · Support vector machines

1 Introduction

The Internet phenomenal growth and the wide spread use of computers to store, process, and share text have created the need for tools that help individuals and business find the information they need in the most effective and efficient manner.

C. Silva · B. Ribeiro
CISUC – Departamento de Engenharia Informática – Universidade de Coimbra, Coimbra, Portugal

C. Silva (✉)
Escola Superior de Tecnologia e Gestão, Instituto Politécnico de Leiria, Leiria, Portugal
E-mail: catarina@dei.uc.pt

Applications of text mining are everywhere, since almost 80% of the information available is stored as text, transforming the organization of that information into a complex and vital task [1].

Text mining is a broad topic that has many different applications. Here, we focus our interest on text categorization or classification, which can be defined as the assignment of natural language texts to one or more categories, based on their content. Some authors refer to text categorization strictly as the module that defines the hierarchy of categories, but we prefer the previous wider definition.

Text classification can be represented as illustrated in Fig. 1, where we can identify two main subjects of research in text mining/classification: the pre-processing steps (document representation and space reduction/feature extraction) and the learning procedure [support vector machine, neural nets, genetic algorithms, etc].

Great relevance has been rightfully given to learning procedures in text categorization [2]. However, before a learning procedure, a number of pre-processing steps are usually taken, leading to a document representation in the input space thus defined. These pre-processing steps interact with each other and their study is relevant in order to assert of their individual influence in space feature reduction and in categorization performance.

Automatic text classification is an important issue of any large scale information retrieval or text mining system and can play a decisive role in information management tasks, such as text retrieval, routing, sorting and filtering. With wide scopes such as the internet, it is often impossible to use human categorization in most tasks, since manual labeling this volume of documents is not tolerable for the user [3]. Nevertheless, there are still trained specialists that assign new items to categories in large taxonomies. As an example, journal papers, like this one, use a set of keywords that authors must provide, from a set of available categories. The applicability and generalization of this procedure to commercial sets is not practicable, since it is costly and time-consuming. Therefore, it would be of huge help if document collections could

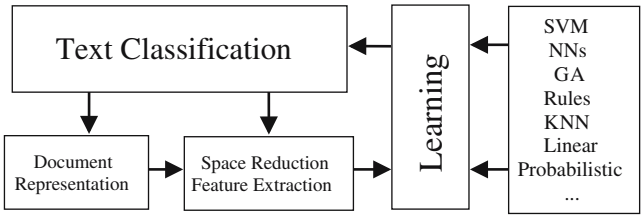

Fig. 1 Text classification

be organised in a way that could automate or semi-automate the task of text categorization [4].

One key difficulty with most text categorization algorithms is that their performance is greatly conditioned by the labeled training set available. Thus, methods that need small sets of labeled examples are currently being explored. Labeling data is expensive but, in most text categorization tasks, unlabeled data are often inexpensive, abundant and readily available. Hence, to achieve this purpose, i.e., the use of relatively small training sets, the information that can be extracted from the testing set, or even unlabeled examples can potentially be used as a way to improve classification performance. This combination of labeled and unlabeled data is usually referred as partially labeled data, where a few labeled learning examples describing the purposes are provided and the remaining data is used to build a custom classifier [5].

In this paper we consider two ways to use unlabeled data information in the learning strategy: background knowledge and active learning. Background knowledge can be defined as any unlabeled collection of text from any source that is related to the classification task [6]. Active learning designs and analyses learning algorithms that can effectively filter or choose the samples to be labeled by a supervisor. The reason for using active learning is mainly to expedite the learning process and reduce the labeling efforts required by the teacher [7].

Dan [8] groups active learning methods according to the selection strategy: committee-based and certainty-based. The first group determines the active examples combining the outputs of a set of committee members. As in [9], most effort is done in determining the examples in which the members disagree the most as examples to be labeled. The certainty-based methods try to determine the most uncertain examples and point them as active examples to be labeled. The certainty measure depends on the learning method used.

Schohn and Conhn [3] propose a method to actively learn with SVMs, exploring the examples that are orthogonal to the space spanned by the training set, in order to give to the classifier information about dimensions not yet explored.

Tong and Koller [10] introduce a method that considers the examples, which better split the current version of the space into equal parts are most informative for the model.

Seeger [11] and Szummer [5] present reports on learning with unlabeled data, comparing several approaches.

Joachims [12] presents a study on transductive SVMs (TSVMs), originally introduced by Vapnik [13]. TSVMs make use of the testing set and extend inductive SVMs, finding an optimal separating hyperplane not only for the training examples, but also of the testing examples [14].

In this work we propose two approaches to incorporate unlabeled data into a SVM: the integration of background knowledge into the learning task and actively choosing the examples to be introduced.

The background knowledge approach automatically introduces into the training set previously unlabeled examples, now classified by the baseline SVM model, increasing the initially reduced training set. These added examples provide the learning machine more information about the underlying distribution with no extra labeling cost.

The active learning method, despite not being fully automated, since a user must classify the margin-based chosen examples, has the potential to efficiently improve classification performance. These examples help the learning machine regarding doubtful areas of classification.

This paper is organized as follows. Section 2 addresses several text classification issues, setting guidelines for problem formulation. Section 3 presents SVM stressing their applicability to text mining. Section 4 introduces the data set used as benchmark in our experiments and the performance criteria applied. Section 5 presents a study on preprocessing methods, concluding on their relative significance. Section 6 focuses on the issues related to the use of partially labeled data on text classification and proposes two approaches: background knowledge and active learning using SVMs. Section 7 presents some conclusions and future work.

2 Automatic text classification

The task of text categorization is the classification of natural text or hypertext (documents for simplicity) into a fixed number of categories based on their content. Each document can be assigned to none, one or several categories, making it a multi-class and multi-label problem.

Most machine-learning systems are designed to handle multi-class data, but much less common are systems that can handle multi-label data. While numerous categorization algorithms can be adapted to multi-label categorization problems, when machine-learning and other approaches are applied to text categorization, a common technique is to decompose the multi-class, multi-label problem into multiple, independent binary classification problems (one per category).

Using this decomposition, there are two possible approaches: a category-pivoted approach or a document-pivoted approach. On one hand, using a category-pivoted approach, for each category a document either belongs or does not belong to it. On the other hand, when a document-pivoted approach is followed, each document belongs or not to each one of the available categories [15]. The approach to follow depends on the kind of application one is running. If the set of categories is a priori fixed and the objective is to categorize new documents then a category-pivoted categorization should be used. If you are in a category discovering phase, a document-pivoted categorization should be

followed. Our attention will be pointed to document-pivoted approaches, since we will consider the categories are a priori defined.

There has been given some relevance to the learning procedure, since it constitutes the key element of text classification. However, the initial steps of document representation and space reduction/feature extraction are also relevant to the success of the learning procedure and consequently to categorization performance.

2.1 Document representation

The most common, simple and successful document representation used so far is the vector space model, also known as the *Bag of Words*. Each document is indexed with the *bag* of the terms occurring in it, i.e., a vector with one component for each term occurring in the whole collection, having as value the number of times the term occurred in the document. Each document is thus represented as a point in a vector space with one dimension for every term in the vocabulary. When a word does not appear in a given document that particular vector dimension is set to zero.

This simple representation implies that the word order in the document is not deemed relevant, what has been proven right by several studies [16], thus limiting the search for novel representations, which would necessarily be more complex.

However, the *Bag of Words* is not usually used in its simplest form so far presented[1]. Instead of using the *term frequency* or $tf(t)$, i.e., the number of times a term t occurs in a document, the terms can be weighted according to their discriminative power within the document collection. This is usually done based on the idea that terms occurring in fewer documents are better selectors. The *document frequency* $df(t)$ of a term t is the number of documents in the collection in which the term occurs. The *inverse document frequency* or $idf(t)$ is (1):

$$idf(t) = \frac{|D|}{df(t)}, \tag{1}$$

where $|D|$ is the number of documents in the collection. The $idf(t)$ of a term t is lower the more documents it appears in [17].

Vector components are weighted according to the $idf(t)$ of the corresponding term. Usually some monotonous function of the $idf(t)$, such as the logarithm or the square root, is used instead of the $idf(t)$ itself, to avoid giving more importance than appropriate to multiple occurrence of terms [15].

The most common, although not the simpler, weighting scheme used is the *tfidf* representation (2):

$$tfidf(t) = tf(t) \times \log(idf(t)) = tf(t) \times \log\left(\frac{|D|}{df(t)}\right). \tag{2}$$

Using the *Bag of Words* approach, and fairly all document representation schemes, a high dimensional space is obtained, where the number of features (terms or words) is much larger than the number of documents available for training. This fact is inevitable, but can be mitigated by the use of some dimension reduction techniques, referred in the next Section and tested in Sect. 5.

2.2 Feature space reduction

Feature space reduction is often performed to improve the performance of the learning algorithm and control the computational time involved.

These techniques, as the name reveals, reduce the size of the document representation. There are a number of issues to be considered, namely, the time spent in reduction, the possible information lost, the learning time reduction and the classification improvement. A reduction technique should only be applied if these issues evaluation yields positive.

In the simplest approach, a term is any space-separated word in a document[2]. However, there are a great number of non-informative words, such as articles, prepositions and conjunctions, called *stopwords*. For this reason, a *stopword* list is usually built with words that should be filtered in the document representation process. Words that are to be included in the *stopword* list are language and task dependent.

Besides the obvious reduction in the number of features, there can potentially be another advantage associated with stopword removal since some stopwords can mislead the learning machine in defining non-existent correlations between documents.

Another technique employed is to consider only words with a document frequency greater than some pre-defined threshold. The fewer documents a word appears in, the more relevant it is. Nevertheless there is a limit to this rule. If the word appears in less than, e.g., three documents, one can consider that there will be a low probability of being a good classifier element.

Another commonly used method is *stemming*, where the word stem is derived from the occurrence of a word by removing case and inflection information. For example "viewer", "view", and "preview" are all mapped to the same stem "view". Stemming does not alter significantly the information included in document representation, but it does avoid feature expansion.

Yet another dimensionality reduction technique, based on feature construction, is latent semantic indexing, whose aim is to handle problems of synonymous[3] and polysemous[4] words. This technique compresses vectors representing documents into other vectors of a lower-dimensional space, whose dimensions are obtained as combination of the original dimensions by looking at their patterns of co-occurrence [18,19].

Section 5 presents an experimental setup that evaluates dimension reduction techniques.

[1] There is an even simpler representation, where the only information is if the term is present or not in the document.

[2] This is a valid definition for English text, not for German or Finnish, for instance, where there are composite nouns.

[3] Words that mean the same, but are written differently.

[4] Words that are written the same way by have different contextual meanings.

2.3 Learning procedure and classification

Having a suitable document representation, adequately processed, the goal is to find a mapping from the representation of documents into the class of possible labels. A classifier is a function that maps an input attribute vector $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ to the confidence that the input belongs to a class – that is $f(\mathbf{x}) = confidence(class)$. In the case of text categorization, attributes are words in the document and classes correspond to text categories.

The problem of inductive construction of a text classifier has been tackled in a variety of different ways. Kwok [20] presents a synopsis of the representative machine-learning approaches for text categorization.

Although these approaches have been successfully applied in numerous problems, they rely heavily on dimension reduction as a pre-processing step, which can rise up design classifier constraints while being computational expensive. Therefore, new methodologies for categorization of text such as kernel-based learning techniques, which circumvent those difficulties, have been recently developed and are one of the current focuses in machine-learning research.

3 Support vector machines

SVM are a learning method introduced by Vapnik [13] based on his statistical learning theory and Structural Minimization Principle. When using SVMs for classification, the basic idea is to find the optimal separating hyperplane between the positive and negative examples. The optimal separating hyperplane is defined as the one giving the maximum margin between the training examples that are closest to the hyperplane. The group of examples (vectors) that lie closest to the separating hyperplane are referred to as support vectors. Once this hyperplane is found, new examples can be classified simply by checking which side of the hyperplane they fall on. Figure 2 shows a simple two-dimensional example, the optimal separating hyperplane, four support vectors, and the minimal margin (ρ), defined by the support vectors.

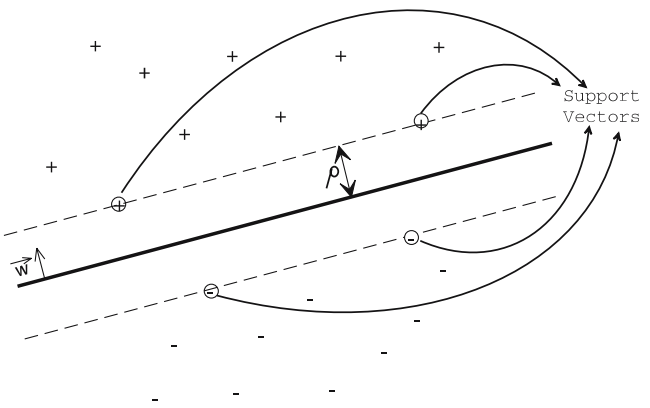

Fig. 2 Optimal Separating Hyperplane

3.1 Foundations

The formulation of SVMs is constructed starting from a simple linear maximum margin classifier. A general two-class problem is posed as follows. Given an i.i.d. sample $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_l, y_l)$, where $\mathbf{x}_i$ for $i = 1, \ldots, l$ is a feature vector of length l and $y_i = \{+1, -1\}$ is the class label for $\mathbf{x_i}$, find a classifier with the decision function $f(x)$, such that $y = f(x)$, where y is the class label for x.

The performance of the classifier is measured in terms of classification error, as defined in Eq. (3).

$$E(y, f(x)) = \begin{cases} 0 & \text{if } \; y = f(x), \\ 1 & \text{otherwise.} \end{cases} \tag{3}$$

Usually, learning machines, SVMs included, have a set of adjustable parameters, λ. Given the above classification task, the machine will tune its parameters λ to learn the mapping $\mathbf{x} \rightarrow y$. This will result in a possible mapping $\mathbf{x} \rightarrow f(\mathbf{x}, \lambda)$, which defines the particular learning machine. The performance of this machine can be measured by the expectation of the test error, as shown in Eq. (4).

$$R(\lambda) = \int E(y, f(\mathbf{x}, \lambda)) \;\; \mathrm{d}P(\mathbf{x}, y). \tag{4}$$

This is called the expected risk, or actual risk and requires that at least an estimate of $P(\mathbf{x}, y)$, which is not available for most classification tasks. Hence, one must settle for the empirical risk measure, defined in Eq. (5).

$$R_{emp}(\lambda) = \frac{1}{l} \sum_{i=1}^{l} E(y, f(\mathbf{x}, \lambda)). \tag{5}$$

This is just a measure of the mean error over the available training data. Most training algorithms for learning machines implement empirical risk minimization (ERM), i.e., minimize the empirical error using maximum likelihood estimation for parameters λ. These conventional training algorithms do not consider the capacity of the learning machine and this can result in over fitting, i.e., using a learning machine with too much capacity for a particular problem.

In contrast with ERM, the goal of structural risk minimization (SRM) is to find the learning machine that yields a good trade-off between low empirical risk and small capacity [13].

There are two major problems in achieving this goal:

(1) SRM requires a measure of the capacity of a particular learning machine or, at least, an upper bound on this measure;
(2) an algorithm to select the desired learning machine according to SRM's goal is needed.

To address these two problems Vapnik and Chervonenkis [13] proposed the concepts of *Vapnik Chervonenkis (*VC*) confidence* and SVMs.

Putting no restrictions on f, it is possible to choose a function that classifies well training data, but does not generalize well on test or real data, therefore the real *Risk* (*see* Eq. 4) will not be minimized.

In VC *theory,* Vapnik and Chervonenkis prove that it is necessary to restrict the class of functions that f is chosen from to one with the *capacity* suitable for the amount of training data. VC theory provides bounds on the test error, circumventing the generalization problems presented earlier. Minimizing these bounds leads to the principle of *SRM*. A function's capacity can take the form of VC *dimension*, defined as the largest number h of points that can be separated in all possible ways, using functions of the given class. If $h < l$ is the VC dimension of the class of functions that the learning machine can implement, then for all the functions of that class, with a probability of at least $1 - \eta$, the bound (6):

$$R(\lambda) \leq R_{emp}(\lambda) + \phi\left(\frac{h}{l}, \frac{\log(\eta)}{l}\right), \tag{6}$$

holds, where the *confidence term* ϕ is defined as (7):

$$\phi\left(\frac{h}{l}, \frac{\log(\eta)}{l}\right) = \sqrt{\frac{h(\log\frac{2l}{h} + 1) - \log(\frac{\eta}{4})}{l}}. \tag{7}$$

3.2 Support vector classification

As already refered, text categorization is a multi-class, multi-label problem and can be broken into a number of binary class problems without loss of generality. This means that instead of classifying each document into all available categories, for each pair $\{document, category\}$ we have a two class problem: the document either belongs or does not belong to the category.

Although there are several linear classifiers that can separate both classes, only one, the optimal separating hyperplane, maximizes the margin ρ, i.e., the distance to the nearest data point of each class, thus presenting better generalization potential.

The output of a linear SVM is $u = \mathbf{w} \times \mathbf{x} - b$, where $\mathbf{w}$ is the normal vector to the hyperplane (see Fig. 2) and $\mathbf{x}$ is the input vector. The margin can then be defined as [21]:

$$\rho = \frac{2}{\|w\|} \tag{8}$$

Maximizing the margin can be seen as an optimization problem:

$$\begin{aligned} &\text{minimize} \quad \frac{1}{2}||\mathbf{w}||^2, \\ &\text{subjected to} \quad y_i(\mathbf{w}.\mathbf{x} + b) \geq 1, \quad \forall i, \end{aligned} \tag{9}$$

where $\mathbf{x}_i$ is the training example and y_i is the correct output for the ith training example, as represented in Fig. 2.

Intuitively the classifier with the largest margin will give low expected risk, and hence better generalization.

To deal with the constrained optimization problem in (9) it is appropriate to introduce Lagrange multipliers, $\alpha_i \geq 0$, and the Lagrangian (10):

$$L_p \equiv \frac{1}{2}||\mathbf{w}||^2 - \sum_{i=1}^{l} \alpha_i (y_i(\mathbf{w}.\mathbf{x} + b) - 1). \tag{10}$$

The Lagrangian has to be minimized with respect to the primal variables $\mathbf{w}$ and b and maximized with respect to the dual variables α_i, i.e., a saddle point has to be found [22].

SVM are universal learners. In their basic form, shown so far, SVMs learn linear threshold functions. However, using an appropriate kernel function, they can be used to learn polynomial classifiers, radial-basis function networks and three layer sigmoid neural networks.

A note-worthy property of SVMs is that their ability to learn is independent of the dimensionality of the feature space. Complexity is based on the margin with which they separate the data, instead of the number of features. This property is very important where automatic text classification is concerned, since usually there are very large number of features.

4 Data set and performance criteria

4.1 Reuters-21578

Reuters-21578 is a financial corpus with news articles averaging 200 words each. Reuters-21578 is publicly available at http://kdd.ics.uci.edu/databases/ reuters21578/ reuters21578 .html. In this corpus there are about 12000 classified stories into 118 possible categories.

Reuters is a very heterogeneous corpus, since the number of stories assigned to each category is very variable. There are stories not assigned to any of the categories and stories assigned to more than ten categories. On the other hand the number of documents assigned to each category is also not constant. There are categories with only one assigned document and others with thousands of assigned documents.

The ModApte split was used, using 75% of the articles (9603 items) for training and 25% (3299 items) for testing. Table 1 presents the ten most frequent categories and the number of training and testing examples, comprising 75% of the items.

In addition to ModApte split, a Small split was also tested to reproduce a real situation in which a real user would be asked to provide the examples. The testing set was exactly the same for the sake of comparison, but the training set, instead

Table 1 Number of positive training and testing documents for the Reuters most frequent categories

Category	Train	Test
Earn	2715	1044
Acquisitions	1547	680
Money-fx	496	161
Grain	395	138
Crude	358	176
Trade	346	113
Interest	313	121
Ship	186	89
Wheat	194	66
Corn	164	52

of 9603 examples was randomly reduced to 10 positive examples and 10 negative examples.

4.2 Performance criteria

For evaluating the simulation results accuracy, recall and precision were used. Take note that all these measures are computed in the testing set for each category.

Accuracy can be defined as (11).

$$accuracy = \frac{documents\ correctly\ classified}{total\ documents}. \tag{11}$$

When the number of positive examples is reduced, as in text classification, common error or accuracy measures are not appropriate, since they value equally both false positives (negative testing examples classified as positive) and false negatives (positive testing examples classified as negatives). It is possible to define different weights to these errors and obtain useful measures, but the usual performance criteria used are recall and precision.

Recall is the percentage of total documents for the given topic that are correctly classified (12).

$$recall = \frac{categories\ found\ and\ correct}{total\ categories\ correct}. \tag{12}$$

Precision is the percentage of predicted documents for the given topic that are correctly classified (13).

$$precision = \frac{categories\ found\ and\ correct}{total\ categories\ found}. \tag{13}$$

An alternative representation is the use of true positives (TP), false positives (FP) and false negatives (FN) as depicted in equations (14) and (15).

$$recall = \frac{TP}{TP + FN}, \tag{14}$$

$$precision = \frac{TP}{TP + FP}. \tag{15}$$

$F1$ measures were considered to provide a unique value to simplify comparisons. To compute $F1$ measure we have used (16).

$$F1 = \frac{2 * precision * recall}{precision + recall}. \tag{16}$$

ROC graphs will also be presented to offer a visual evaluation of classifiers. ROC graphs are two-dimensional graphs in which TP rate is plotted on the Y axis and FP rate is plotted on the X axis. A ROC graph depicts relative trade-offs between benefits (TP) and costs (FP) discretely by fixing the parameters or continuously by varying some parameters resulting in a line instead of a single point [23]. In this work we will use discrete graphs. True positive rate is the ratio of TP in all positive examples (P) and false positive rate is the ratio of FP in all negative examples (N). The purpose is to develop classifiers with high TP rate and low FP rate, i.e., on the northeast corner of a ROC graph.

$$TP\ \ rate = \frac{TP}{P}. \tag{17}$$

$$FP\ \ rate = \frac{FP}{N}. \tag{18}$$

As the text categorization multi-class problem has been sub-divided in several two-class problems, averaging has to be used to find total *criteria* values. There are two types of averaging: *micro-averaging* and *macro-averaging*. In *micro-averaging*, FP, FN, TP and TN values for each category are added, and the final criteria, like $F1$, are computed just once. In *macro-averaging*, performance measures, like $F1$, are computed separately for each category and the mean of the resulting performance is taken. The results presented in this paper use macro-averaging.

5 Pre-processing methods evaluation

An empirical study for evaluation of the importance of the pre-processing methods on text categorization performance has been accomplished using the Reuters-21578 collection with the ModApte split, as explained in Sect. 4.1. The experiments were carried out using the SVM classifier, detailed in Sect. 2.

To fulfill the objectives delineated, a set of eight combinations of pre-processing methods was defined and is represented in Table 2, presenting three possible differences between the test conditions:

- Low frequency word removal: whether or not words that appeared in less than three documents were removed;
- Stopword removal: to removal or not of words in a stopword list;
- Stemming: whether stemming was applied or not.

Table 3 presents the percentage values for accuracy, precision, recall and $F1$, achieved for the eight different combinations of pre-processing methods defined.

Comparing the results achieved it is clear that, where accuracy and precision are concerned, the results present only slight differences. This leads to the assertion that the pre-processing methods tested do not influence greatly these values.

However, in text categorization tasks recall values are usually more sensible due to the distribution of positive and

Table 2 Eight test conditions defined

Test	Low frequency words	Stopwords	Stemming
A	No	No	No
B	No	No	Yes
C	No	Yes	No
D	No	Yes	Yes
E	Yes	No	No
F	Yes	No	Yes
G	Yes	Yes	No
H	Yes	Yes	Yes

Table 3 Accuracy, precision, recall and $F1$ for Reuters' corpus with different pre-processing methods

Test	Accuracy	Precision	Recall	$F1$
A	96.98	83.96	55.66	65.59
B	97.01	84.23	55.92	66.14
C	96.54	84.06	62.81	71.26
D	97.30	83.95	62.54	71.09
E	97.00	84.93	58.05	67.74
F	97.05	85.05	56.05	66.27
G	97.32	85.05	63.06	71.75
H	97.31	85.91	62.54	71.77

negative examples. Usually the number of examples is large, but the number of positive examples is small (less than 5% in Reuters case). Hence, if a learning machine classifies all documents as not belonging to a category, it will still have a large accuracy, making the false negatives a problem to struggle against.

Thus, there is margin for improvement in the recall values, which are normally associated with the number of false negatives, as can be verified in Eq. (14).

Table 4 shows the average number of false negatives and false positives for each combination of pre-processing methods. Examining these values, as expected, the false positive values do not present great divergence, while false negative exhibit substantial differences. The worst (largest) false negatives test values are those where stopword removal was not carried out, especially tests A, B, but also tests E and F, suggesting that preserving those words can be harmful to recall values.

Comparing G and H, which, respectively, correspond to tests without and with stemming, one can also conclude that stemming is not of major importance for recall values, but it can play an important role in precision matters.

While stopword removal alters significantly the contents of input data, stemming only alters its shape, i.e., the reduction of information is not significant. We can therefore say that stemming is more relevant in terms of efficiency of the learning machine (the data is less redundant).

Table 5 presents true positive and false positive rates and Fig. 3 illustrates the corresponding ROC graph for the eight combinations of pre-processing methods defined. Notice that the axis are not equally ranged, to allow an easier comparison.

Analyzing this figure, it can be verified that C,D,G,H, where stopword removal was performed, are definitely better than the rest.

Table 4 False positive and false negative values for Reuters' corpus on the eight different pre-processing conditions defined

Test	False positives	False negatives
A	22.90	62.90
B	22.40	62.50
C	23.30	55.00
D	22.20	54.60
E	22.60	62.60
F	21.80	62.20
G	23.00	53.30
H	22.00	54.50

Table 5 False positive and true positive rates values for Reuters corpus on the eight test conditions defined

Test	False positive rate(%)	True positive rate(%)
A	1.03	55.83
B	1.02	56.12
C	1.02	62.63
D	1.00	55.92
E	1.01	64.98
F	0.98	56.05
G	0.98	62.53
H	0.98	62.54

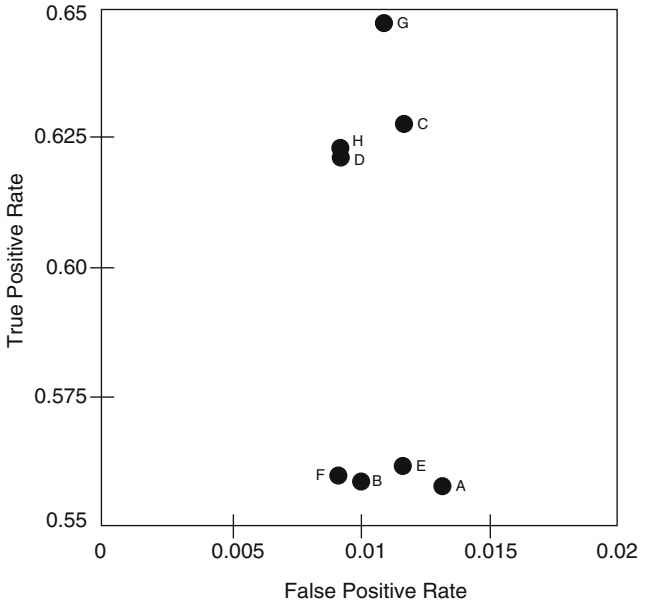

Fig. 3 Discrete ROC graph for the eight testing conditions

The results achieved confirmed that stopword removal removes information that could mislead the learning machine.

6 Partially labeled data

To achieve the best classification performance with a machine learning technique, there has to be enough labeled data. However these data are costly and sometimes difficult to gather. Therefore, using unlabeled data for text classification purposes has recently been actively investigated [24,25].

In general, unlabeled examples are much less expensive and easier to gather than labeled ones. This is particularly true for text classification tasks involving online data sources, such as web pages, email or news stories, where large amounts of text are readily available. Collecting these texts can frequently be done automatically, so it is feasible to collect a large set of unlabeled examples. If unlabeled examples can be integrated into supervised learning, then building text classification systems will be significantly faster, less expensive and more effective.

There is a catch however, because, at first glance, it might seem that nothing is to be gained from unlabeled data, since an unlabeled document does not contain the most important piece of information – its classification.

Consider the following example to give some insight of how unlabeled data can be useful. Suppose we are interested in recognizing web pages about conferences. We are given just a few conferences and non-conferences web pages, along with a large number of pages that are unlabeled. By looking at just the labeled data, we determine that pages containing the word *paper* tend to be about conferences. If we use this fact to estimate the classification of the many unlabeled web pages, we might find that the word *deadline* occurs frequently in the documents that are classified in the positive class. This co-occurrence of the words *paper* and *deadline* over the large set of unlabeled training data can provide useful information to construct a more accurate classifier that considers both *paper* and *deadline* as indicators of positive examples.

6.1 Background knowledge and active learning

In this Section we will propose and compare two approaches that incorporate unlabeled examples in the learning/classification task.

The idea underlying both approaches is that the information contained in the testing set (or in any set of unlabeled data that can be gathered) can be useful to improve the classification performance. Therefore, we propose two ways of integrating those examples, based on the margin with which they are classified. The margin is one of the most important issues in SVMs and basically defines their performance. Thus, our partially labeled approaches, described in the following, can be considered margin-based approaches.

First an inductive SVM classifier (see Sect. 3) is applied to all training examples. Then, both approaches add examples from the unlabeled/testing set to the training set. The difference between approaches is in the way the incorporated examples are chosen and in the number of examples added.

Approach 1: background knowledge

The background knowledge approach incorporates, in the training set, new examples classified by the SVM with more confidence (larger margin, see Fig. 4). The way this is done is the following: examples (only the features, not the classification) from the testing set are directly incorporated into the training set as classified by the baseline inductive SVM, i.e., an example $(\mathbf{x}_i, y_i)$ will be chosen if Eq. (19) holds.

$$(\mathbf{x}_i, y_i) : \rho(\mathbf{x}_i, y_i) = \frac{2}{\|w\|} > \Delta_1. \tag{19}$$

This approach can be considered as the use of background knowledge to improve text classification performance.

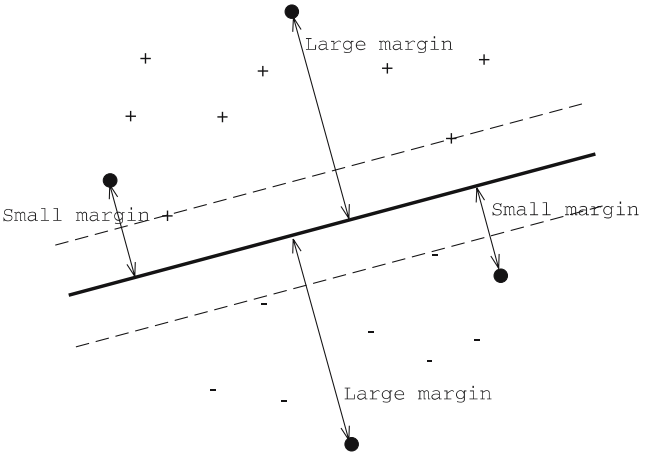

Fig. 4 Unlabeled examples (*black dots*) with small and large margin

Approach 2: active learning

The active learning approach includes a certain number of examples from the testing set (only the features, not the classification) in which the SVM has less confidence (smaller margin, see Fig. 4) after they are correctly classified by the supervisor. Thus, an example $(\mathbf{x}_i, y_i)$ will be included if Eq. (20) holds.

$$(\mathbf{x}_i, y_i) : \rho(\mathbf{x}_i, y_i) = \frac{2}{\|w\|} < \Delta_2. \tag{20}$$

This number of examples can not be large, since the supervisor will be asked to manually classify them. After being correctly classified, they are integrated in the training set. This approach can be regarded as a form of active learning, where the information that an example can introduce in the classification task is considered inversely proportional to its classification margin.

Both approaches have advantages and disadvantages, and we expect to conjugate them to use the advantages and mitigate the disadvantages.

For a straight comparison, we can use the following criteria:

User interaction while the first approach is automated, the second approach needs some user interaction, since the selected items must be classified by the supervisor;

Correctness of training set the first approach does not guarantee its correctness, since the added examples are classified by the inductive SVM, whereas in the second approach all examples in the training set are (correctly) classified by the supervisor;

Computational time there is not a significant difference in the computational time used, however the first approach can take longer, because the examples are automatically classified and there is no limit on the number of examples added;

Performance measured as detailed in Section 4.2: the second approach has greater potential, since the information added is more reliable, but has limitations on the number of items the supervisor can tolerate/is able to classify.

6.2 Discussion of results

This section presents and discusses the results achieved with the baseline SVM inductive classifier, with the background knowledge approach and with the active learning approach.

Baseline results

As refered, the baseline of comparison will be the results obtained with the SVM in the inductive setting, as described in Sect. 3, and are presented in Tables 6 and 7 for ModApte split and Small split, respectively.

Figure 5 black circles present the ROC graph for these two baseline classifiers, whose values are also represented on Table 8.

Approach 1 results: background knowledge

Our background knowledge approach has a parameter Δ_1 (see Equation 19) to be defined. Empirically we considered $\Delta_1 = 0.6$, representing 60% of confidence. Tables 9 and 10 present the results obtained for the first approach with both training/testing splits.

Table 11 presents the false positive and true positive rates used to define the ROC graph with background knowledge approach on Fig. 5 (black squares).

Analysing $F1$ values, there is an improvement of 1% (from 71.77% to 72.50%) where the ModApte split is concerned (Tables 6 and 9), but not with the Small split (Tables 7 and 10), where there is a decrease (from 32.56% to 20.82%).

For this approach to be successful the baseline classifier can not be too weak, since it will be responsible for classifying the unlabeled examples. That is not the case with

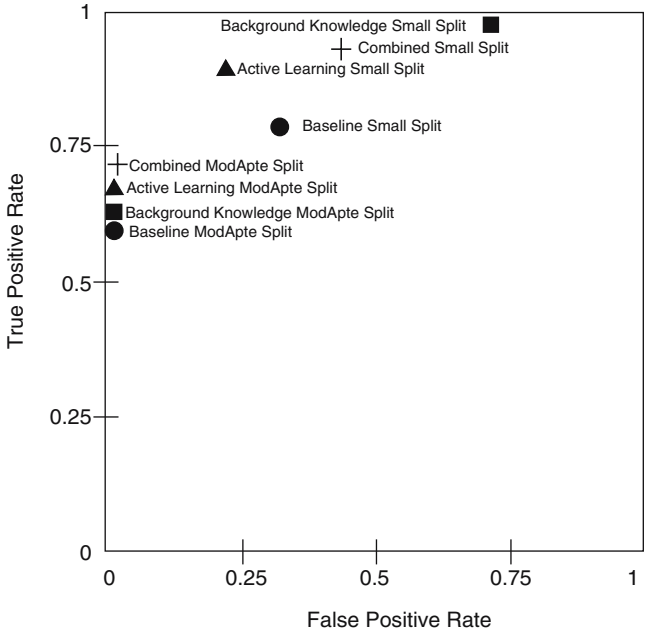

Fig. 5 Discrete ROC graph for the baseline SVM inductive classifiers (*black circles*), background knowledge (*black squares*), active learning (*black triangles*) and combined approaches (*plus signs*)

Table 8 False positive and true positive rates values for baseline inductive classifiers

Test	False positive rate(%)	True positive rate(%)
ModApte split	0.98	62.55
Small split	32.69	80.51

Table 6 Number of support vectors (SV), accuracy (Acc), precision (Prec), recall (Rec) and $F1$ with baseline SVM for ModApte split (9603/3299)

Category	SV	Acc	Prec	Rec	$F1$
Earn	1632	95.92	93.53	95.50	94.50
Acquisitions	1751	94.93	93.09	85.15	88.94
Money-fx	908	96.13	71.43	52.80	60.72
Grain	771	97.96	92.55	63.04	75.00
Crude	693	97.04	84.85	63.64	72.73
Trade	647	97.64	79.49	54.87	64.92
Interest	742	97.15	77.03	47.11	58.46
Ship	500	98.45	89.36	51.85	65.62
Wheat	487	98.77	84.44	57.58	68.47
Corn	484	99.08	93.33	53.85	68.29
Average	861.50	97.31	85.91	62.24	71.77

Table 7 Number of support vectors (SV), accuracy (Acc), precision (Prec), recall (Rec) and $F1$ with baseline SVM for Small split (20/3299)

Category	SV	Acc	Prec	Rec	$F1$
Earn	19	90.32	90.26	82.57	86.24
Acquisitions	19	49.77	32.07	98.24	48.35
Money-fx	18	38.33	8.11	95.65	14.95
Grain	20	81.31	16.06	67.39	25.94
Crude	18	70.50	15.52	84.66	26.23
Trade	18	79.41	15.50	93.81	26.60
Interest	18	54.10	8.02	93.39	14.77
Ship	19	32.31	3.90	96.30	7.50
Wheat	19	95.49	29.61	68.18	41.29
Corn	20	98.20	52.00	25.00	33.77
Average	18.80	68.97	27.11	80.52	32.56

Table 9 Background knowledge – number of support vectors (SV), accuracy (Acc), precision (Prec), recall (Rec) and $F1$ for ModApte split (9603/3299)

Category	SV	Acc	Prec	Rec	$F1$
Earn	1651	95.85	93.27	95.59	94.42
Acquisitions	1800	95.04	92.71	86.03	89.25
Money-fx	928	96.13	71.07	53.42	60.99
Grain	802	98.03	92.71	64.49	76.07
Crude	697	97.18	85.29	65.91	74.36
Trade	661	97.68	79.75	55.75	65.62
Interest	744	97.22	76.92	49.59	60.30
Ship	505	98.49	89.58	53.09	66.67
Wheat	490	98.77	82.98	59.09	69.03
Corn	505	99.08	93.33	53.85	68.29
Average	878.30	97.35	85.76	63.68	72.50

Table 10 Background knowledge – Number of support vectors (SV), accuracy (Acc), precision (Prec), recall (Rec) and $F1$ for Small split (20/3299)

Category	SV	Acc	Prec	Rec	F1
Earn	42	90.14	82.43	93.01	87.40
Acquisitions	92	40.65	28.63	99.12	44.43
Money-fx	44	25.31	6.83	96.27	12.76
Grain	23	37.35	6.74	92.75	12.57
Crude	31	37.63	8.87	97.73	16.26
Trade	36	21.89	4.81	99.12	9.17
Interest	33	22.74	5.11	97.52	9.71
Ship	86	30.83	3.82	90.30	7.35
Wheat	24	5.03	2.39	100.00	4.67
Corn	23	9.43	1.98	100.00	3.88
Average	43.50	32.10	15.16	97.18	20.82

Table 11 False positive and true positive rates values for background knowledge approach

Test	False positive rate(%)	True positive rate(%)
ModApte split	1.02	63.68
Small split	72.25	97.19

Small split. With only 20 examples the initial classifier is not accurate enough to determine new training examples.

Approach 2 results: active learning

The threshold Δ_2 on equation 20 was empirically defined as 0.5. Tables 12 and 13 present the results obtained for the second approach with both training/testing splits.

Table 14 presents the false positive and true positive rates used to define the ROC graph with active learning approach on Fig. 5 (black triangles).

The improvement is more relevant (improvement of 40%, from 32.56 to 44.61%) on the Small split (Tables 7 and 13) than on the ModApte split, a predictable outcome, since the training set was substantially increased (20 initial examples plus 40 examples actively chosen to be classified by the supervisor).

Table 12 Active learning – Number of support vectors (SV), accuracy (Acc), precision (Prec), recall (Rec) and $F1$ for ModApte split (9603/3299)

Category	SV	Acc	Prec	Rec	$F1$
Earn	1662	96.27	94.00	95.98	94.98
Acquisitions	1788	95.28	94.03	85.74	89.69
Money-fx	947	96.59	76.67	57.14	65.48
Grain	791	98.28	94.95	68.12	79.33
Crude	719	97.43	88.72	67.05	76.38
Trade	676	98.20	83.70	68.14	75.12
Interest	777	97.64	84.62	54.55	66.34
Ship	545	98.84	92.86	64.20	75.92
Wheat	482	99.19	90.57	72.73	80.68
Corn	537	99.44	97.37	71.15	82.22
Average	892.40	97.72	89.75	70.48	78.61

Table 13 Active learning – number of support vectors (SV), accuracy (Acc), precision (Prec), recall (Rec) and $F1$ for Small split (20/3299)

Category	SV	Acc	Prec	Rec	$F1$
Earn	54	92.64	86.78	94.35	90.41
Acquisitions	56	57.76	35.92	97.50	52.50
Money-fx	56	93.95	48.15	88.82	62.45
Grain	55	66.42	12.01	93.48	21.29
Crude	54	57.48	11.83	90.91	20.94
Trade	55	95.04	43.81	87.61	58.41
Interest	53	75.99	13.71	87.60	23.71
Ship	47	70.47	8.62	97.53	15.84
Wheat	52	95.39	32.43	90.91	47.81
Corn	56	97.92	45.21	63.46	52.80
Average	53.80	80.31	33.85	89.22	44.61

Table 14 False positive and true positive rates values for active learning approach

Test	False positive rate(%)	True positive rate(%)
ModApte split	0.84	70.49
Small split	21.40	89.21

In what ModApte split (Tables 6 and 12) is concerned this active approach improves 10% the baseline results (from 71,77 to 78,61%).

6.3 Combining both approaches

Tables 15 and 16 present the results of combining active learning and background knowledge in both training splits, i.e., both traning sets were enriched with both items classified by the baseline SVM (background knowledge) and by the user (active learning).

Table 17 presents the false positive and true positive rates used to define the ROC graph with combined background knowledge and active learning approach on Fig. 5 (plus signs).

The combined approach has better performance with ModApte split. It surpasses the baseline results (71.77%) circa 11% reaching 79.66%.

The combined approach with Small split presents poorer results than the single active learning approach, 34.29% com-

Table 15 Active learning and background knowledge combination – number of support vectors (SV), accuracy (Acc), precision (Prec), recall (Rec) and F1 for ModApte split (9603/3299)

Category	SV	Acc	Prec	Rec	$F1$
Earn	1682	96.30	94.00	96.07	95.02
Acquisitions	1836	95.46	93.52	87.06	90.17
Money-fx	961	96.69	77.24	59.01	66.91
Grain	806	98.28	94.95	68.12	79.33
Crude	721	97.71	89.36	71.59	79.49
Trade	690	98.20	83.70	68.14	75.12
Interest	780	97.61	83.54	54.55	66.00
Ship	547	98.98	91.94	70.37	79.72
Wheat	491	99.26	90.91	75.76	82.65
Corn	537	99.44	97.37	71.15	82.22
Average	905.10	97.79	89.65	72.18	79.66

Table 16 Active learning and background knowledge combination – Number of support vectors (SV), accuracy (Acc), precision (Prec), recall (Rec) and F1 for Small split (20/3299)

Category	SV	Acc	Prec	Rec	$F1$
Earn	61	92.50	87.13	93.39	90.15
Acquisitions	105	44.84	30.10	98.68	46.13
Money-fx	78	34.25	7.73	96.89	14.32
Grain	57	54.17	9.07	93.48	16.54
Crude	67	51.14	10.55	92.05	18.93
Trade	70	90.50	28.25	90.27	43.03
Interest	65	47.45	7.07	93.39	23.71
Ship	130	34.60	4.08	97.53	7.83
Wheat	56	93.59	25.42	90.91	39.73
Corn	59	98.13	49.18	57.69	53.46
Average	74.80	64.12	25.86	90.43	34.29

Table 17 False positive and true positive rates values for combined background knowledge and active learning approach

Test	False positive rate(%)	True positive rate(%)
ModApte split	0.87	72.18
Small split	38.78	90.42

pared with 44.61%, confirming that the baseline classifier is to weak to be used as background knowledge incorporator.

6.4 Comparison with transductive SVM

Besides the comparison with the baseline SVMs, we present now a comparison with Transductive SVM (TSVM), first introduced by Vapnik[13]. Unlike the inductive baseline setting, in transductive setting the location of testing examples is studied to define the structure of the classsfier, maximizing the margin that separates all examples, training and testing examples.

Previous work already tested this approach on the same testing set also with ModApte split [14] and Table 18 presents the comparison with Background Knowledge, Active learning and their combination, presented in this work.

Analysing Table 18 it can be concluded that the proposed approaches present an improvement over TSVM, and thus constitutes a valid learning approach.

Table 18 Comparison between TSVM, background knowledge (BK), active learning (AL) and the combination of BK and AL (Combined) for Reuters' ModApte split

Category	TSVM	BK	AL	Combined
Earn	94.46	94.42	94.98	95.02
Acquisitions	89.38	89.69	98.24	90.17
Money-fx	70.22	60.99	65.48	66.91
Grain	78.49	76.07	79.33	79.33
Crude	75.63	74.36	76.38	79.49
Trade	69.88	65.62	75.12	75.12
Interest	76.28	60.30	66.34	66.00
Ship	79.49	66.67	75.92	79.72
Wheat	78.96	69.03	80.68	82.65
Corn	67.24	68.29	82.22	82.22
Average	78.00	78.61	80.52	79.66

7 Conclusions and future work

Support vector machines present good results in automatic text categorization, since their objective is to maximize the margin between positive and negative examples for each class.

Several items that influence the categorization task were examined, to assert their importance in categorization performance. It was concluded that the most significant of them was the stopword removal, whose influence was determinant and worthy of more research. The ROC graph presented in Fig 3 undoubtedly shows this assertion.

This paper presented two approaches to introduce unlabeled documents information into the learning procedure. The results presented in the Sect. 6.2 are encouraging to the improvement achieved by both methods.

The background knowledge method has the advantage of being completely automated. However, it should not be used with small training sets, i.e., with too weak initial classifiers. When this is not the case it can introduce an improvement.

The proposed margin-based active learning method has potential to substantially improve performance when small training sets are available. This conclusion is very important in text mining tasks, since usually there are a small number of classified examples and a huge number of unlabeled ones. The ROC graph presented in Fig. 5 shows these conclusions very clearly.

Research on other active methods to incorporate background knowledge is forseen as future work.

Acknowledgements CISUC – Center of Informatics and Systems of University of Coimbra and Project POSI/SRI/41234/2001 are gratefully acknowledged for partial financing support.

References

1. Hearst MA (1998) Trends and controversies: Support vector machines. IEEE Intelligent Syst, 13(4): 18–28
2. Kwok JT-Y (1998) Support vector mixture for classification and regression problems. In: Proceedings of the 14th International Conference on Pattern Recognition, vol. 1.1998 IEEE Computer Society, pp 255–258
3. Schohn G, Conhn D (2000) Less is more: active learning with support vector machines. In Proceedings of the 17th international conference on machine learning, pp 839–846, Morgan Kaufmann, San Francisco
4. Dumais S, Platt J, Heckerman D, Sahami M (1998) Inductive learning algorithms and representations for text categorization. In: Proceedings of the 7th international conference on information and knowledge management, ACM Press, pp 148–155
5. Szummer M (2002) Learning from partially labeled data. PhD thesis, Massachusetts Institute of Technology
6. Zelikovitz S, Hirsh H (2001) Improving text classification with LSI using background knowledge. In: Proceedings of the 7th international joint conference on artificial intelligence (IJCAI-2001)
7. Baram Y, El-Yaniv R, Luz K (2003) Online choice of active learning algorithms. In: Proceedings of ICML-2003, 20th international conference on machine learning, pp 19–26
8. Dan S (2004) Multi-criteria-based active learning for named entity recognition. Master's thesis, National University of Singapore
9. McCallum AK, Nigam K (1998) Employing EM and pool-based active learning for text classification, In: Proceedings of ICML-98,

15th international conference on machine learning. Morgan Kaufmann Publishers, San Francisco, pp 350–358
10. Tong S, Koller D (2001) Support vector machine active learning with applications to text classification. J Machine Learn Res 2: 45–66
11. Seeger M (2001)Learning with labeled and unlabeled data. Technical Report
12. Joachims T (1999) Transductive inference for text classification using support vector machines. In: Proceedings of ICML-99, 16th international conference on machine learning (Bratko I, Dzeroski S (eds), Morgan Kaufmann Publishers, San Francisco pp 200–209
13. Vapnik V (1998) The nature of statistical learning theory. Springer, 1998
14. Silva C, Ribeiro B Berlin Heidelberg Newyork Labeled and unlabeled data in text categorization. In: IEEE international joint conference on neural networks (IJCNN' 2004)
15. Sebastiani F (1999) A tutorial on automated text categorisation. In: Proceedings of ASAI-99, 1st Argentinian symposium on artificial intelligence (Amandi A, Zunino R (eds.), pp 7–35
16. Joachims T (2001) Learning to classify text using support vector machines. Kluwer, Dordrecht
17. Cooley R (1999) Classification of news stories using support vector machines. IJCAI 99 workshop on text mining. Stockholm, Sweden, August 1999
18. Chen C-M, Stoffel N, Post M, Basu C, Bassu D, Behrens C (2001) Telcordia LSI engine: implementation and scalability issues. In: 11th international workshop on research issues in data engineering (RIDE 2001), pp 51–58
19. Cristianini N, Shawe-Taylor J, Lodhi H (2001) Latent semantic kernels. In: Proceedings of ICML-01, 18th international conference on machine learning. Morgan Kaufmann Publishers, San Francisco, pp 66–73
20. Kwok JT (1998) Automated text categorization using support vector machine. In: Proceedings of ICONIP'98, 5th international conference on neural information processing, pp 347–351
21. Gunn S (1998) Support vector machines for classification and regression. Technical. report, Faculty of engineering and applied science. Department of Electronics and Computer Science
22. Schölkopf B, Burges C, Smola A (1999) Advances in Kernel methods. MIT Press, Cambridge, pp 1–15
23. Fawcett T (2004) Roc graphs: notes and practical considerations for data mining researchers. Technical Report HPL-2003–4, HP Laboratories, http://www.hpl.hp.com/personal/Tom_Fawcett/papers/
24. Hong J, Cho S-B Incremental support vector machine for unlabeled data classification. In: Proceedings of the 9th International conference on neural information Processing (ICONIP), pp 1403–1407 (2003)
25. Liu B, Dai Y, Li X, Lee WS, Yu PS (2003) Building text classifiers using positive and unlabeled examples. In: Proceedings of the Third IEEE International Conference on Data Mining. IEEE Computer Society p 179