TECHNISCHE UNIVERSITÄT
CHEMNITZ

Patent Document Classification Based on Mutual Information Feature Selection

I. Costantea, R. I. Boţ, G. Wanka

Preprint 2004-11

Fakultät für Mathematik

Patent Document Classification Based on Mutual Information Feature Selection

Ioana Costantea[*] Radu Ioan Boţ[†] Gert Wanka [‡]

Abstract. We describe a supervised text classification approach based on a greedy feature selection method, which uses a support vector machine (SVM) classifier. As feature selection method we use the mutual information. This measures the quantity of information about the categories contained by the words. To train and test the algorithm we used patent documents from the US Patent Classification System. Average break-even point (BEP) for some US Classes is reported as conclusion.

Key Words. Supervised Classification, Support Vector Machines, Mutual Information, Patent Classification

1 Introduction

Because of the increasing volume of patent (or, more generally, text) documents in the last years, there is a growing interest in developing methods that generate good results in the management (i.e. classify and retrieve) of some huge amount of information. Text categorization techniques are used, for example, to classify new documents or to find interesting information. Since the use of human trained professionals by the construction of adequate classifiers is a time-consuming and costly process, it is advantageous to build artificial classifiers that learn from examples. Some advantages of these so-called inductive learning techniques are that they are easy to construct, to up-date and on the other hand that they depend only on information which is easy to be provided (e.g. items that are in or out the class). The standard approach to text categorization consists in the use of a document representation in a word-based input space, i.e. as a vector in some high-dimensional Euclidian space, where each dimension corresponds to a word and then in the training of a

[*] Faculty of Mathematics, Chemnitz University of Technology, D-09107 Chemnitz, Germany, e-mail: ioana.costantea@mathematik.tu-chemnitz.de

[†] Faculty of Mathematics, Chemnitz University of Technology, D-09107 Chemnitz, Germany, e-mail: radu.bot@mathematik.tu-chemnitz.de

[‡] Faculty of Mathematics, Chemnitz University of Technology, D-09107 Chemnitz, Germany, e-mail: gert.wanka@mathematik.tu-chemnitz.de

classification algorithm in a supervised learning manner. Since the early days of the text classification ([9]), the theory and the practice advanced considerably and some strong learning methods have been produced ([8], [12]). Although there are a lot of sophisticated document representation techniques ([1], [10], [11]), the simple and natural word-based representation, known as bag-of-words (BOW), remains very popular. Indeed, the best known multi-class classification results for the well-known Reuters-21578 data set (available at [17]) have been obtained by using the BOW representation ([2], [5], [7]).

In this paper we describe some results obtained from experiments using some classes of patent documents from the US-Patent Classification System ([18]). We use a BOW representation in a (high dimensional) Euclidian vector space. Only binary feature values are used. In a preprocessing phase, the digits, the non-letter characters and the one-character-words were eliminated. We performed a stopwords elimination using the SMART-stopwords list ([14]) and also applied the Porter stemmer ([15]), in order to reduce the dimension of the working dictionary. For further dimensionality reduction, we used the greedy mutual information feature selection [13]. For learning and classifying we used a Support Vector Machine classifier (SVM^{lights}, [16]). The goal of our work is to analyze the classification power of the SVM algorithm combined with mutual information feature selection, applied to the special case of patent documents, a combination of methods which, to the best of our knowledge, has not been tested until now on patent documents.

The paper is structured as follows. The next section contains a description of the mutual information feature selection method (section 2.1) and of the Support Vector Machines classifier (section 2.2), finalized with the description of the combination of both methods in a supervised learning algorithm (section 2.3). In the third section we make a description of the data used for training and tests, including the preprocessing steps (section 3.1). We also introduce some indices to quantify the performance of the classification (section 3.2). Finally we present some concrete results of our tests (section 3.3) and put together some conclusions of our work and some further remarks.

2 Methods and tools

In this section we present the mutual information feature selection method, the Support Vector Machines (SVM) algorithm and finally a supervised learning algorithm based on a combination of the two methods mentioned above.

2.1 Feature selection via mutual information

Mutual information, also known as cross entropy or information gain is a widely used information theoretical measure for the stochastic dependency of two random variables.

Let W be the set of words in our training documents (dictionary) and C the set of categories in which the documents are classified. Let $w \in W$ be a word and $c \in C$ a category (they can be seen as random variables). Then the mutual information between the word w and the category c can be defined as (cf. [13])

$$I(w, c) = \log \frac{P(w \wedge c)}{P(w) \times P(c)}, \tag{1}$$

where $P(w)$ and $P(c)$ are the a-priori probabilities of the word w, and category c, to be selected from W and C, respectively, $P(w \wedge c)$ is the join distribution of the two variables and log is the natural logarithm. One can see that the mutual information has a natural value of 0 if the two variables are (statistically) independent (i.e. $P(w \wedge c) = P(w) \times P(c)$).

If one considers the co-occurrence documents-words matrices (i.e. the 0-1 matrix having a number of lines equal the number of documents and a number of columns equal the number of words in the dictionary, where position (i, j) will be 1 if the document and the word co-occur (i.e. the word occurs in the document) and 0 otherwise) and documents-categories matrices (obtained in an analogous way), by taking into consideration the following notations

cw is the number of times the word and the category co-occur,

$w \setminus c$ is the number of times w occurs without c,

$c \setminus w$ is the number of times c occurs without w and

N is the total number of training documents,

one can estimate the mutual information as (cf. [13])

$$I(w, c) \approx \log \frac{cw \times N}{(cw + (c \setminus w)) \times (cw + (w \setminus c))}. \tag{2}$$

In other words, $I(w, c)$ measures the quantity of information the word w contains about the category c.

If one takes into consideration the big amount of (patent) documents available and their increasing complexity, it is clear that the documents-words co-occurrence matrix can have a very high dimension. This matrix is usually also very sparse. In order to improve the classification quality (via noise reduction) and to reduce the computational complexity, a lot of dimensionality reduction methods are proposed in the literature. Despite the existence of sophisticated methods (e.g. [6]), many authors have considered simple and greedy approaches ([13]). In [5], as in our experiments, the mutual information was used for feature selection. For each category the mutual information contained by each word is computed and for further use only the words corresponding to the largest K values are selected.

A characteristic of mutual information is that its value is strongly influenced by the marginal probability $P(w|c)$ of the words, as it can be seen below

$$\begin{aligned} I(w,c) &= \log \frac{P(w \wedge c)}{P(w) \times P(c)} \\ &= \log P(w \wedge c) - \log P(w) - \log P(c) \\ &= \log P(w|c) - \log P(w). \end{aligned}$$

This means that for words with an equal conditional probability $P(w|c)$ rare words will have higher score than the common ones. But this is not necessary a weakness, as [13] claims, because exactly these "rare words" (i.e. words which do not appear in so many documents) could be very useful when one wants to make more than a rough classification.

2.2 Support Vector Machines

The SVM classifiers have been first introduced in [3] and they are based on the Structural Risk Minimization Principle [12] widely used in the computation learning theory. The basic idea of this principle is to find a classifier for which one can guarantee the lowest true error (the probability that the classifier will misclassify an unseen and randomly selected test example). This is done in [12] by controlling the complexity of the classifiers space (known as VC-dimension).

We consider the problem of learning pattern recognition from examples. Let be the following training data set

$$S = \{(x_1, y_1), \ldots, (x_l, y_l)\} \subseteq \mathbb{R}^n \times \{1, ..., m\}.$$

If $m = 2$ we speak about a binary classification problem. We will treat in what follows only this case, because any m-class classification problem can be reduced to m binary classification problems (each class will be separated from the rest). We will denote in this case the two classes with $+1$ and -1 (the set of positive, respectively negative examples).

For simplicity, let us also assume that the data are linearly separable, i.e. there exists a hyperplane

$$H = \{x \in \mathbb{R}^n | < w, x > +b = 0\}, w \in \mathbb{R}^n, b \in \mathbb{R}$$

which correctly classifies the training data. Here $< w, x >$ denotes $w^T x$, the scalar product between w and x. A clear formulation of the problem is: Find an optimal hyperplane H, such that the margin γ of the training set is maximal (i.e. H is the maximal margin hyperplane).

This formulation of the problem is known as linear hard-margin SVM ([8]). The margin γ is the distance from the hyperplane to the closest training example (see Figure 1).

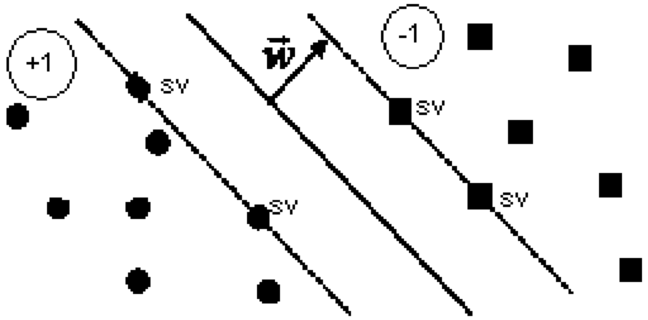

Fig 1. Linear Support Vector Machine

For each separable training set there exists exactly one hyperplane with a maximum margin ([4]). The closest training points to the hyperplane are called support vectors.

Because the problem is linearly separable, there exist $w \in \mathbb{R}^n$ and $b \in \mathbb{R}$ such that $y_i(< w, x_i > +b) > 0, \forall i = 1, \ldots, l$. Rescaling w and b such that the point(s) closest to the hyperplane satisfy $| < w, x_i > +b| = 1$, we obtain a pair (w, b) with

$$y_i(< w, x_i > +b) \geq 1, \forall i = 1, \ldots, l.$$

One can prove the following statement (see [4])

Theorem 1. *Given a linearly separable training set*

$$S = \{(x_1, y_1), \ldots, (x_l, y_l)\},$$

the hyperplane $H = \{x \in \mathbb{R}^n | < w, x > +b = 0\}$*, where* (w, b) *solves the optimization problem*

$$\begin{array}{lll} (P) & \min\limits_{(w,b)} & \frac{1}{2} < w, w > \\ & s.t. & y_i(< w, x_i > +b) \geq 1, \; i = 1, \ldots, l, \end{array}$$

releases the maximal margin hyperplane with $\gamma = \frac{1}{||w||}$.

One can solve this quadratic optimization problem with the tools of duality theory. Using the Lagrange function

$$L(w, b, \alpha) = \frac{1}{2} < w, w > - \sum_{i=1}^{l} \alpha_i [y_i(< w, x_i > +b) - 1],$$

where $\alpha = (\alpha_1, \ldots \alpha_l), \; \alpha_i \geq 0, \; i = 1, \ldots, l$, are the Lagrange multipliers, one can obtain the following dual to the optimization problem (P) (cf. [4])

$$\begin{aligned}(D) \quad \max_{\alpha} \quad & \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{l} y_i y_j \alpha_i \alpha_j < x_i, x_j > \\ \text{s.t.} \quad & \sum_{i=1}^{l} y_i \alpha_i = 0, \\ & \alpha_i \geq 0, \ i = 1, \ldots, l.\end{aligned}$$

The following theorem can be formulated (cf. [4]).

Theorem 2. *Consider a linearly separable training set*

$$S = \{(x_1, y_1), \ldots, (x_l, y_l)\}$$

and suppose the parameters α^* solve the quadratic optimization problem (D). Then the vector $w^* = \sum_{i=1}^{l} y_i \alpha_i^* x_i$ provides the maximal margin hyperplane with $\gamma = \frac{1}{||w^*||}$.

Because b does not appear in the dual problem, its value must be found by making use of the primal constraints. It follows that

$$b^* = -\frac{1}{2} \left(\max_{(y_i=-1)} < w^*, x_i > + \min_{(y_i=1)} < w^*, x_i > \right).$$

Using the Karush-Kuhn-Tucker optimality conditions (see for example [4]), one can see that the optimal solutions (w^*, b^*) and α^* must also satisfy

$$\alpha_i^* [y_i (< w^*, x_i > + b^*) - 1] = 0, \ \forall i = 1, \ldots, l.$$

From here one can see that (cf. [4]) for all support vectors, which are points with $y_i(< w^*, x_i > + b^*) = 1, i = 1, \ldots, l$, the corresponding α_i^* may be non-zero. All the other parameters α_i are 0.

The optimal hyperplane can be expressed in terms of these elements as

$$\begin{aligned} h(x, \alpha^*, b^*) &= \sum_{i=1}^{l} y_i \alpha_i^* < x_i, x > + b^* \\ &= \sum_{i \in SV} y_i \alpha_i^* < x_i, x > + b^*, \end{aligned}$$

where $SV = \{i : \ i \in \{1, \ldots, l\}, x_i is \text{support vector}\}$ Another consequence of the KKT-optimality conditions is that for $j \in SV$ we have

$$y_j h(x_j, \alpha^*, b^*) = y_j (\sum_{i \in SV} y_i \alpha_i^* < x_i, x > + b^*) = 1$$

and therefore

$$\begin{aligned} < w^*, w^* > &= \sum_{i,j=1}^{l} y_i y_j \alpha_i^* \alpha_j^* < x_i, x_j > \\ &= \sum_{j \in SV} \alpha_j^* y_j \sum_{i \in SV} y_i \alpha_i^* < x_i, x_j > \\ &= \sum_{i \in SV} \alpha_i^* (1 - y_i b^*) \\ &= \sum_{i \in SV} \alpha_i^*. \end{aligned}$$

One problem with the simple formulation above is that it fails when the training data are not linearly separable and this is often the case in real world problems. Therefore it might make sense to allow some errors on the training data. In [3] a solution to this problem was suggested. It is called "soft margin" SVM. By introducing the slack variables $\xi_i, i = 1, \ldots, l$, and the cost factor C, one can rewrite the primal problem (P) as follows

$$\begin{array}{lll} (P') & \min\limits_{(w,b,\xi)} & \frac{1}{2} < w, w > + C \sum_{i=1}^{l} \xi_i \\ & \text{s.t.} & y_i(< w, x_i > + b) \geq 1 - \xi_i, \ i = 1, \ldots, l, \\ & & \xi_i > 0, \ i = 1, \ldots, l. \end{array}$$

If a training example lies on the wrong side of the hyperplane, the corresponding ξ_i is greater than 1. The factor C establishes a connection between the training error and the model complexity. A large value of C causes a behaviour similar to that of hard-margin SVM and a small one increases the number of allowed training errors.

Using similar paths, one can find the following dual

$$\begin{array}{lll} (D') & \max\limits_{\alpha} & \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{l} y_i y_j \alpha_i \alpha_j < x_i, x_j > \\ & \text{s.t.} & \sum_{i=1}^{l} y_i \alpha_i = 0, \\ & & 0 \leq \alpha_i \leq C, \ i = 1, \ldots, l. \end{array}$$

The training examples with $\alpha_i > 0$ are also called support vectors, those with $\alpha_i = C$ are called bounded support vectors and for $0 < \alpha_i < C$ we have unbounded support vectors.

A very useful generalization of the SVM is the so-called non-linear (or kernel-based) SVM. The basic idea is that the non-linear separable input data are mapped via a non-linear function (a so-called kernel function) into a higher dimensional feature space, where one uses a linear SVM classifier. Details and examples of kernel functions can be found for instance in [8] and [4].

2.3 The Mutual Information Feature Selection Algorithm

Our goal is to present a supervised classification method, which will be able, after the training phase, to classify new (patent) documents into one or several of the given classes. Based on an idea presented in [2], we developed and experimented a supervised classification program that combines the mutual information feature selection method with the classification power of SVM. In what follows, we shortly present the structure of the algorithm.

First, our program deals with the multi-class classification problem, this means that each document can be classified in one or more classes. In a supervised method, the program has two parts: a training one, in which one learns a classification model based on given labeled documents and a so-called classification (or test) part, in which new documents can be classified using the learned model.

The training part of the algorithm takes as input a bag-of-words (BOW) representation of the documents (i.e. each document is represented as a vector of the contained words, related to a given dictionary) and also a list of classes in which each document is classified (more details about the representation of the input data will come in the next section). The number K of features to be selected is also given.

Even when we work with a small number of documents (about a few hundreds), the size of the corresponding dictionary could be very high. In order to avoid unnecessary long computation time and in the same time to diminish the noise (and so to increase the classification accuracy), we have done a greedy mutual information feature selection, using the method presented in section 2.1. This means that, for each given class and for all the words in the dictionary we computed the mutual information using the relation (2) and then we selected for further computations only the K words for which $I(w, c)$ takes the greatest values. In this way we obtain for each class a much smaller "active" dictionary. We project then the whole set of training documents onto this reduced dictionary. Because of the sparseness of the initial document representation, the loss of information during this process is minimal. We train then an SVM classifier (SVM^{lights}) in this small dimensional vector space in order to obtain a classification model for each class.

MI-SVM based training

Input

$d_i = (B_i, C_i),\ i = 1, \ldots, N$, characterizing the document i, where

B_i is a BOW representation of d_i

```
    $C_i$ is the set of categories in which $d_i$ appears,
    $C_i = \{c_{i1}, \ldots, c_{il}\}$ and $D_{train} = \{d_1, \ldots, d_N\}$
    $K$ is the number of selected features (the same for each category)

Output

    $\{M_1, \ldots M_l\}$ set of binary classifiers
    $\{W_1, \ldots, W_l\}$ set of selected features for each class

Loop

    initialize the dictionary
    for each class $c_i \in C = \{c_1, \ldots, c_l\}$ and each $w \in W$
        $Pos_i := \emptyset$, $Neg_i := \emptyset$ (set of positive and negative examples)
        compute $I(w, c_i)$
        sort the words in $W$ according to $I(w, c_i)$
        extract the $K$ top words $W_i = \{w_1^i, \ldots, w_K^i\}$
        for each $d_j = (B_j, C_j) \in D_{train}$
            represent $d$ on the basis of the set $W_i$
            let $Project(d) := (B_j \cap W_i, C_j)$
            if $c_i \in C_j$
                  add $Project(d_j)$ to $Pos_i$
                 else add $Project(d_j)$ to $Neg_i$
            end if
            end for $d_j$
        train $SVM^{lights}$ to separate $Pos_i$ and $Neg_i$ and obtain $M_i$
    end for $c$
```

One can observe that our l-class classification problem was divided into l binary classification problems, where each class is separated from the others. This is also the way in which SVM^{lights} works.

The classification (or test) part takes as input a BOW representation of one (or more) document(s), project this onto the set of selected features $W_i = \{w_1^i, \ldots, w_K^i\}$, $i = 1, \ldots, l$, and, using the classifiers $M_i, \ldots, M_l$, predicts a class (or more) in which this document could be classified. It could also happen that a document is nowhere classified. Here is the classification part of SVM^{lights} that is used.

MI-SVM based classification

Input

```
        $d = (B_j, C_j)$ a test document
        $M = \{M_1, \ldots, M_l\}$ set of binary classifiers
        $\{W_1, \ldots, W_l\}$ set of selected features for each category
    Output
        $V = (v_1, \ldots, v_l)$ the values of the separating hyperplane on the test document, where
            $v_i > 0$ means $d \in c_i$
            $v_i < 0$ means $d \notin c_i$
            $v_i = 0$ is undecided
    Loop
        for each $M_i \in M$
            determine $Project(d) = (B_j \cap W_i, C_j)$
            run $M_i$ on $Project(d)$ to obtain $v_i$
        end for
```

In the next section we will present details about the training and test data, some results and their interpretation.

3 Applications

The USPCS (United States Patent Classification System) is a well-known patents and trademarks collection. The basis of this library can be found in 1871, when the ”Patent and Trademarks Depository Library Program” was started, with the goal of helping people to get informed about the already existent patents and about the laws which protect the invention (and the inventors). Since 1977 the size of this library has grown at least by a factor of four. Due to this continuously increasing number of patent documents, the process of finding a patent document or ordering (classifying) a new one is getting more difficult. Nowadays, millions of patent documents are hierarchically classified on the basis of their claims in about 450 US-classes and 150000 subclasses (see [18]).

As a consequence of the increasing number of patents, grows also the interest in finding a reasonable automatic classification algorithm. Here, reasonable is used in the sense that it approximates well the present USPCS, offers new and better classification possibilities and is easy to update. With this goal we try to use the supervised MI-SVM classification method (see section 2.3) to learn a classification model (on the basis of some labeled examples) and then to classify new patent documents.

3.1 Data used and preprocessing steps

For our work we used the following US classes (left-US class number, right-class name)

036 Boots, shoes and leggings
175 Boring and penetrating the earth
219 Electric heating
307 Electrical transmission or interconnected systems
318 Motive power systems
323 Power supply or regulation systems
329 Demodulators
330 Amplifiers
331 Oscillators
332 Modulators
338 Electrical resistors
343 Radio wave antenna
361 Electrical systems and devices
372 Coherent light generators
379 Telephonic communications
703 Structural design, modelling, simulation and emulation
704 Speech signal processing, linguistics, language translation etc.
706 Artificial intelligence
707 Data base, file management or data structures.

Details about all the patents in these classes and about the present classification can be found, for example, in [18]. Some preprocessing have been done by IP Century AG (www.ipcentury.de), which also provided our data.

In order to perform our tests, we first did some other preprocessing, in order to eliminate the redundant information. For each US Class we constructed some binary documents - words (DW) and documents - classes (DC) matrices. Let us first describe the construction of the DW matrix.

The DW matrix is a 0-1 matrix, having a number of lines equal to the number of documents and a number of columns equal to the number of words in the corresponding dictionary. The words in the dictionaries are from the claims, abstract and description of each patent. The position (i, j) will be 1 if the document and the word co-occur (i.e. the word occurs in the document) and 0 otherwise. The following transformation have been done to the dictionaries:

1. One-letter words, digits and ”_”, ”-”, ”’” characters were eliminated, for example ”h2o” became ”ho”, ”clark’s” became ”clarks” and so on.

2. Stopwords were eliminated, using the SMART-stoplist ([14]), which contains 571 stopwords, like ”all”, ”and”, ”or”, etc.

3. The Porter stemmer ([15]) was applied on the remaining dictionary and, for instance, instead of ”computer”, ”compute”, ”computing” we kept only ”comput” (the stem of the word). After this preprocessing, for the class 323 for example we reduced the dictionary from 20025 words to 12148.

The DC matrix is also a 0-1 matrix, which contains the documents-classes co-occurrence. Each document could be classified in subclasses of one US main-class and in the same time in subclasses of other US-class. We take into consideration once only the subclasses of one class. As an example, we had the patent US05281906 classified in the subclasses 323/313, 323/314, 323/907 and 327/537 (in our data for the class 323). In this case we took in consideration only the first 3 subclasses. Moreover, one can see that 323/314 is a sub-subclass of 323/313 and in this case we kept only the ”higher” class (parent) (323/313). In an analogous manner, we eliminated some sub-subclasses, keeping only the highest level subclasses (they were always two or three level subclasses). For US Class 323 for example, we had initially 174 subclasses of different levels and after this processing, we got exactly 22 subclasses of two- and three- level, who have no direct connection with each other.

Since our method is a supervised one, we had to split the DW and DC matrices in training and test parts. The DC ”test part” (which contains the labels of the documents used for test) was used to evaluate the results. The training documents were selected such that for each subclass at least one document for each class will be selected. In Table 1 one can see the dimensions of the matrices we worked with.

US class	Nr. trains	Nr. tests	Nr. subclasses	Nr. words
036	4437	196	24	12561
175	5654	229	63	12866
219	12721	3162	18	33136
307	3476	1125	26	15077
318	9460	543	60	22082
323	3503	421	22	12148
329	545	185	7	5621
330	4590	1507	50	14702
331	3738	517	60	13403
332	506	181	8	5808
338	1991	251	46	11384
343	5601	768	3	17135
361	16468	3329	15	33372
372	7567	1081	22	25621
379	11240	1585	48	26584
703	2439	343	6	21552
704	5340	726	4	24036
706	1893	362	10	20943
707	9357	1860	9	36394

Table 1. The dimensions of the matrices

3.2 Performance measure

For measuring the performance of a supervised learning algorithm on a multi-class classification problem, it is customary to use the break-even point (BEP) measure which, for a binary categorization problem, is the arithmetic average of precision (P) and recall (R). Let us consider the problem of classifying documents into l classes $c_1, \dots, c_l$, using classifiers $M_1, \dots, M_l$, where the classifier M_i is responsible to discriminate between the class c_i and the rest of the classes. We considered a multi-class classification problem, where each document can be classified in many classes.

If one considers for each class c_i the following numbers

f_{++} = number of documents classified in c_i and which should be in c_i,

f_{+-} = number of documents classified out of c_i but which should be in c_i,

f_{-+} = the number of documents classified in c_i, but which should not be there,

one can compute the so-called precision and the recall with the formulas

$$P_i = \frac{f_{++}}{f_{++} + f_{+-} + 1} \text{ and } R_i = \frac{f_{++}}{f_{++} + f_{-+} + 1}.$$

One can introduce

$$BEP_i = \frac{P_i + R_i}{2}, \text{ for each class } c_i.$$

We also defined the partial average BEP as

$$BEP_{av\_part} = (BEP_1 + \cdots + BEP_l)/l_0$$

where BEP_i denote the break-even point corresponding to the class i and l_0 is the number of classes for which $BEP \neq 0$. We will report the partial average BEP and the best BEP for some US classes and also some particular cases.

3.3 Results and discussions

Table 2 summarizes the multi-class classification results obtained with the MI-SVM method on some US patent classes (where Time^1 is the training time, measured in seconds, and Time^2 is the test time). The programs were run on a Intel(R) Pentium 4 CPU, 1.60 GHz, 1GB Ram, under Linux.

A natural question that may arise is why have we introduced the BEP_{av_part} measure?

The distribution of the data in the subclasses of the USPCS is not uniform in the sense that there are subclasses which contain large numbers of documents and there are others which are "almost" empty. For instance, the documents from US 343-*Radio wave antenna* are grouped (after preprocessing) into three subclasses, as follows (please note that a document may appear in more than one class): 343/700R contains 6241 documents, 343/DIG1 contains only 5 and 343/DIG2 contains 83. Because of this big negative bias for the second subclass, our program was not able to classify any document there. That is why there are only two of three subclasses for which $BEP \neq 0$ (see the last column of the table). In the same time, we recognized that there is only a small number of documents spread in such classes (in our example, at most five of 6369). It is not a big loss when we do not take these classes (and the corresponding documents) into consideration. The BEP_{part_av} measure this average BEP only for classes in which the number of documents is comparable (and in conclusion the BEP is not 0).

Class	K	Time^1	Time^2	BEP_{av_part}	Best BEP	Cls. BEP>0
036	7000	-	-	0.476	0.875	7
175	7000	-	-	0.472	0.79	19
219	8000	1670	340	0.612	0.918	9
307	7000	523	154	0.516	0.896	8
318	8000	2972	893	0.5	0.5	1
323	5000	331	37	0.558	0.79	6
329	1000	5	2	0.480	0.805	6
330	7000	1598	552	0.46	0.895	14
331	7000	1291	166	0.556	0.77	16
332	1000	5	2	0.523	0.65	5
338	5000	338	102	0.533	0.79	8
343	7000	163	21	0.64	**0.98**	2
361	10000	3030	648	**0.735**	0.9	9
372	10000	1424	178	0.57	0.87	10
379	10000	4559	566	0.53	0.775	22
703	9000	259	36	0.508	0.76	6
704	9000	158	46	0.716	0.975	3
706	9000	157	27	0.615	0.88	6
707	10000	733	124	0.683	0.81	3

Table 2. Classification results with MI-SVM

We used SVM^{lights} in its standard variant (i.e. all the parameters have standard values, see [16]). We had also made tests with different values for K, between 500 and 20000 (depending on the size of the dictionary). Good results are produced with K between 5000 and 15000. We also varied the number of training documents, learned a model with fewer or more examples and the number of test documents. We have also tried to classify non-patent documents.

The best partial average BEP was 0.735 and a best particular BEP value for one class was 0.98. We have not reached in general the values reported in [5] or in [2] (there the tests have not been done for patent documents and the parameters of SVM were tuned).

One can say that when the distribution of the data in classes is (more) equilibrate, the average BEP is also better (and so the classification quality). But the algorithm is also sensitive with respect to the choice of K and the training set. The same number of documents but different ones in the training set produce different results. One must take into consideration also the quality of the documents-words matrices and of the vocabulary. We could not test how the program distinguishes between two or more US main classes because we had not enough store capacity to build the input matrices. This should be done in the future using computer systems having more store capacity and computing power (e.g. parallel computing systems).

Table 3 contains some particular examples. After the training part was done, we took some particular patent documents and tried to classify them on the basis of the learned model, using our program.

Patent number	Should be classified in	Was classified in
US0406457261	036/083 036/25R 036/34R	036/083 036/25R
US05075983	036/045 036/083	036/045 036/083
US05287639	036/083	036/083
US06092307	036/012 036/25R	036/083
US06167640	036/012 036/083	036/043 036/083
US04694831	036/043 036/083 036/140	036/043 036/083 036/140
US06360830	175/082 175/085	175/052 175/085
US05381867	175/085	-
US06223839	175/092 175/207	175/092 175/207
US04828053	175/073 175/092	175/073 175/092
US04844180	175/092 175/320	175/092

Table 3. classification of particular documents

The patent number and the classes names are taken from [18]. To find out also the name of the patent and other details see also [18]. We can see that although there are cases when misclassifications (line 5 for example) appear (this means differences between our results and the hand-made classification from the USPCS), there are still a lot of cases where the results of our program are remarkable.

So we may conclude that the MI-SVM combination is a promising method for the classification of patent documents. Although our average BEP does not reach the values reported until now, there are still particular classes with better results (see US 343 with a best BEP of 0.98, the best one reported until now).

Acknowledgments. The authors would like to thank Dr. U. Bauerschäfer from the firm IP Century AG (www.ipcentury.com) for providing the test data and for the fruitful collaboration. The first author would also like to thank the Federal Ministry for Economy and Technology (Germany) which financially supported this research through the PRO INNO program.

References

[1] L.D. Backer, A.K. McCallum, Distributional clustering of words for text categorization. Proceedings of SIGIR'98, 1998.

[2] R. Bekkerman, R. El-Yaniv, N. Tishby, Y. Winter, On Feature Distributional Clustering for Text Categorization. SIGIR'01, September 9-12,New Orleans, Louisiana, USA, 2001.

[3] C. Cortes, V.N. Vapnik, Support-vector networks. Machine Learning Journal, 20, pp. 273-297, 1995.

[4] N. Cristianini, J. Shawe-Taylor, Support Vector Machines and other kernel based learning methods. Cambridge University Press, 2000.

[5] S. Dumais, J. Platt, D. Heckerman, M. Sahami, Inductive Learning Algorithms and Representations for Text Categorization. Proceedings of ACM-CIKM'98, 1998.

[6] D. Koller, M. Sahami, Hierarchically Classifying Documents using very few words. Proceedings of CML'97, pp. 170-178, 1997.

[7] T. Joachims, Text Categorization Using Support vector machines: Learning with many relevant features. Proceedings of the Tenth European Conference on Machine Learning, pp. 137-142, 1998.

[8] T. Joachims, Learning to Classify Text Using Suport Vector Machines. Methods, Theory and Algorithms. Kluwer Academic Publishers, 2002.

[9] G. Salton, M. McGill, Introduction to modern information retrieval, McGraw Hill, 1983.

[10] N. Sloanim, N. Tishby, Aglomerative Information Bottleneck. Advances in Neural Information Processing Systems, pp. 617-623, 2000.

[11] N. Sloanim, N. Tishby, The Power of Word Clustering for Text Classification. Proceedings of European Colloquium on IR Research, ECIR, 2001.

[12] V.N. Vapnik, The Nature of Statistical Learning Theory. Springer, New York, 1995.

[13] Y. Yang, J. O. Pederson, A Comparative Study on Feature Selection for Text Categorization. Proceedings of ICML'97, pp. 412-420, 1997.

[14] ftp://ftp.cs.cornell.edu/pub/smart/english.stop

[15] http://www.tartarus.org/∼martin/PorterStemmer

[16] http://svmlight.joachims.org

[17] http://www.research.att.com/lewis

[18] http://www.uspto.gov