„Lucian Blaga” University of Sibiu
“Hermann Oberth” Engineering Faculty
Computer Science Department


# Relevant characteristics extraction from semantically unstructured data

3rd PhD Report

PhD Thesis Title: “Data Mining for Unstructured Data”

Author:
Daniel MORARIU, MSc

Supervisor:
Professor Lucian N. VINTAN, PhD

SIBIU, 2006

# Contents

# 1 Introduction

Most data collections from real world are in text format. Those data are considered semi structured data because they have a small organized structure. Modeling and implementing on semi structured data from recent data bases grows continually in the last years. More over, information retrieval applications, as indexing methods of text documents, have been adapted in order to work with unstructured documents.

Traditional techniques for information retrieval became inadequate for searching in a large amount of data. Usually, only a small part of the available documents are relevant for the user. Without knowing what is in the documents it is difficult to formulate effective queries for analyzing and extracting interesting information. Users need tools to compare different documents like effectiveness and relevance of documents or finding patterns to direct them on more documents.

There are an increasing number of online documents and an automated document classification is an important task. It is essential to be able to automatically organize such documents into classes so as to facilitate document retrieval and analysis. One possible general procedure for this classification is to take a set of pre-classified documents and consider them as the training set. The training set is then analyzed in order to derive a classification scheme. Such a classification scheme often needs to be refined with a testing process. After that, this scheme can be used for classification of other on-line documents. The classification analysis decides which attribute-value pairs set has the greatest discriminating power in determining the classes. An effective method for document classification is to explore association-based classification, which classifies documents based on a set of associations and frequently occurring text patterns. Such an association-based classification method proceeds as follows: (1) keywords and terms can be extracted by information retrieval and simple association analysis techniques; (2) concept hierarchies of keywords and terms can be obtained using available term classes, or relying on expert knowledge or some keyword classification systems. Documents in the training set can also be classified into class hierarchies. A term-association mining method can then be applied to discover sets of associated terms that can be used to maximally distinguish one class of documents from another. This produces a set of association rules for each document class. Such classification rules can be ordered - based on their occurrence frequency and discriminative power - and used to classify new documents.

Text classification is a very general process that includes a lot of requirements that need to be made in order to solve the problem. One of those requirements has a high influence on the final accuracy of classification. This can be seen as a flow where each part receives some information, process it and then further transfer it, see Figure 1.1. Each part of the flow can have more than one algorithm attached to it. At a certain time, for each part we can choose one of the attached algorithms and modify its input parameters. In our first two PhD technical reports we presented some parts of this flow (grey parts in the Figure 1.1); in this technical report we want to complete some of these parts with new methods (white parts in the figure) and finalize this chain.

Thus, in the first PhD report [Mor05_1] we presented some techniques for preprocessing the text documents. Especially we presented preprocessing of the Reuters 2000 database. We continued with a short introduction of web mining processing, especially what is the difference between those two general concepts. In this step the input is text documents (text files or web pages) and

we represent them into the form of a feature vector. These are frequency vectors of words that occur into the document. This representation is closer to the understanding of the computer. This step contains a module of eliminating the stop-words, one module of extracting the root of the word and count word occurrences. Due to huge dimensionality of resulting vectors this step continues with a step of selecting relevant features. Thus, the second PhD report continues by presenting three methods of feature selection: Random selection, Information Gain selection and Support Vector Machine selection. A new feature selection method based on Genetic Algorithm is developed and presented in this report.

The second technical report presents in details the algorithm used for classification based on Support Vector Machine (SVM) technique [Pla99]. There we focused in general on the SVM as classification process. We presented then a method of kernel's correlation and its improvement in comparison with LibSVM implementation. In this report we also present results that justify the methodology of choosing the kernel correlation.

The flow ends with the development of a meta-classifier in order to improve classification accuracy. This method is presented in this report and obtains results for more classifiers based on SVM and tries to explore the classifiers' synergism.

This PhD technical report also contains some contributions that improve the flow of classification by making it more reliable. First of them is the ability of our application to work with a lot of documents. We usually present results for a relatively small dimension of the data set. In this report we present a methodology that makes our application able to work with a much larger dimension of the data set, and with small loses as far as the accuracy is concerned. This methodology has to antagonist main objectives – more data in the set and smaller response time with good accuracy.

For more realistic results the same data set should be split more times in training – testing sets pairs. For each pair we should compute the accuracy of classification and present an average over all obtained accuracies. In all presented results the accuracy is computed for a single pair. In the last chapter of this report we presented a part of the results obtained on more than one pair. The obtained results would enable us to justify our methodologies. These results would also provide us with a level of acceptance for accuracy as being good by computing the average of the accuracies obtained over all these training – testing sets pairs.

Our experiments are performed on the Reuters-2000 collection [Reu00], which has 948 Mb of newspapers articles in a compressed format. The collection includes a total of 806,791 documents, with news stories published by Reuters Press Agency covering the period from 20.07.1996 through 19.07.1997. The articles have 9822391 paragraphs and contain 11522874 sentences and 310033 distinct root words. Documents are pre-classified according to 3 categories: by the *Region* (366 regions) the article refers to, by *Industry Codes* (870 industry codes) and by *Topics* proposed by Reuters (126 topics, 23 of them contain no articles). Due to the huge dimension of the database we will present here results obtained using a subset of data. From all documents we selected the documents for which the industry code value is equal to "System software". We obtained 7083 files that are represented using 19038 features and 68 topics. We represent documents as vectors of words, applying a stop-word filter (from a standard set of 510 stop-words) and extracting the steam of the word. From these 68 topics we have eliminated those topics that are poorly or excessively represented. Thus we eliminated those topics that contain less than 1% of the documents from all 7083 documents in the entire set. We also eliminated topics that contain more than 99% of the samples from the entire set, as being excessively represented. The elimination was necessary because with these topics we have the risk to use only a single decision function for

classifying all documents, ignoring the rest of the decision functions. After doing so we obtained 24 different topics and 7053 documents that were split randomly in training set (4702 samples) and testing set (2531 samples). In the feature extraction part we took into consideration the entire article and the title of the article in order to create the characteristic vector.

Next chapter contains experiments that lead to the choice of these kernels correlations. Chapter three contains a new method for feature selection based on Genetic Algorithms with Support Vector Machine technique used for the fitness function. In chapter four we finalize the document classification process by presenting some methods for implementing a meta-classifier in order to improve the final classification accuracy. Chapter five presents the influence of the amount of input data when our algorithm needs to work with huge quantity of data. There we also presented a strategy to make our algorithm work faster when it needs to use huge quantities of input data. In chapter six some results obtained using a new distribution of the training and testing set are presented in order to see the influence of the dataset. The last chapter debates and presents most important results obtained and it proposes some further work.

## Acknowledgments

At the end of this introduction I want to express my sincere gratitude to my PhD supervisor prof. dr. eng. Lucian VINTAN for his scientific coordination during this PhD stage, as well as in the final part of the stage when I will be writing my final thesis. I would also like to thank the ones that guided me from the beginning of my PhD studies: Prof. Ioana MOISIL, Prof. Boldur BĂRBAT, prof. Daniel VOLOVICI, dr. ing. Dorin SIMA and Dr. ing. Macarie BREAZU for their valuable generous professional support.

At the same time, I would like to thank SIEMENS AG, CT IC MUNCHEN, Germany, especially Vice-President Dr. h. c. mat. Hartmut RAFFLER, for his very useful professional suggestions and for the financial support that they have provided. I want to thank my tutor from SIEMENS, Dr. Volker TRESP, Senior Principal Research Scientist in Neural Computation, for the scientific support provided and for his valuable guidance in this wide interesting domain of research. I also want to thank Dr. Kai Yu for his useful information in the development of my ideas. Last but not least, I want to thank all those who supported me in the preparation of this technical report.

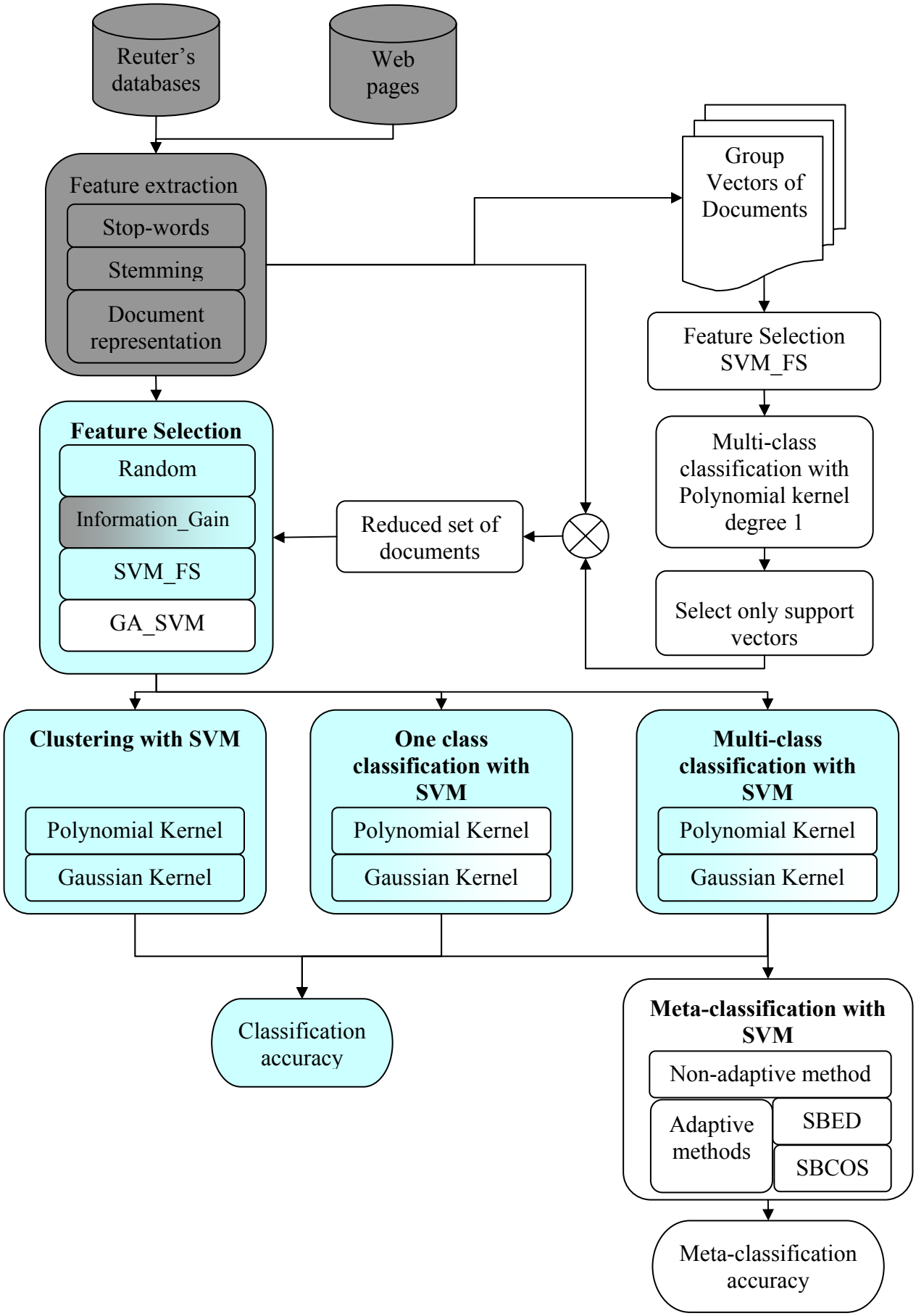


**Figure 1.1 – Documents classification flowchart**

# 2 Correlation of the SVM kernel's parameters

Documents are typically represented as vectors of the features space. Each word in the vocabulary represents a dimension of the feature space. The number of occurrences of a word in a document represents the value of the corresponding component in the document's vector. The native feature space consists of the unique terms that occur into the documents, which can be tens or hundreds of thousands of terms for even a moderate-sized text collection, being a major problem of text categorization.

As a method for text classification we use Support Vector Machine (SVM) technique [Sch02], [Nel00], which was proved as being efficient for nonlinear separable input data. This is a relatively recent learning method based on kernels [Vap95], [Pla99]. We use this method both in the features selection step (feature selection based on SVM) and in the classification step. This method was presented in detail in the second PhD report [Mor05]. In that report we present a comparison of results obtained using my application with the ones obtained using LibSvm [Lib], a common implementation of SVM used in the literature. Here I want to presents experiments that lead to the choice of these kernel correlations.

The idea of the kernel is to compute the norm of the difference between two vectors in a higher dimensional space without representing those vectors in the new space. Adding a scalar constant to the kernel involves better classifying results. In this paper we tested a new idea to correlate this scalar with the dimension of the space where the data will be represented. Thus we consider that those two parameters (the degree and the scalar) need to be correlated.

Those contributions were also published in paper [Mor06_1]. We intend to scale only the degree for the polynomial kernel and only constant *C* for the Gaussian kernel according to the following formulas (*x* and *x'* being the input vectors):

Polynomial kernel:

$$k(\mathbf{x},\mathbf{x}') = \left(\mathbf{2}\cdot d + \langle \mathbf{x},\mathbf{x}' \rangle\right)^d \tag{2.1}$$

*d* being therefore the only parameter that needs to be modified and

Gaussian kernel:

$$k(\mathbf{x},\mathbf{x}') = \exp\left(-\frac{\|\mathbf{x}-\mathbf{x}'\|^2}{n\cdot C}\right) \tag{2.2}$$

where *C* is the only parameter that needs to be modified. The parameter *n* that occurs in the formula is an automatically computed value that represents the number of elements greater than zero from the input vectors.

## 2.1 Polynomial kernel parameters correlation

Usually when learning with a polynomial kernel researchers use a kernel that looks like:

$$\left(\langle \mathbf{x} \cdot \mathbf{x}' \rangle + b\right)^d \tag{2.3}$$

where *d* and *b* are independent parameters. "*d"* is the degree of the kernel and it is used as a parameter that helps mapping the input data into a higher dimensional features space. This is why this parameter is intuitive. The second parameter "*b"*, is not so easy to infer. In all studied articles, the researchers used it, but they don't present a method for selecting it. We notice that if this parameter was eliminated (i.e., chosen zero) the quality of results can be poor. It is logical that we need to correlate the parameters *d* and *b* because the offset *b* needs to be modified as the dimension of the space is modified. Due to this, based on running laborious classification simulations presented in this paper, we suggest using "*b=2*d"* in our application.

## 2.2 Gaussian kernel parameters correlation

For the Gaussian kernel we have also modified the standard kernel used by the research community. Usually the kernel looks like:

$$k(x, x') = \exp\left(-\frac{\|x - x'\|}{C}\right) \tag{2.4}$$

where the parameter *C* is a number that represents the dimension of the training set (and it is usually a very big number in text classification problems). We introduce a new parameter *n* which is a value that represents the number of distinct features that occur into the current two input vectors (*x* and *x'*), having weights greater than 0. This parameter is multiplied by parameter *C*. We kept the notation *C* for a parameter that becomes a small number (usually we obtain best results between 1 and 2).

As far as we know, I am the first author that propos a correlation between these two parameters for both polynomial and Gaussian kernels.

## 2.3 Results for polynomial kernel

In order to improve the classification accuracy using polynomial kernel our idea was to correlate the kernel's bias with the kernel's degree. In this idea we developed tests for four kernel's degrees, considering for each of them 16 distinct values of the bias and, respectively, for each input data representation. Thus for each degree of the kernel we vary the value of the bias from 0 to the total number of features (presenting here only results obtained for 16 distinct values).

I will present here results obtained using a set with 1309 dimensions because, as we showed in the previous PhD report, the best results were obtained with it. So that, in presented cases, we vary the bias from 0 to 1309. Usually in the literature the bias is selected between 0 and the total number of features.

In Figure 2.1 we present results obtained with polynomial kernel and Nominal data representation by varying the degree of the kernel and the bias. In "Our choice" entry we put only the values that were obtained using our formula that correlates the polynomial kernel's parameters. As it can be observed, using our correlation (equation 2.1) assures that in almost all cases we obtain the best results. In this case, only for degree 4 the best value was obtained for bias equal with 2 and with our formula we obtained a value with 0.21% smaller than the best obtained results (84.56% in comparison with the best obtained 84.77%).

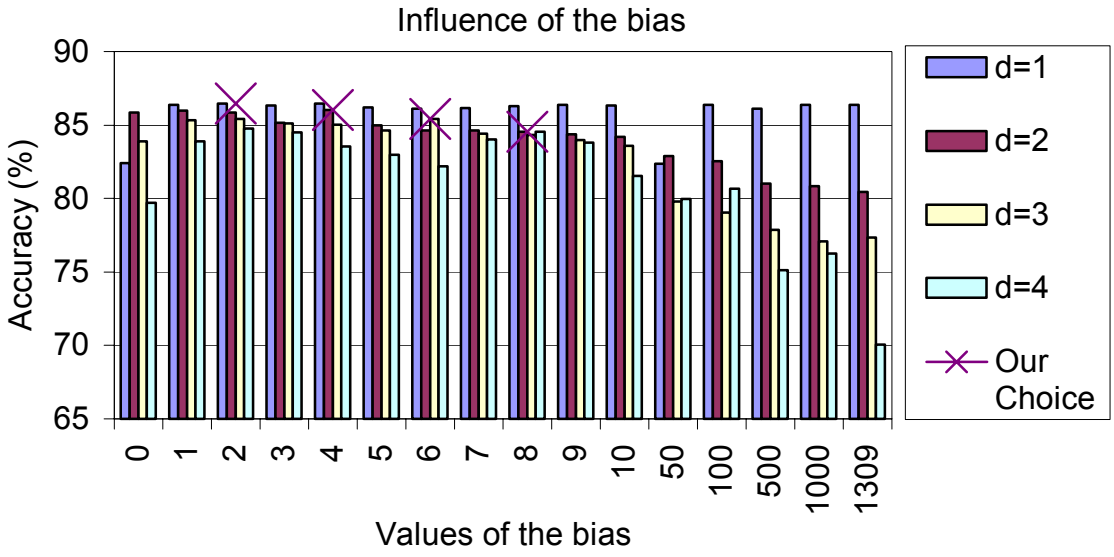


**Figure 2.1 – Influence of bias for Nominal representation of the data**

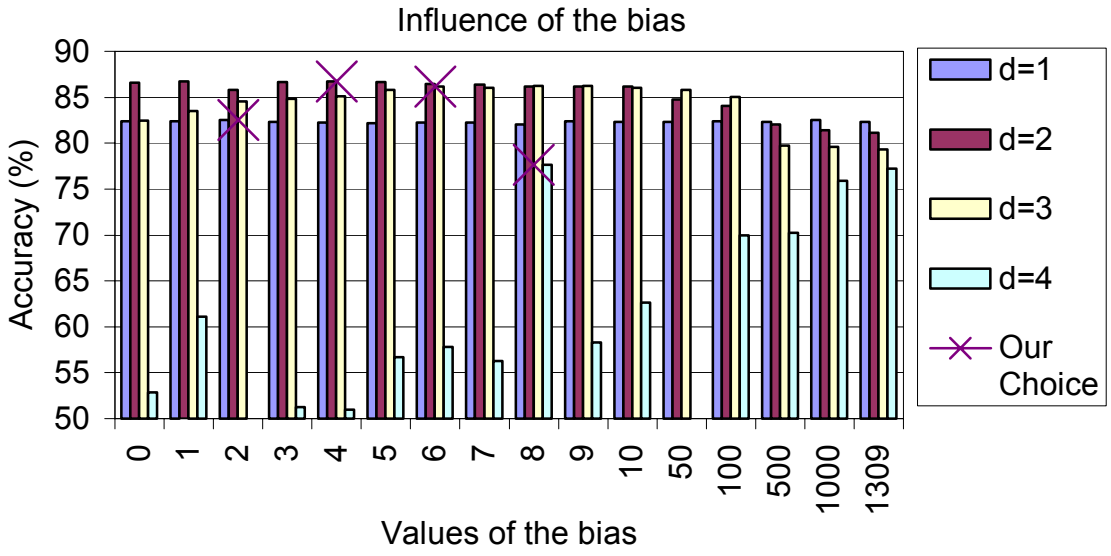


**Figure 2.2 – Influence of bias for Binary representation of the input data**

Effective values of the accuracy obtained using Cornell Smart data representation, for each kernel degree and for each bias, are presented in Table 2.1. For each degree there are multiple bias values involving best results and our proposed formula assures to hit these values in almost all cases. Also an interesting remark is that for kernel degree equal to 1, we usually obtained the same classification accuracy for all bias values, with only 0.51% smaller that the best value. As it can be

observed from Figure 2.1, with no bias we obtain the worst results. The same tests were developed also for Binary data representation and also have more values that obtain best results but we hit these values in almost all cases (Figure 2.2). Only for degree 3 the best value was obtained for bias equal with 8 with 0.085% greater than our choice.

| Bias | D=1 | D=2 | D=3 | Our Choice |
|---|---|---|---|---|
| 0 | 81.84 | 86.69 | 82.35 | |
| 1 | 81.84 | 86.64 | 83.37 | |
| 2 | **82.22** | **86.81** | 84.01 | **82.22** |
| 3 | **82.22** | 86.56 | 84.77 | |
| 4 | **82.22** | **86.81** | 65.12 | **86.81** |
| 5 | 82.01 | 86.47 | 85.54 | |
| 6 | 81.71 | 86.60 | 86.39 | **86.39** |
| 7 | 81.71 | 86.43 | 86.18 | |
| 8 | 82.09 | 86.43 | **86.47** | |
| 9 | 81.84 | 86.18 | **86.47** | |
| 10 | 81.80 | 85.96 | 86.26 | |
| 50 | 81.92 | 84.73 | 84.90 | |
| 100 | **82.22** | 83.71 | 82.82 | |
| 500 | 82.05 | 81.88 | 8.34 | |
| 1000 | 82.09 | 80.94 | 53.51 | |
| 1309 | 82.09 | 80.77 | 50.40 | |

**Table 2.1 Bias influence for CORNELL SMART data representation**

## 2.4 Results for Gaussian kernel

For the Gaussian kernel we modified the usually used parameter *C* that represents the number of features from the input vector, with a product between a small number (noted also *C* in our formula 2.2) and a number *n* that is computed automatically. We present here tests with four distinct values of *C*. For each of these values, we vary *n* from 1 to 1309 (total number of the features used). Because our proposed value for *n* is automatically computed, this number can not be specified by the command line, so that for each value of constant *C* we specified a value called "auto" (in Figure 2.3) that means the value automatically computed using our formula.

We made tests only for Binary and Cornell Smart representations of the input data. Into Gaussian kernel we fill in a parameter that represents the number of elements greater then zero (parameter "*n*" from equation 2.2). Nominal representation (second PhD report Section 4.4) represents all weight values between 0 and 1. When parameter "*n*" is used, all the weights become very close to zero involving very poor classification accuracies (for example, due to its almost zero weight, a certain word really belonging to the document, might be considered as not belonging to that document). So we don't present here the results obtained using Nominal representation and Gaussian kernel.

In Figure 2.3 we present results obtained for Binary data representation. When we use our correlation, we obtained the best results. Also better results were obtained when the value of *n* is between 50 and 100. This occurs because usually each document uses a small number of features (on average between 50 and 100) in comparison with features from the entire set of documents. When *n* is equal with the total number of features (usually used into the literature) the accuracy decrees, on average for all tests, with more than 10% in comparison with using the automatically computed value for *n*. It can also be observed that when the value of parameter *n* increases the accuracy substantially decrees. The same tests were also made using Cornell Smart data representation, obtaining the same tendency and usually accuracy with 1% better than in Binary case (Figure 2.4).

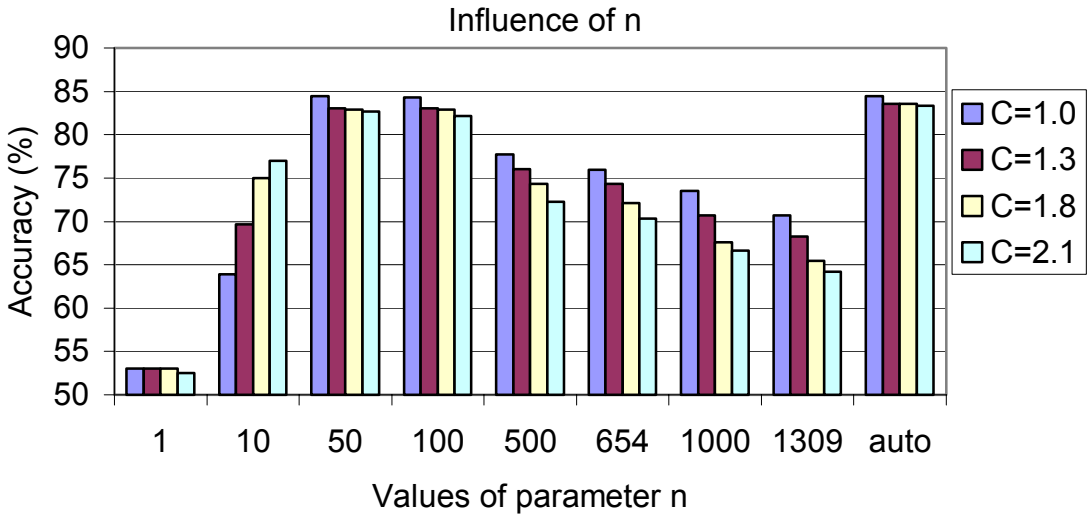


**Figure 2.3 – Influence of the parameter "*n*" for Gaussian kernel and Binary data representation**

We don't present results for Nominal representation here because usually, independent of parameters *n* and *C*, the results are poor (the accuracy is between 40% and 50%). In contrast with the polynomial kernel, in the Gaussian kernel case we obtained best results only with our proposed formula to compute the parameter "*n*".

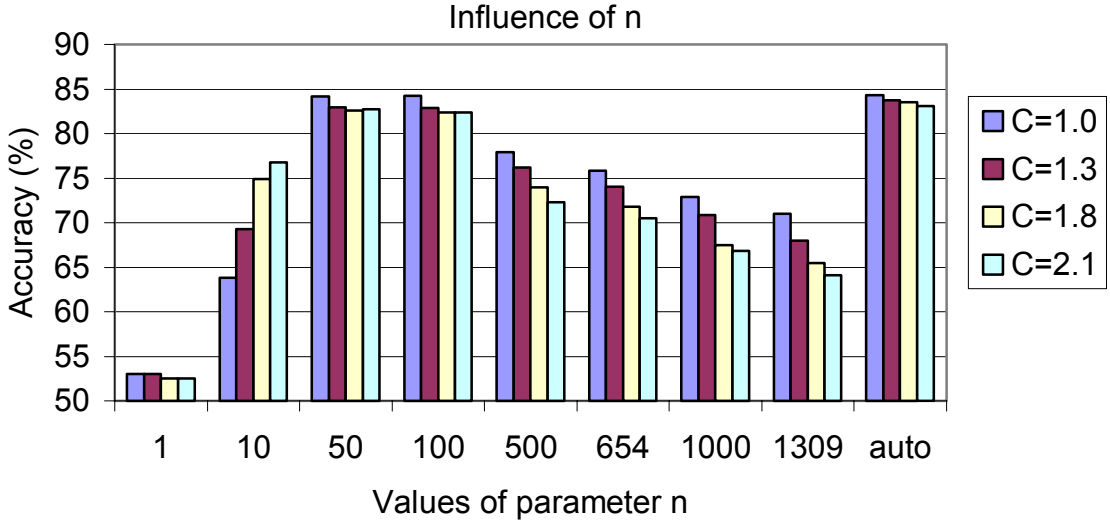


**Figure 2.4 – Influence of the parameter *n* for Gaussian kernel and Cornell Smart data representation**

# 3 Features selection using Genetic Algorithms

In the field of documents classification, features can be characterized as a way to distinguish one class of objects from another in a more concise and meaningful manner than is offered by the raw representations. Therefore, it is of crucial importance to define meaningful features when we plan to develop an accurate classification, although it has been known that a general solution has not been found. In many practical applications, it is usual to encounter samples involving thousand of features. The designer usually believes that every feature is meaningful for at least some of the discriminations. However, it has been observed in practice that, beyond a certain point, the inclusion of additional features leads to worse rather then better performances [Kim00]. Furthermore, including more features means simply increasing processing time. Some features don't help class discrimination, and feature selection is a redundant reduction by retaining useful features and gets rid of the useless ones.

Feature subset selection is defined as a process of selecting a subset of features, *d*, out of the larger set of *D* features witch maximize the classification performance of given procedure over all possible subsets [Gue00]. Searching for an accurate subset of features is a difficult search problem. Search space to be explored could be very large, as in our Reuter's classification problem in which there are $2^{19034}$ possible features combinations! A feature selection process using Genetic algorithms is used in order to isolate features that provide the most significant information for the classification, whilst cutting down the number of inputs required.

In this section we introduce a feature selection method, which can minimize most of the problems that can be found in the conventional approaches, by applying genetic algorithms. I combined a powerful and rigorous mathematical method based on kernels with a randomly starting point permit by genetic algorithm. In [Jeb00] and [Jeb04] are explained the advantage of using the same method in the feature selection step and in the learning step.

## *3.1 The genetic algorithm*

Genetic algorithms are a part of evolutionary computing, and are inspired by Darwin's theory on evolution. The idea of evolutionary computing was introduced in 1960s by I. Rechenberg in his work "Evolution strategies".

Genetic algorithms have been gaining popularity in a variety of application which requires global optimization of solution. A good general introduction to genetic algorithms is given in [Gol89] and [Bal97]. Genetic algorithms are based on a biological metaphor: "They view learning as a competition among a population of evolving candidate problem solutions."[Lug98].

The Genetic algorithm refers to a model introduced and investigated by J. Holland in 1975 [Hol75], and are adaptive procedures that find solutions of problems based on the mechanism of natural selection and natural genetics. The algorithms have strength over the problems, in which finding the optimum solution is not easy or inefficient at least, because of their characteristics of probabilistic search. Genetic algorithms are a family of computational models inspired by evolution. These algorithms encode a potential solution to a specific problem on a simple

chromosome-like data structure and apply recombination operators to these structures so as to preserve critical information.

Generally, genetic algorithms start with an initial set of solutions (usually choused randomly) called into the literature, population. In this population each individual is called "chromosome" and represent a solution to the problem. In almost all cases a chromosome is a string of symbols (usually represented by a binary bit string). These chromosomes evolve during successive iteration, called generations. In each generation, the chromosomes are evaluated using some measures of fitness. For creating the next generation the best chromosomes from current generation are selected and the new chromosomes are formed by three essential operations: selection, crossover and mutation.

Selection assures that some chromosomes from current generation are copied according to their objective function values into the new generation. Copying strings according to their fitness values means that string with a higher value will have a higher probability of contribution one or more offspring in the next generation. Another operation used to create new generation is crossover that is a process of meaning two chromosomes from current generation to form two similar offsprings. The mutation is the process of modifying a chromosome and occasionally one or more bits of the string are altered while the process is being performed.

### 3.1.1 Chromosomes encoding and optimization problems

Usually there are only two main components of most genetic algorithms that are problem dependent: the problem encoding and the evaluation function. The chromosome should in some way to contain information about the solution which it represents and very depend on the problem. There are more encodings, which have been already used with some success like binary encoding, permutation encoding, value encoding and tree encoding. The evaluation function, called also fitness function, is function that allows giving a confidence to each chromosome from the population. In my representation I chose value encoding.

I will use the genetic algorithm in the feature selection step for eliminating the most unnecessary features. In this step I have a set of 7083 documents represented by a vector of features having each of the 19038 entries. Those documents are pre-classified by Reuters into 24 classes. In the features selection step we take into consideration all entries from the input vector and we try to eliminated those entries that are considered irrelevant for this classification. For the fitness function in the genetic algorithm I use Support Vector Machine with linear kernel. Concrete, in my approach the fitness function represents the classification accuracy computed using the decision function. To implement this I start from SVM_FS idea presented in the second PhD report [Mor05]. This is a powerful technique that has a general inconvenience based on the order of selecting the entry sample. With the genetic algorithm I try to propose more starting points in order to find the better solutions. The general scheme of the implemented genetic algorithm is presented in Figure 3.1.

Let $\{\langle \vec{x}_i, y_i \rangle, i = 1,...,m\}$ be a set of documents, where $m$ represents number of documents, $x_i$ represent the vector that characterizes a document and $y_i$ is the document topic. In the implemented algorithm the chromosome for optimization problem is considered to be of the following form:

$$c = (w_0, w_1, ..., w_{19038}, b) \tag{3.1}$$

where $w_i$, $i = 1,2,..,19038$ represent the weight for each feature, and $b$ represents the bias from decision function of SVM. In my approach each weight **w** represents a normal vector perpendicular to the hyperplane (that is characterized by the decision function). Thus potential solutions to the problem encode the parameters of separating hyperplane, $w$ and $b$. In the end of the algorithm, the best candidate from all generations gives the optimal values for separating hyperplane orientation $w$ and location $b$.

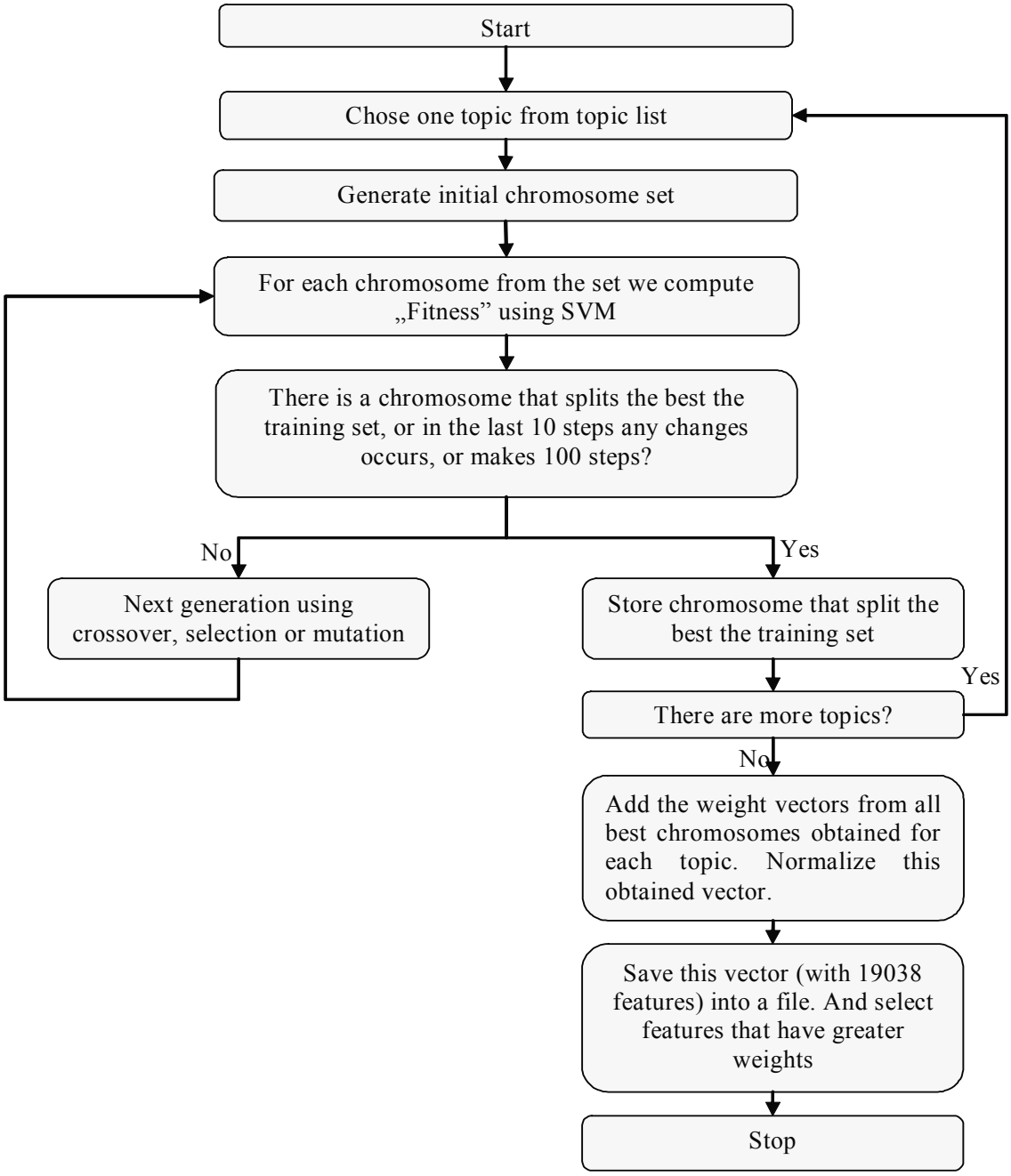


**Figure 3.1 - Genetic Algorithm**

We start with a generation of 100 chromosomes. Because the SVM algorithm is designed for working with two classes (the positive classes and the negative classes) we make, in our case, 24 distinct learning steps. Following the idea proposed for multi-class classification (one versus the rest), I try to find the best chromosome for each topic separately. For each topic I start with a

generation of 100 chromosomes generated randomly. In those generations each chromosome has the form specified earlier (equation 3.1). In each chromosome we put only half number of features, chosen randomly, different from 0. I make this because in the feature selection step we are interested to have sparse vectors in order to have a greater number of features that can be eliminated. For this step I chose $w \in [-1,1], b \in [-1,1]$. For example in first steps the chromosome looks like:

| | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Chromosome 1 | -0.23 | 0 | 0 | 0.89 | 0 | 0.52 | 0 | 0 | 0 | 0 | -0.04 | … | 0.03 |
| Chromosome 2 | 0 | 0 | 0.08 | -0.67 | -0.01 | 0 | 0 | 0 | 0 | 0.01 | 0 | … | 0.01 |

In the flow of genetic algorithm the value of **w** or *b* can extend the limitation of the generation step. Using the SVM algorithm with linear kernel that looks like $\langle w,x \rangle + b$, and considering the current topic, we can compute the fitness function for each chromosome. The fitness function transforms the measure of performance into an allocation of reproductive opportunities. The evaluation through the fitness function is defined as:

$$f(c) = f((w_1, w_2, ..., w_n, b)) = \langle \mathbf{w}, \mathbf{x} \rangle + b \tag{3.2}$$

where **x** represents the current sample and *n* represents the number of features. In the next step we generate the next population using the genetic operators (selection, crossover or mutation). The operating chart is presented in Figure 3.2. Into the step of generating the next population those operators are chosen randomly for each parent. We use selection operator in order to assure that the best obtained values from the current population aren't lost by putting them unchanged in the new population.

From the initial population we select two parents using two methods: Gaussian selection or Roulette Wheel [Whi94] selection. These methods will be explained later. Because we want to randomly select one of the operators to generate the next candidates, we randomly delete one of parents with a small probability to do this. Thus with a small probability we can have only selection or mutation and with a greater probability to have crossover. If in the new generation we find a chromosome that best splits (without error) the training set we stop and consider this chromosome as the decision function for the current topic. If not, we generate a new population and stop when in the last 100 generations no evolution occurs.

At the end of the algorithm, when we obtain for each topic from the training set a distinct chromosome that represents the decision function, we normalize each of the weight vectors in order that all weights are between -1 and 1. As for feature selection with SVM method we compute the average of all those obtained decision functions and obtain the final decision function. From this final decision function we take the weight vector and select only the features with an absolute value of the weight that exceeds a specified threshold.

### 3.1.2 Roulette Wheel and Gaussian selection

Another important step in genetic algorithm is how to select parents for crossover. This can be done in many ways, but the main idea is to select the better parents (hoping that the better parents will produce better offspring). A problem can occur in this step. Making the new population only

by new offsprings, can cause lose of the best chromosome from the previous population. Usually this problem is solved by using the so called elitism. This means, that at least one best solution is copied without changes into the new population, so the best solution found can survive to the end of the run.

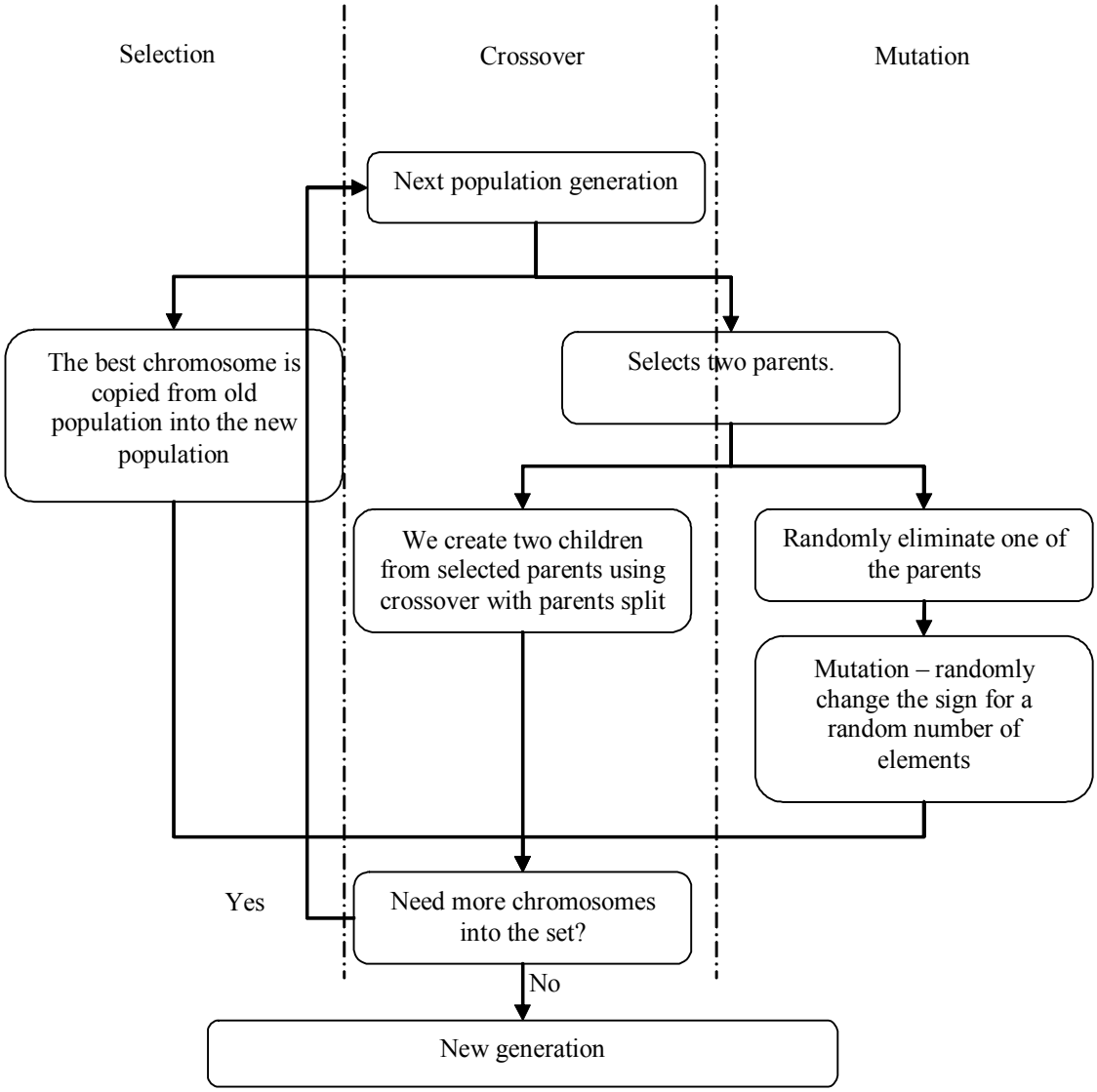


**Figure 3.2 – Obtaining the next generation**

Into Roulette Wheel selection method each individual from the current population is represented by a proportional space to its fitness function. By repeatedly spinning the roulette wheel, individuals are chosen using "stochastic sampling" that assures the good chromosome to have more chances to be selected. This selection will have problems when the fitness function differs very much. For example, if the best chromosome fitness is 90% of all, the Roulette Wheel is very unlikely to select the other chromosomes.

In the Gaussian selection method we randomly choose a chromosome from the current generation. Using equation (1.3) we compute the probability that the chosen chromosome is the best chromosome (that obtains the fitness function with maxim value *M*).

$$P(c_i) = \exp\left(-\frac{1}{2}\left(\frac{(fitness(c_i) - M}{\sigma}\right)^2\right) \quad (3.3)$$

where *P(.)* represents the probability computed for chromosome $c_i$, *M* represents the mean that here is the maximum value that can be obtained by the fitness function (my choice was *M* equal to 1) and $\sigma$ represent the dispersion (my choice was $\sigma = 0.4$).

This computed probability is compared with a probability randomly chosen, selected at each step. If the computed probability is greater than the probability randomly chosen then the selected chromosome will be used for creating the next generation, otherwise we choose randomly another chromosome for the test. This method assures the possibility of not taking into consideration only the good chromosomes.

## 3.1.3 Using genetic operators

### 3.1.3.1 Selection and mutation

Selection is the process in which individual strings are copied in the new generation according to their objective function values. Copying strings means that strings with a higher value will have higher probability of contribution to one or more offspring in the next generation. Also, in order that we don't loose the best solution obtained in the current generation we select the best chromosome obtained and copy it to the next generation (elitism). There are a number of ways to do selection. We are doing selection based on the Roulette Wheel or Gaussian.

Mutation is another important genetic operator and it is a process of modifying a chromosome and occasionally one or more bits of a string are altered while the process is being performed. The mutation depends on the encoding as well as the crossover. Mutation takes a single candidate and randomly changes some aspects of it. Mutation works by randomly choosing an element of the string and replacing it with another symbol from the code (for example changing the sign or changing the value).

| | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Original offspring 1 | **-0.23** | 0 | 0 | 0.89 | 0 | 0.52 | 0 | 0 | **0** | 0 | -0.04 | … | **0.03** |
| Original offspring 2 | **0** | 0 | 0.08 | -0.67 | -0.01 | 0 | 0 | 0 | 0 | **0.01** | 0 | … | 0.01 |
| Mutated offspring 1 | **0.23** | 0 | 0 | 0.89 | 0 | 0.52 | 0 | 0 | **-1.0** | 0 | -0.04 | … | **-0.03** |
| Mutated offspring 2 | **1.0** | 0 | 0.08 | -0.67 | -0.01 | 0 | 0 | 0 | 0 | **0.1** | 0 | … | 0.01 |

### 3.1.3.2 Crossover

Crossover can be viewed as creating the *next population* from the current population. Crossover can be rather complicated and it is very dependent on encoding of the chromosome. Specific crossover made for a specific problem can improve performance of the genetic algorithm. Crossover is applied to paired strings chosen using one of the presented methods. Pick a pair of strings. With probability $p_c$ "recombine" these strings to form two new strings that are inserted in

the next population. For example taken 2 strings from current population, divide them, and swap components to produce two new candidates. Those strings would represent a possible solution to some parameter optimization problems. New strings are generated by recombining two parent strings. Using a simple randomly chosen recombination point, we create new strings by recombining first part from one parent and second part from the other parent. After recombination, we can randomly apply a mutation operation for one ore more parameters. In my implementation we can have one or two randomly recombination points.

| | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Chromosome 1 | -0.23 | 0 | 0 | 0.89 | 0 | 0.52 | 0 | 0 | 0 | 0 | -0.04 | … | 0.03 |
| Chromosome 2 | **0** | **0** | **0.08** | **-0.67** | **-0.01** | **0** | **0** | **0** | **0** | **0.01** | **0** | **…** | **0.01** |
| Offspring 1 | -0.23 | 0 | 0 | 0.89 | **-0.01** | **0** | **0** | **0** | **0** | **0.01** | **0** | **…** | 0.03 |
| Offspring 2 | **0** | **0** | **0.08** | **-0.67** | 0 | 0.52 | 0 | 0 | 0 | 0 | -0.04 | … | **0.01** |

## *3.2 Experimental results*

For this experiment we present results obtained using Genetic Algorithm for feature selection (GA_FS) comparatively with results obtained with Support Vector machine for features selection (SVM_FS). The results will be presented for both types of kernels and for all the three types of data representation. These formulas were detailed in my second PhD report [Mor05]. For the Genetic algorithm we have used as a threshold the number of features that we want to use in order to be able to compare the two methods. The results obtained with SVM_FS were also presented in the second PhD report and were selected because they obtained the best results. Initially I also expected to obtain roughly the same results for GA_FS because the fitness function that I use in Genetic algorithm is the same function (SVM with linear kernel) as in SVM_FS method. But as we will see the results are sometimes worse than in SVM_FS method and sometimes better, especially when we increase the number of features.

For Polynomial kernel and vector dimension equal with 475 (Figure 3.3) all the time SVM_FS method obtains better results.

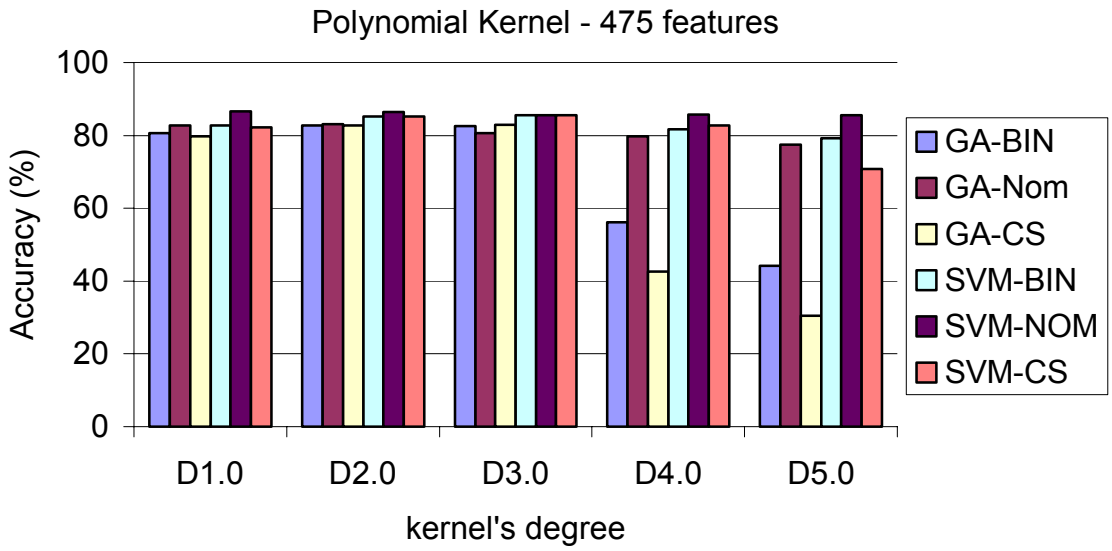


**Figure 3.3 – Comparison between results obtained using GA_FS and SVM_FS for polynomial kernel**

In the presented charts I denoted by GA – Genetic Algorithm and by SVM - Support Vector Machine for feature selection. We also denote by BIN - Binary representation for input data, by NOM – Nominal representation of input data and by CS - Cornell Smart representation of input data. As it can be observed when the degree of the kernel increases the classification accuracy doesn't increase so much, actually it decreases for GA_FS. This shows that for a small number of features the documents are linearly separable. For example the best value obtained for this dimension of the set was 86.64% for SVM_FS with Nominal representation and degree equal to 1.

In Figure 3.4 we present results obtained when the dimension of the vector increases to 1309. In the previous report we showed that for this dimension best results were obtained and also on average for all tested degrees we obtained the best results. GA_FS obtains better results (85.79%) for this dimension comparatively with the previous chart (83.07%). When we have fewer features and the degree increases the accuracy decreases for GA_FS. When we have more features and the degree increases the accuracy doesn't decrease significantly. But, for this dimension however SVM_FS obtains better results. The maximum value obtained is 86.68% for SVM_FS with nominal data representation and degree 1 comparatively with GA_FS that obtains only 85.79% for Cornell Smart data representation and degree 2. For this dimension of the feature space for all tested degrees the SVM_FS method obtains better results.

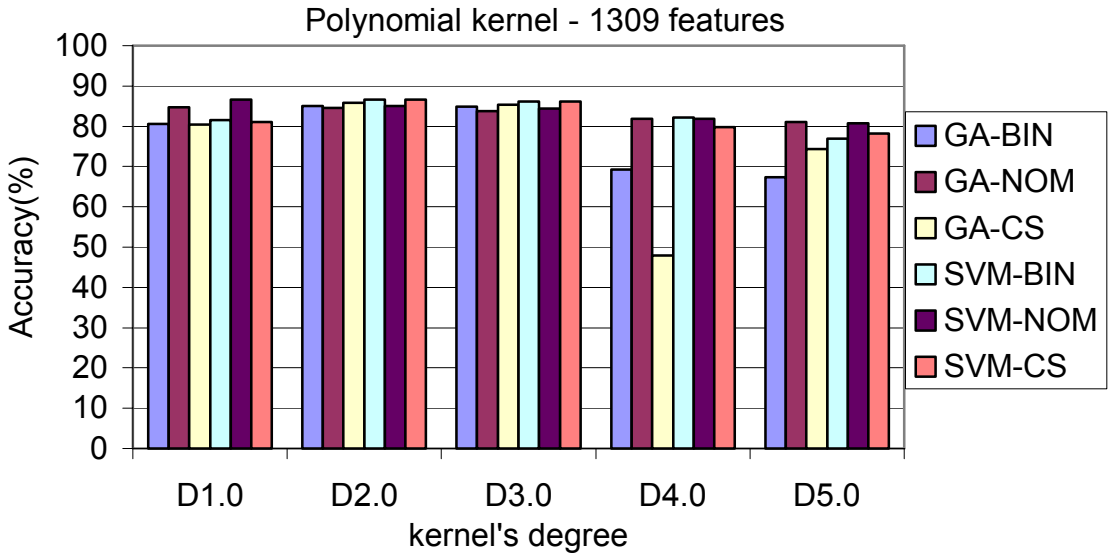


**Figure 3.4 – Comparison between GA_FS and SVM_FS for 1309 features**

When the vector dimension increases even more (Figure 3.5 and Figure 3.6) the accuracy of classification doesn't increase. Especially for SVM_FS, the maximum accuracy obtained decreases to 86.64% for 2488 features and 86.00% for 8000 features. For results obtained with GA_FS the accuracy of classification increases comparatively with the previous tested dimensions. Thus for 2488 features the maximum accuracy obtained increases to 86.94% but when the number of features increases more the accuracy decreases to 86.77%. Actually if GA_FS obtains for those greater dimensions better results comparatively with SVM_FS, those results exceed with only 0.26% results obtained by SVM_FS for 1309 features. But taking in consideration the time needed for training this excess takes 32 more minutes (as it can be seen in Figure 3.7).

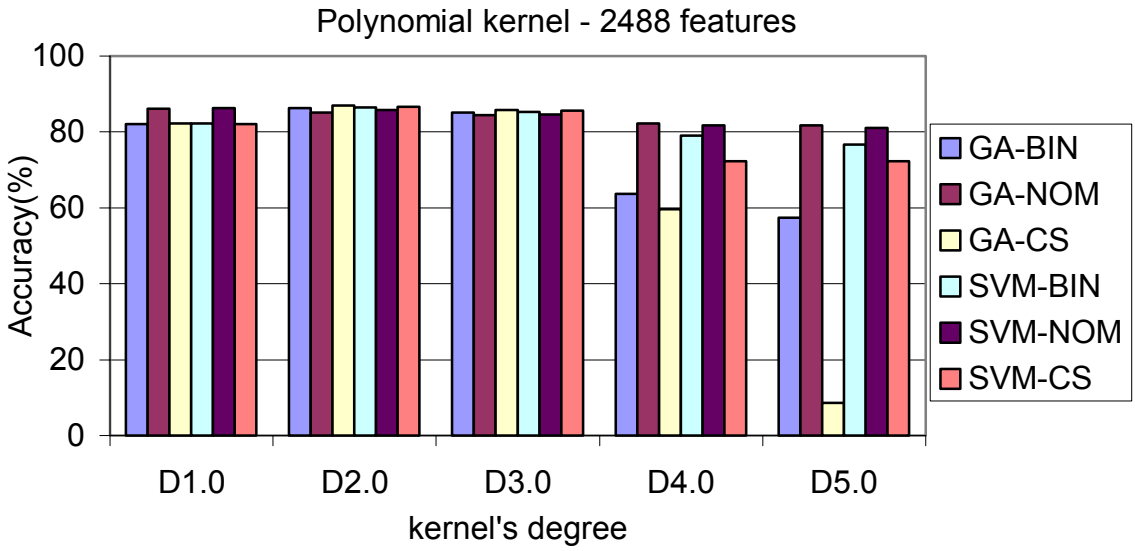


**Figure 3.5 – Comparison between GA_FS and SVM_FS with 2488 features**

When the degree of the kernel increases the accuracy of classification also increases comparatively with previously tested values for some types of data representation. But due to the huge vector dimension some of the data representation doesn't obtain such good results. Taking in

consideration the influence of data representation for polynomial kernel the best results were obtained all the time with the polynomial kernel and for a small degree of the kernel.

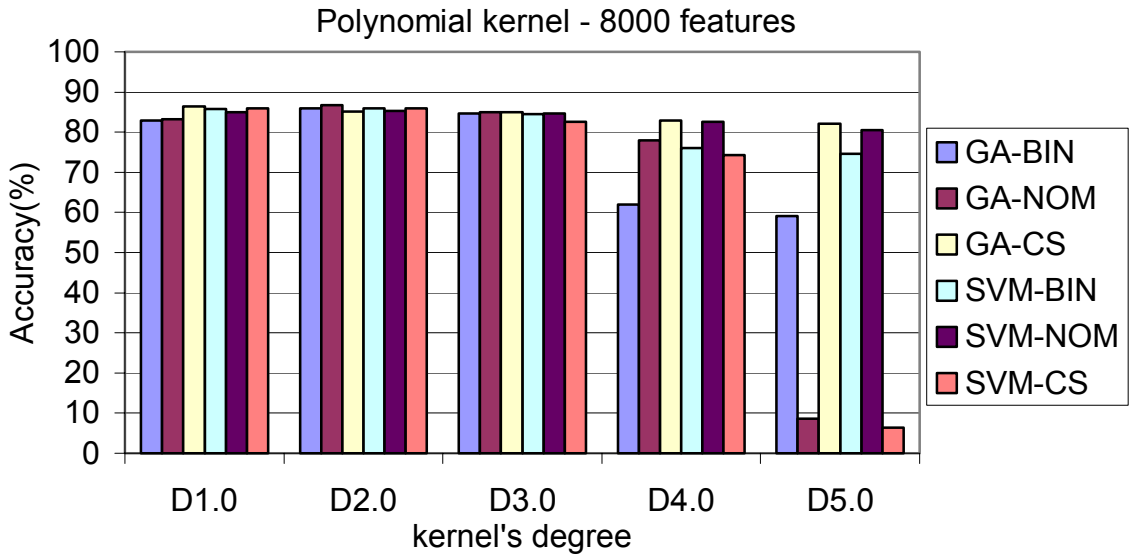


**Figure 3.6 – Comparison between GA_FS and SVM_FS for 8000 features**

As a conclusion for the polynomial kernel the GA_FS obtains better results comparatively with SVM_FS only when we work with a greater dimension of the feature vector. This can occur due to the fact that GA_FS has a starting point which is randomly implied. When the number of features increases the probability to choose better features increases too.

In Figure 3.7 the training times for polynomial kernel with degree 2 are presented. The training time for SVM_FS method increases from 11.52 minutes for 475 features to 14.56 minutes for 1306 features and to 46.55 minutes for 2488 features. Thus for fast learning we need a small number of features and as it can be observed from Figure 3.4, with SVM_FS method we can obtain better results with a small number of features Also the time needed for training using GA_FS is usually greater than the time needed for training with SVM_FS (for example, for 1309 features it takes 14.56 minutes for SVM_FS versus 26.42 minutes for IG and 18.14 minutes for GA_FS). When the number of features increases the training time for SVM_FS method exceeds the training time for GA_FS even though the results are better for GA_FS. The resulting times are given for a Pentium IV at 3.4 GHz, with 1GB DRAM and 512KB cache, and WinXP. In Figure 3.7 we also present the training time obtained with sets generated using Information Gain as feature selection method. This method was presented in the second PhD report. We present here these results in order to have a big picture for the learning time necessary for each selected set. As a conclusion, for the same dimension of the input vector, the sets obtained using GA_FS obtain results faster.

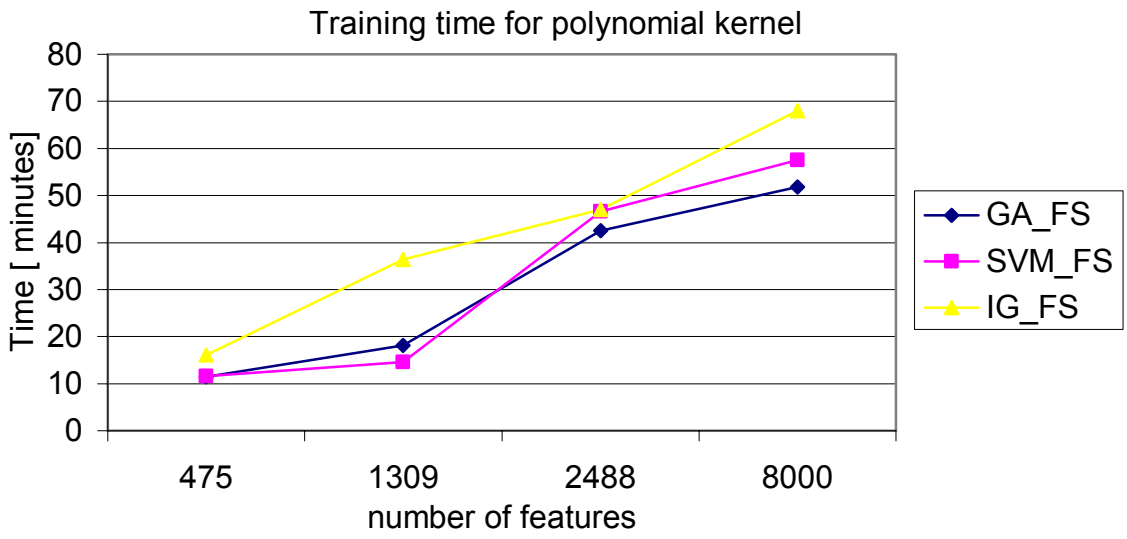


**Figure 3.7 – Training time for polynomial kernel with degree 2 and Nominal data representation**

In the next charts I present a comparison of results obtained for Gaussian kernel and different values of parameter *C* and only two types of data representation Binary and Cornell Smart.

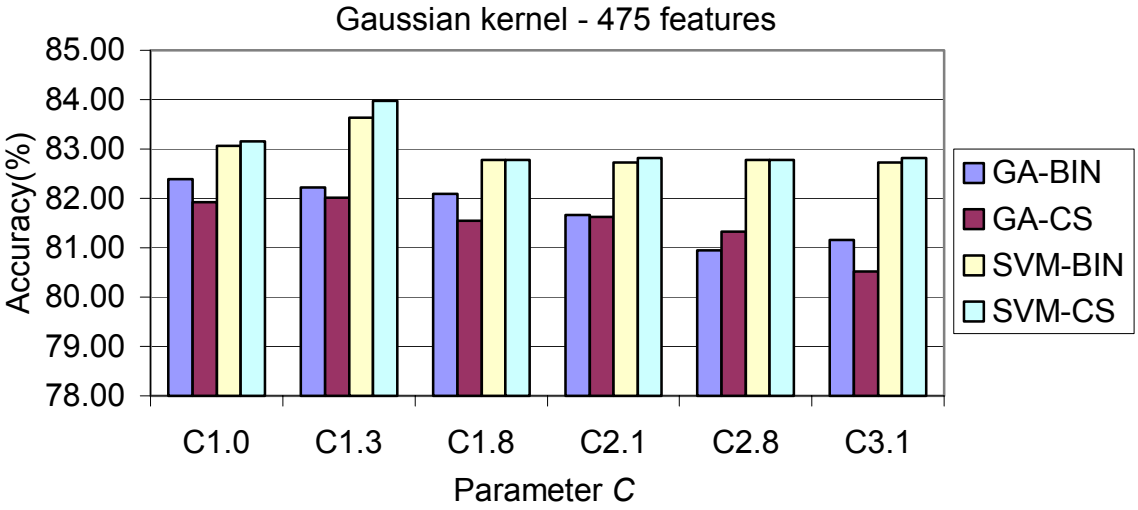


**Figure 3.8 – Comparison between GA_FS and SVM_FS for 475 features**

For a small dimension of the feature space (Figure 3.8) the SVM_FS obtains better results in all tested cases than GA_FS. Thus the maximum value 83.98% was obtained for SVM_FS with *C*=1.3 and Cornell Smart data representation in comparison with the maximum value obtained by GA_FS which was only to 82.39% obtained for *C*= 1.0 and Binary data representation.

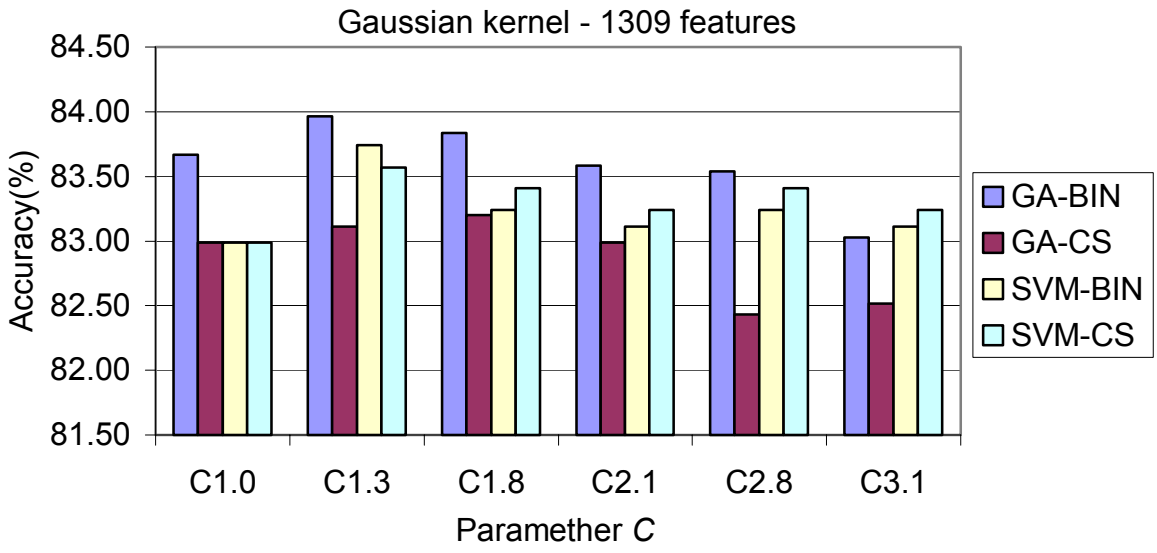


**Figure 3.9 – Comparison between GA_FS and SVM_FS for 1309 features**

When the number of features increases to 1309 (Figure 3.9) the GA_FS method obtains results with 0.22% better then SVM_FS. Thus GA_FS obtains result of 83.96% for *C*=1.3 and Binary data representation and SVM_FS obtains only 83.74% also for *C*=1.3 and Binary data representation. But as it can be observed from the chart for Binary data representation at all times the GA_FS obtains better results.

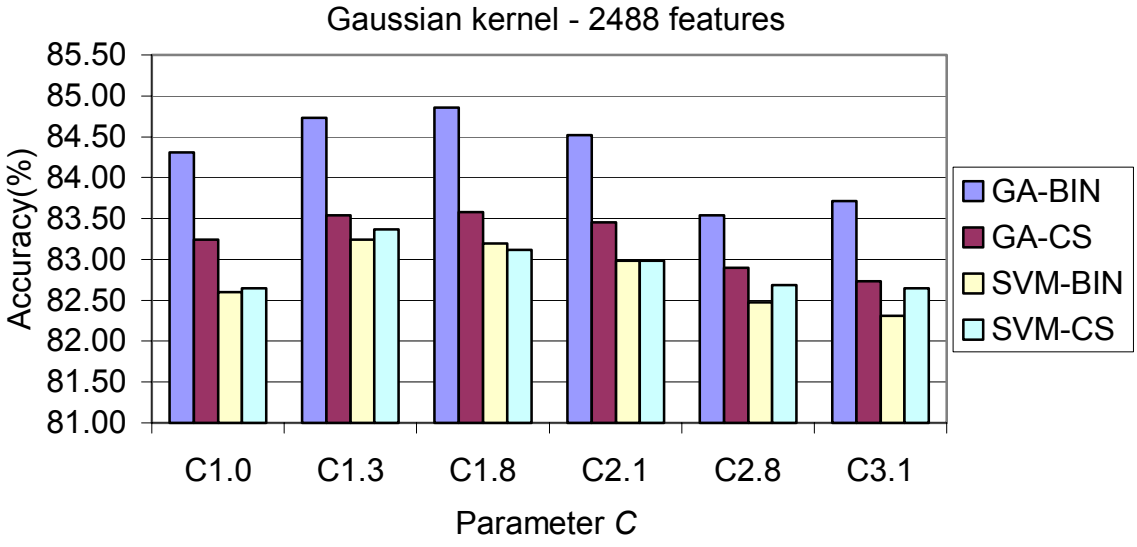


**Figure 3.10 – Comparison between GA_FS and SVM_FS for 2488 features**

In Figure 3.10 when the number of features increases more the discrepancy between the results obtained with Binary data representation and other representations increases. The maximum accuracy obtained also increases to 84.85% for GA_FS and *C*=1.8 Binary data representation. With SVM_FS the maximum value obtained is only 83.36% for *C*=1.3 and Cornell Smart data representation. The discrepancy has been more than 1% in accuracy.

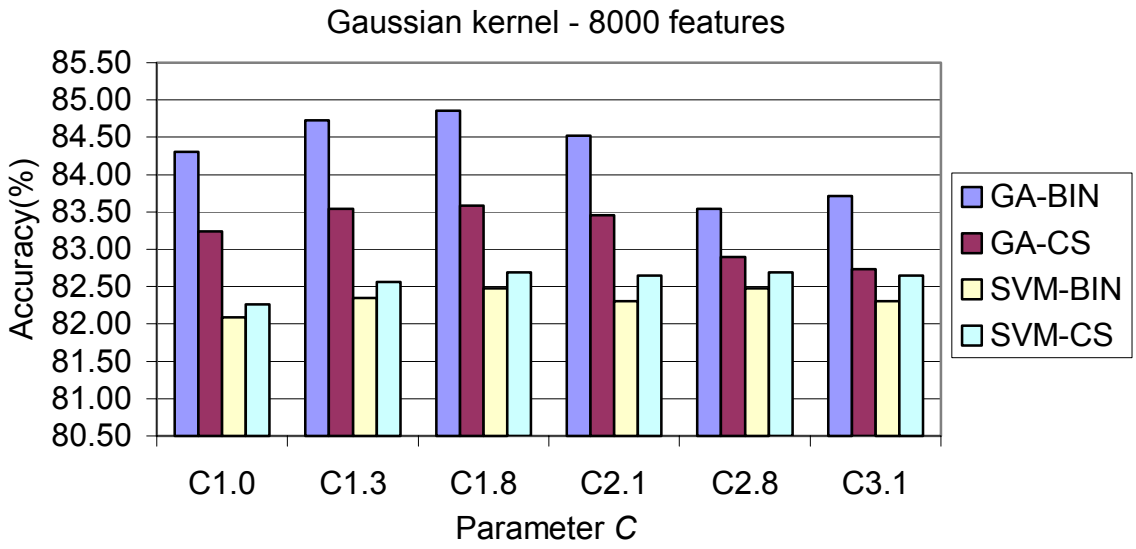


**Figure 3.11 – Comparison between GA_FS and SVM_FS for 8000 features**

When the number of features increases more the maximum accuracy obtained doesn't increase, see Figure 3.11. It remains 84.85% for GA_FS with the same characteristics as in the previous chart. For SVM_FS method when the number of features increases the accuracy of classification decreases to 82.68%. In general the GA_FS obtains better results for all the tested values.

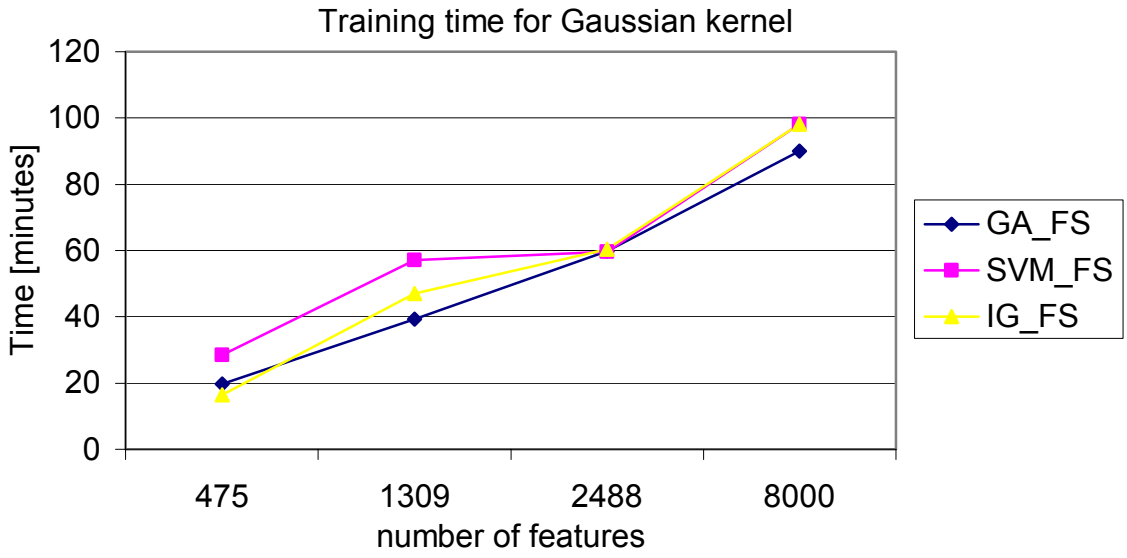


**Figure 3.12 – Training time for Gaussian kernel with *C* = 1.3 and binary data representation**

The training times for the Gaussian kernel are given for parameter *C*=1.3 and Binary data representation in Figure 3.12. We present here also the training time obtained with sets generated using IG_FS method presented in second PhD report. For this type o kernel SVM_FS method takes in all tested cases more time then GA_FS even if it doesn't obtain better results. Although, IG_FS method obtains for a small dimension the best learning time, when dimension increase, increase also the training time more than time needed with GA_FS method.

For Gaussian kernel results obtained with GA_FS are for all the tested dimensions are better in comparison with SVM_FS. Also this newly presented method obtains better results with a simplified form of data representation (Binary).

Comparing the two types of kernels tested the best results were obtained for the polynomial kernel with a small degree (86.68% for 1309 features and degree equal to 2 with SVM_FS), for Gaussian kernel we only obtained 84.84% for 2488 features and $C$=1.8 with GA_FS. In Table 3.1 we presented an average of classification accuracy over all results obtained for each feature set dimension and kernel type.

| **Kernel type** | **Nr. of features** | **GA_FS** | **SVM_FS** |
|---|---|---|---|
| Polynomial | 475 | 71.20% | 83.38% |
| | 1309 | 78.46% | 82.92% |
| | 2488 | 74.49% | 81.88% |
| | 8000 | 75.85% | 77.33% |
| Gaussian | 475 | 81.61% | 83.00% |
| | 1309 | 83.23% | 83.27% |
| | 2488 | 83.75% | 82.85% |
| | 8000 | 83.75% | 82.45% |

**Table 3.1 – Averages over all tests made for each feature dimension and each kernel type**

As it can be observed the GA_FS method obtains better results for the Gaussian kernel and for relatively high number of features. We can also see that for SVM_FS the optimal dimension of the feature space is a small one (about 4% of the total number of features). For the Gaussian kernel the differences between the results obtained by GA_FS and SVM_FS are not larger then 1.5%.

# 4 Meta-classifier with Support Vector Machine

Meta-learning focuses on predicting the right (classifier) algorithm for a particular problem based on characteristics of the dataset [Brz94]. One of the main problems when machine learning classifiers are employed in practice is to determine whether classification done for the new instances is reliable. The meta-classifier approach is one of the simplest approaches to this problem. Having more base classifiers, the approach is to learn a meta-classifier that predicts the correctness of each classification of the base classifiers. Meta labeling of an instance indicates the reliability of classification, if the instance is classified correctly by the base classifier from the used classifiers. The classification rule of the combined classifiers is that each based classifier assigns a class to the current instance and then the meta-classifier decides if the classification is reliable.

The method of combining multiple results taken from classifiers is not trivial and will determine the effectiveness of the whole system. There are two approaches to develop a meta-classifier. One of these is to append features from each classifier to make a longer feature vector and use it for the final decision. This approach will suffer from the "curse of dimensionality"[Lin02], as more features are included and the features vectors grow excessively. The other one is a usual approach, and it involves building individual classifiers and later combining their judgments to make the final decision. Anyway, meta-classification is effective only if it involves synergism.

In almost all studies those schemes for combining strategies can be considered ad hoc because they don't have any underlying theory. Selection based on the importance of each classifier is ignored or is arbitrary assigned. Those strategies are majority vote, linear combination, winner-take-all [Dim00], or Bagging and Adaboost [Siy01]. Also, some rather complex strategies have been suggested; for example in [Lin02] a meta-classification strategies using SVM [Lin02_2] is presented and compared with probability based strategies. In [Lin02] the authors present two interesting methods to combine the classifiers for a video classification. One of those strategies presented in [Kit98] is a framework based on probabilities and they propose a decision rule that computes a probability based on weights. Another strategy is based on Support Vector Machine technique that computes a decision function for each classifier based on the input vector and the classifier judgment. Those decision functions are used later to make the final decision.

## *4.1 Selecting Classifiers*

There are many different classification methods that are used for building base classifiers: decision tree, neural networks or naive Bayes networks [Siy01] and [Lin02] . Our meta-classifier is build using SVM classifiers. We do this because in our previous work we showed that some documents are correctly classified only by some certain type of SVM classifier. Thus, we put together many SVM classifiers with different parameters in order to improve the classification accuracy. Our strategies to develop the meta-classifier are based on the idea of selecting adequate classifiers for difficult documents. Our selected classifiers are different through: type of the kernel, kernels' parameters and type of the input data representation. We chose the input representation because, as we showed in [Mor06_2], this can have a great influence on the classification accuracy. Analyzing test results for all classifiers for the same training and testing data sets leads us to selecting 8 different combinations to build the classifiers. In the selection of the classifiers we were influenced

by the best obtained results and the type of input data correctly classified. Some results used for selecting the classifiers are presented in Table 4.1 for polynomial kernel and in Table 4.2 for Gaussian kernel. In those tables we present results obtained using only SVM_FS method that was showed to obtain the best results (with bold will be the best obtained results).

| No. of features | Data representation | P1.0 | P2.0 | P3.0 | P4.0 | P5.0 |
|---|---|---|---|---|---|---|
| 475 | BIN | 82.69 | 85.28 | 85.54 | 81.62 | 75.88 |
| | NOM | **86.64** | 86.52 | 85.62 | **85.79** | **85.50** |
| | SMART | 82.22 | 85.11 | 85.54 | 79.41 | 78.59 |
| 1309 | BIN | 81.45 | 86.64 | 85.79 | 74.61 | 72.22 |
| | NOM | 86.69 | 85.03 | 84.35 | 81.54 | 80.73 |
| | SMART | 80.99 | **87.11** | **86.51** | 71.84 | 8.34 |
| 2488 | BIN | 82.35 | 86.47 | 85.28 | 78.99 | 72.86 |
| | NOM | **86.30** | 85.75 | 84.56 | 81.79 | 81.16 |
| | SMART | 82.09 | 86.64 | 85.11 | 36.41 | 6.81 |
| 8000 | BIN | 20.71 | 85.96 | 84.43 | 76.05 | 74.61 |
| | NOM | 11.95 | 85.37 | 84.64 | 82.56 | 80.48 |
| | SMART | 20.93 | **86.01** | 82.60 | 74.22 | 6.42 |
| 18428 | BIN | 83.03 | 85.79 | 83.96 | 53.64 | 61.68 |
| | NOM | **86.22** | 85.50 | 84.94 | 82.99 | 81.54 |
| | SMART | 82.52 | 85.92 | 77.16 | 59.34 | 8.55 |

**Table 4.1 – Possible classifiers for Polynomial kernel**

| No. of features | Data representation | C1.0 | C1.3 | C1.8 | C2.1 | C2.8 |
|---|---|---|---|---|---|---|
| 475 | BIN | 83.07 | 83.63 | 82.77 | 82.73 | 82.71 |
| | SMART | **83.16** | **83.98** | 82.77 | 82.82 | 82.79 |
| 1309 | BIN | 82.99 | 83.74 | 83.24 | 83.11 | 83.01 |
| | SMART | 82.99 | 83.57 | **84.30** | **83.83** | **83.66** |
| 2488 | BIN | 82.18 | 83.11 | 82.94 | 82.86 | 82.80 |
| | SMART | 82.52 | 83.37 | 82.94 | 82.99 | 82.88 |
| 8000 | BIN | 82.09 | 82.35 | 82.48 | 82.31 | 82.11 |
| | SMART | 82.26 | 82.56 | 82.69 | 82.65 | 81.54 |

| | | | | | | |
|---|---|---|---|---|---|---|
| 18428 | BIN | 82.01 | 82.69 | 82.86 | 82.56 | 82.14 |
| | SMART | 81.75 | 82.39 | 82.60 | 82.43 | 81.92 |

**Table 4.2 – Possible classifiers for Gaussian kernel**

In the tables we presented results obtained for different dimensions of the input feature vectors. In our second PhD report and in section 3.2 of this report we presented some techniques of features selection. Also we showed that the best results are obtained for an optimal dimension of the feature vector (1309 features). Thus, we use here only the results using the feature vector having 1309 features selected using SVM_FS method that obtains better results with polynomial kernel [Mor06_2] and comparable results with GA_SVM for Gaussian kernel [Mor06_3]. In Table 4.3 we present the selected classifiers, each of them with the specified selected parameters for polynomial and Gaussian kernels.

| Nr. Crt. | Kernel type | Kernel parameter | Data representation | Obtained accuracy (%) |
|---|---|---|---|---|
| 1 | Polynomial | 1 | Nominal | 86.69 |
| 2 | Polynomial | 2 | Binary | 86.64 |
| 3 | Polynomial | 2 | Cornell Smart | 87.11 |
| 4 | Polynomial | 3 | Cornell Smart | 86.51 |
| 5 | Gaussian | 1.8 | Cornell Smart | 84.30 |
| 6 | Gaussian | 2.1 | Cornell Smart | 83.83 |
| 7 | Gaussian | 2.8 | Cornell Smart | 83.66 |
| 8 | Gaussian | 3.0 | Cornell Smart | 83.41 |

**Table 4.3 – Selected classifiers**

## *4.2 The Limitations of the developed Meta-Classifier System*

Another interesting question that occurs when we choose embedded classifiers is: Where is the upper limit of our meta-classifier? With others words, we want to know if there are some input documents for which all selected classifiers assign them to an incorrect class. However, we select classifiers following the idea of having a small number of incorrectly classified documents. We remind that in all comparisons we take as a reference the Reuters' classification.

In order to do this we take all selected classifiers and we count the documents that are incorrectly classified by all classifiers. The documents are from the testing sets because we are interested here if there are documents with problems into this set. We found 136 documents from 2351 that are incorrectly classified by all the base classifiers. Thus the maximum accuracy limit of our meta-classifier is 94.21%. Obviously we will select other classifiers to develop the meta-classifier it would be obtain another upper limit.

## *4.3 Meta-classifier Models*

The main idea that we had when we designed our meta-classifiers was that classifiers should have implemented a simple and faster idea in order to give the response. Also we were interested in

implementing in our meta-classifier the idea of selecting the adequate classifier for a given input vector. In order to design the meta-classifier we are using three models. First of them is a simple approach based on the voting principle, thus without any adaptation. The other two approaches are implementing adaptive methods.

### 4.3.1 A non-adaptive method: Majority Vote

The first model of meta-classifier was tested just due to its simplicity. It is a maladjusted model that obtains the same results in time. The idea is to use all the selected classifiers to classify the current document. Each classifier votes a specific class for a current document. The meta-classifier will keep for each class a counter; increment the counter of that class when a classifier votes for it. The meta-classifier will select the class with the greatest count. If we obtain two or more classes with the same value of the counter we classify the current document in all proposed classes. The great disadvantage of this meta-classifier is that it doesn't modify the evolution with the input data in order to improve the classification accuracy, in other words, it is non-adaptive (static). The percentage of documents correctly classified with this meta-classifier is 86.38%. This result is with 0.73% smaller that the maxim value obtained by one of the selected classifiers, but is greater than their average accuracy (85.26%).

### 4.3.2 Adaptive developed methods

#### 4.3.2.1 Selection based on Euclidean distance (SBED)

Because that previous presented meta-classifier doesn't obtain such good results we develop a meta-classifier that changes the behavior depending on the input data, adaptive. To do this, we build a meta-classifier that selects a classifier based on the current input data. To do this we make our meta-classifier learn the input data. We are expecting that the number of correctly classified samples will be greater than the number of incorrectly classified input samples. So that our meta-classifier will learn only the input samples incorrectly classified. Our meta-classifier will learn only data that is incorrectly classified by the selected classifier. Thus the meta-classifier will contain for each classifier a self queue where are stored all incorrectly classified documents. Therefore, our meta-classifier contains 8 queues attached to the component classifiers.

For this approach of the meta-classifier we implemented two different versions of choosing the classifier that will be used to classify the current sample. The first version is faster but it tries to find the first classifier that is reliable to be used in the current classification so that it doesn't offer the highest performance possible. Another version is to find all the time the best classifier that can be used for the current classification. This method takes more time for choosing the classifier. The difference in accuracies is insignificant so we prefer the faster method.

##### *4.3.2.1.1 First classifier Selection Based on Euclidian Distance (FC-SBED)*

Considering an input document (current sample) that needs to be classified, first we randomly chose one classifier. We will compute the Euclidean distance (equation 4.1) between the current sample and all samples that are in that self queue of the selected classifier. If we obtain at least one distance smaller than a predefined threshold we renounce to use that selected classifier. In this case we will randomly select another classifier, excepting the already rejected one. If there are cases when all component classifiers are rejected, however, we will choose that classifier with the greatest Euclidean distance.

$$Eucl(\mathbf{x},\mathbf{x}') = \sqrt{\sum_{i=1}^{n}([x]_i - [x']_i)^2} \quad (4.1)$$

, where $[x]_i$ represents the value from the entry *i* of the vector **x**, and **x** and **x'** represent the input vectors.

After selecting the classifier we will use it to classify the current sample. If that selected classifier succeeds to correctly classify the current document, nothing is done. Otherwise we will put the current document into the selected classifier is queue. We do this because we want to prevent that this classifier classify further this kind of documents. To see if the document is correctly or incorrectly classified we compare the proposed class with Reuters proposed class.

The complete scheme of evolution for this meta-classifier FC-SBED is presented in Figure 4.1. One document is written into the queue of misclassified documents only when the selected classifier proposes a different result than the result proposed by Reuters.

This meta-classifier has two steps. All presented actions are made in our meta-classifier into the first step called the learning step. In this step the meta-classifier analyzes the training set and each time when a document is misclassified it is put in the selected classifier queue. In the second step, called the testing step, we test the classification process. In the testing step the characteristics of the meta-classifier remain unchanged. Because after each training part the characteristics of the meta-classifier might be changed, we repeat these two steps many times.

#### *4.3.2.1.2 Best classifier Selection Based on Euclidian Distance (BC-SBED)*

This method follows the method FC-SBED presented in Section 4.3.2.1 with one change. This is that the current tested classifier is not randomly selected. In contrast, we take into the consideration all classifiers. We'll compute the Euclidean distance between the current sample and all misclassified samples that are into the queues. We will choose the classifier that obtains the maximum distance. In comparison with the previous method this method is slower. The complete scheme of evolution for this meta-classifier (BC-SBED) is presented in Figure 4.2.

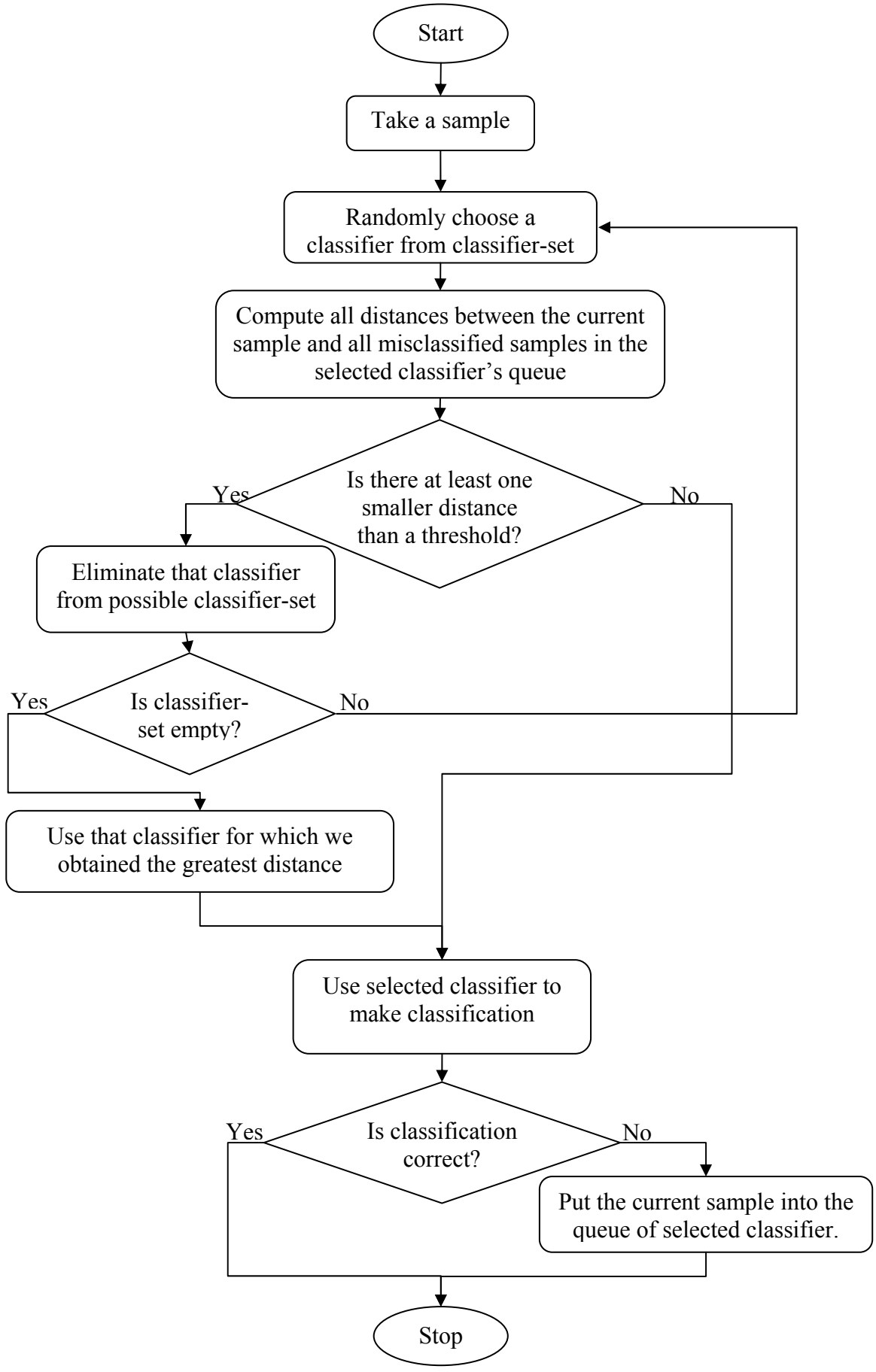


**Figure 4.1 – Meta-classifier diagram – FC-SBED**

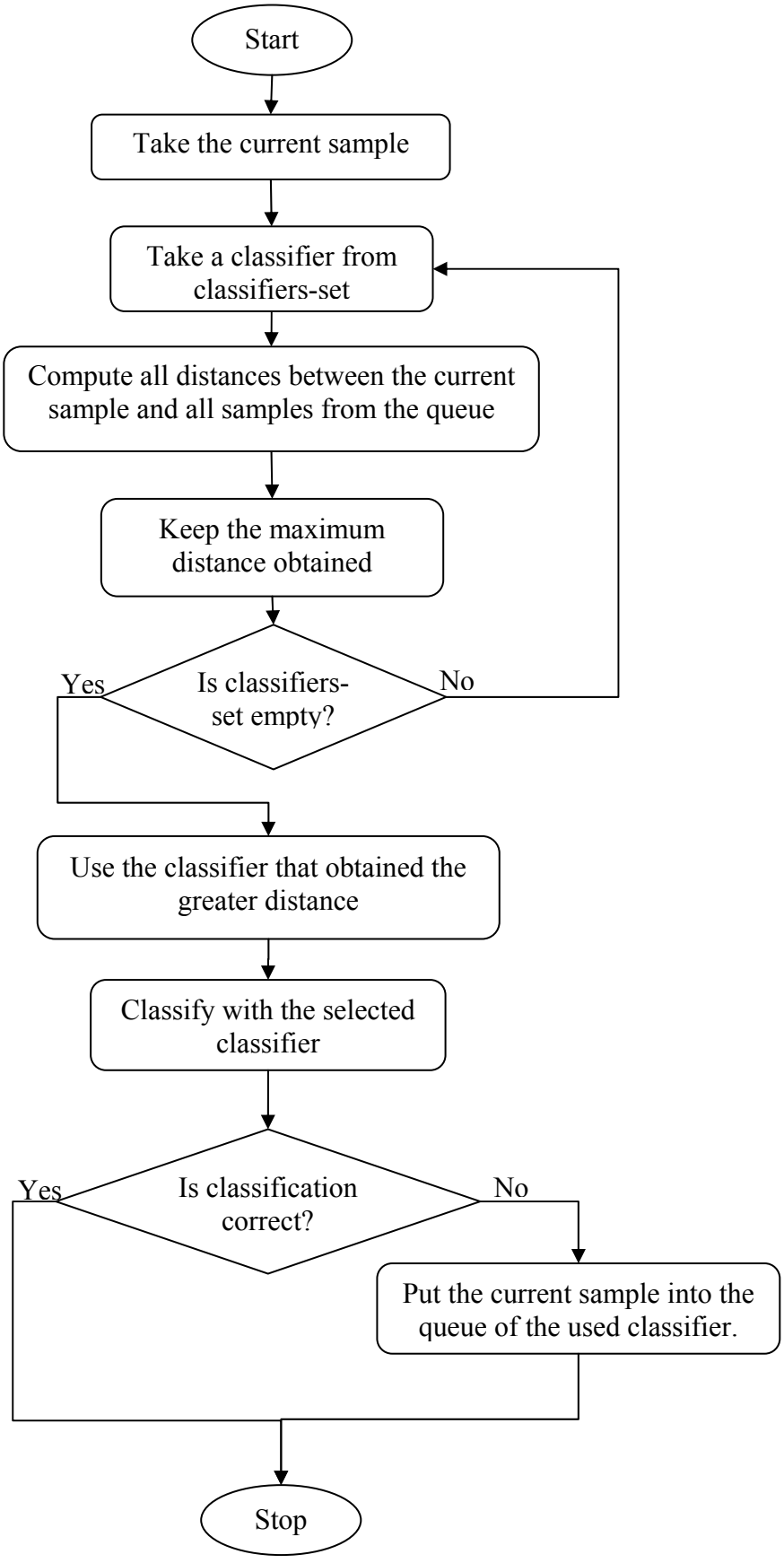


**Figure 4.2 - Meta-classifier diagram – BC-SBED**

#### 4.3.2.2 Selection based on cosine (SBCOS)

The cosine is another possibility to compute the document similarity, usually used into the literature when we work with vectors that characterize documents. This is based on computing the dot product between two vectors. The used formula to compute the cosine angle $\theta$ between two input vectors **x** and **x'** is:

$$\cos\theta = \frac{\langle \mathbf{x}, \mathbf{x}' \rangle}{\|\mathbf{x}\| \cdot \|\mathbf{x}'\|} = \frac{\sum_{i=1}^{n} [x]_i [x']_i}{\sqrt{\sum_{i=1}^{n} [x]_i^2} \cdot \sqrt{\sum_{i=1}^{n} [x']_i^2}} \quad (4.2)$$

, where **x** and **x'** are the input vectors (documents) and $[x]_i$ represents the vector's $i^{th}$ component. The domain of this formula is between -1 and 1. The value 1 is obtained when the input vectors are similar. The value 0 represent that the input vectors are orthogonal and value -1 is obtained when the input vectors are dissimilar. Since our vectors contain only positive components the cosine is between 0 and 1 which yields to values of $\theta \in \left[2k\pi - \frac{\pi}{2}, 2k\pi + \frac{\pi}{2}\right], \quad (\forall)k \in Z$. In our particular case $\theta \in \left[0, \frac{\pi}{2}\right]$ since we make the reduction to the first quarter.

This method follows the method SBED with modifications in computing the similarity between vectors. Also for this method to compute the similarity between documents we implement two methods for selecting the current classifier as for SBED called FC-SBCOS and BC-SBCOS. In those methods we consider that the current selected classifier is acceptable if all computed cosines between the current sample and all samples that are into the queue are smaller than a threshold. We will reject them if at least one cosine angle is greater then a threshold.

We can have the following situation: two documents are very close to one another as far as the angle is concerned but they are at a very large distance which can actually mean that they are not similar at all (they can belong to different classes). This situation can be very common considering our vectors and can lead to a large number of misclassifications.

#### 4.3.2.3 Selection based on dot product with average

In all presented methods we kept in the queue of each classifier the vector of documents that was incorrectly classified by that classifier. As an alternative to this, we also tried to reduce the queues' dimension by keeping into them only the average over all vectors that are needed to be kept. More precisely, in each queue we kept only a single vector representing the partial sum of all the error vectors, and a number that represent all vectors that were kept. This makes the algorithm faster but unfortunately the results are not so good.

## *4.4 Experimental Meta-classifier results*

In [Mor05] and [Mor06_2] we showed that the best results are obtained using a dimension of the feature space around 1309. So that for the meta-classifier we will present results obtained using

only this feature dimension. As feature selection we used a method based on support vector machine technique with linear kernel (SVM_FS). This method was detailed in [Mor06_2].

In all results that we will present we take as a reference the Reuters' classification that was considered to be perfect. Also all results are presented for multi-class classification, taking into consideration all 24 selected classes.

First we do a short comparison between the influence of selecting the classifier, first good classifier or best classifier, so for similarity we computed both Euclidean distance and cosine angle.

### 4.4.1 Results for Selection based on Euclidean distance

As we already mentioned the two presented methods based on Euclidean distance, "First classifier - selection based on Euclidean distance" (FC-SBED) and "Best classifier – selection based on Euclidean distance" (BC-SBED), request some steps for training. We do 14 learning steps with different values for the threshold. After each learning step we make a testing step. We stop after 14 learning steps because we noticed that after this value the accuracy doesn't increase but it sometime even decreases. In Figure 4.3 we present results for each step as a percentage of correctly classified documents.

In order to have a good view in all following charts, we also present the maximum limit for our meta-classifier (upper limit). The upper limit (94.21%) has been multiplied for each test because it remains the same.

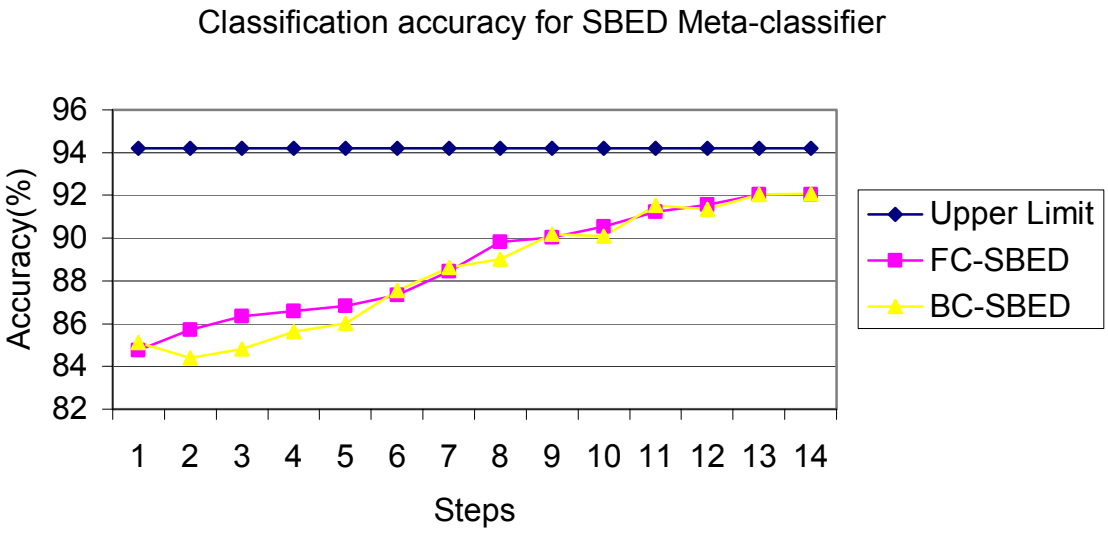


**Figure 4.3 - Evolution of classification accuracy for SBED**

When using selection based on Euclidean distance the threshold was chosen during the first 7 steps equal to 2.5 and during the last 7 steps equal to 1.5. First time we selected a greater threshold value in order to reject more possible classifiers. When in the queue there is already an error sample we will reject that classifier easier. We make this because in the first steps we are interested in populating all queues from our meta-classifier. In the last 7 steps we decrement the threshold to

make the rejection more difficult. Those two thresholds are chosen after laborious experiments with different threshold values that are not presented here.

At the beginning of the training, when there aren't any documents in the queues, the classification accuracy is not so good (84.77%). But, as it can be observed, after each step the accuracy improves growing up to 92.04% in 13$^{th}$ step. Considering the ratio between our maximum obtained value and the upper limit we see that our learning reaches 97.69% of its potential. Thus the results obtained after the 13$^{th}$ step can be considered as good results.

In the Figure 4.4 we present the response time for methods based on Euclidian distance in order to compute the similarity. Also, for a good view we present the response time for Majority Vote, the method that takes the longest time.

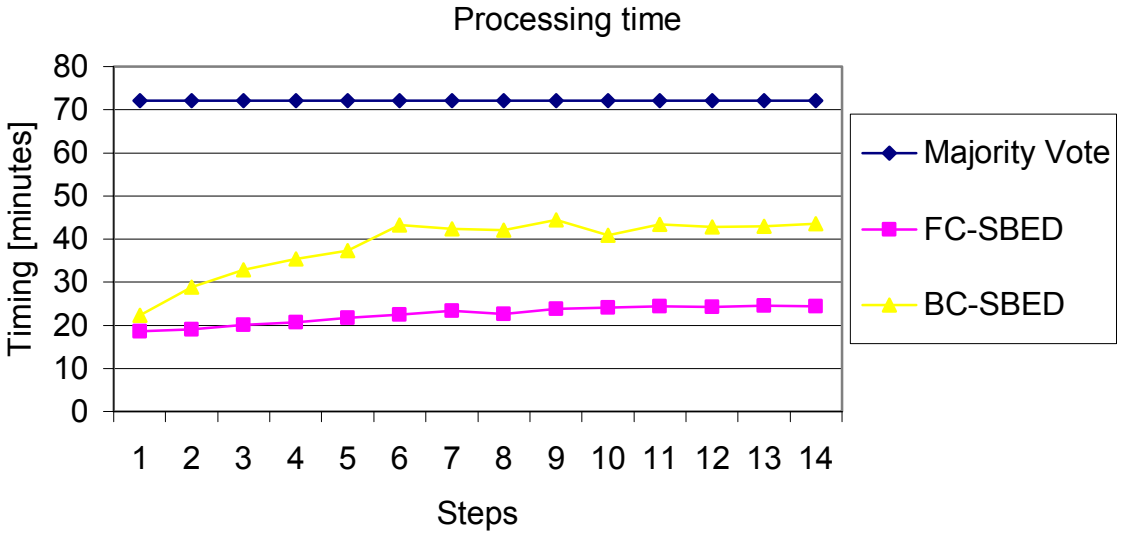


**Figure 4.4 – Processing time for SBED**

As we expected it takes less time for the FC-SBED then for the BC_SBED, but in comparison with the accuracy of classification the time difference is not justified.

### 4.4.2 Results for Selection based on cosine angle

For methods that use the cosine to compute the similarity, "First classifier - selection based on cosine" (FC-SBCOS) and "Best classifier – selection based on cosine" (BC-SBCOS), there are also necessary some steps for training. In order to make later a comparison between those methods (SBED and SBCOS) we also do 14 learning steps with different values for the threshold. We have also noticed that we obtained good results for the accuracy after the 14 steps similar to the SBED. After each learning step we make a testing step. In Figure 4.5 we present results for each step as a percent of correctly classified documents.

In order to have a good view, we will also present the maximum limit for our meta-classifier. In this chart we also put the results obtained with average error vectors stored into the queue presented in section 4.3.2.3.

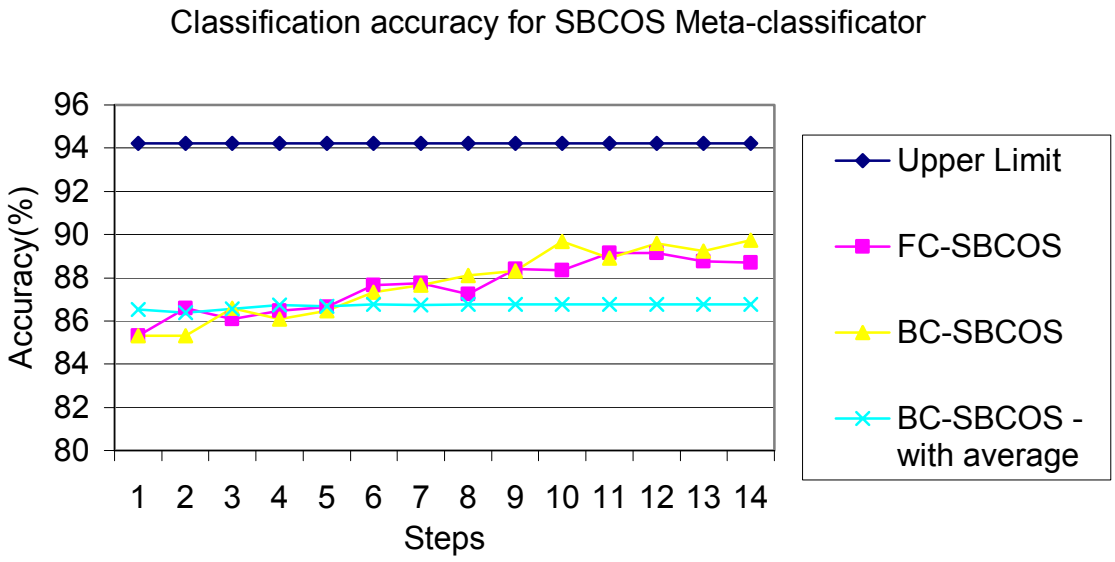


**Figure 4.5 – Evolution of classification accuracy for SBCOS**

In the case of selection based on cosine the threshold was chosen during the first 7 steps equal to 0.8 and during the last 7 steps equal to 0.9. We are also interested to be able to easily reject a classifier in the first 7 steps. This is way we chose a value much smaller than 1, which means much less similar documents. In the last 7 steps we used a value closer to the value that means similar documents. This assures that in the first steps we populate the queues and in the last steps we actually calibrate our meta-classifier for documents which are more difficult to classify. We made also experiments with other values of the threshold between 0.25 and 1 and here we present only the best obtained results.

In comparison with SBED, this method has a better starting point (85.33%). After 14 steps the accuracy increases only to 89.75%. Considering the ratio between our maximum obtained value and the upper limit we see that our learning reaches 95.26% of its potential. The results obtained with the average do not improve so much the evolution of our meta-classifier. The accuracy improves from 86.38% to 86.77% in the last steps. This maximum value being greater than the value obtained with Majority Vote with only 0.39%. Also, as it can be observed the difference between the first good classifier and best classifiers methods are not so great, usually they obtain the same results; sometimes one of them obtains better results sometimes the other. At the end, in the last step, the BC-SBCOS method obtains a result with 1.06% greater than the other method. Considering the learning time for these two methods the times are considerably different. In Figure 4.6 we present all response times needed to compute these results. Thus, for each of them we put the time needed for training and testing parts together because we think that our meta-classifier can learn continually. All values are presented in minutes, because some of the methods are slow (more than one hour as majority vote). The numbers are given for a Pentium IV at 3.4 GHz, with 1 GB memory, 10 GB HDD a (7200 rpm) and Windows XP.

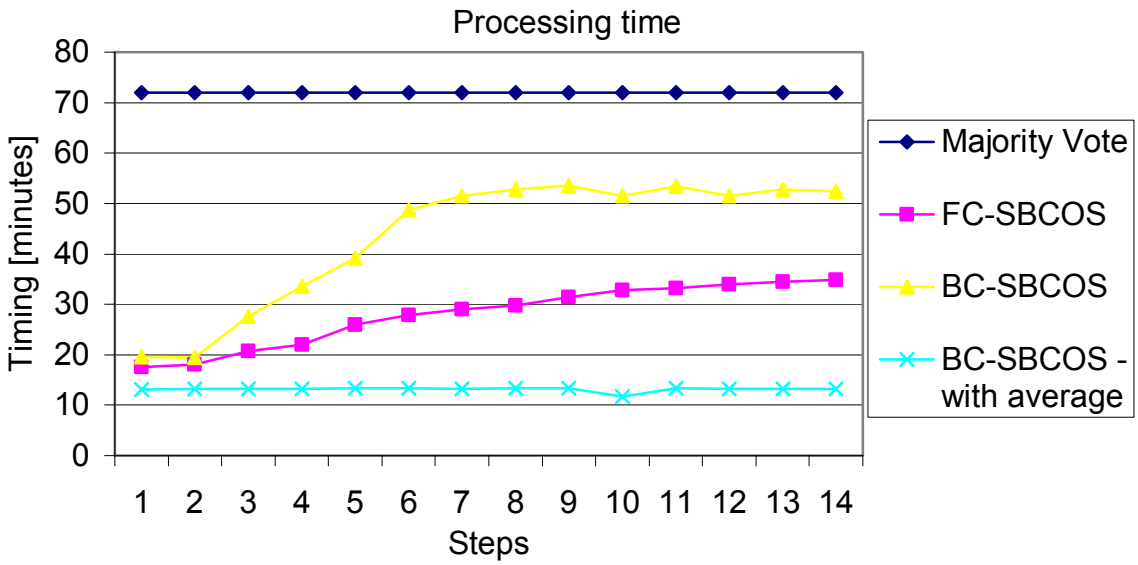


**Figure 4.6 – Processing time for SBCOS**

We also present here the time needed for Majority Vote method because we want to have a good view. As we expected Majority Vote method takes the longest time because it waits after results from all classifiers from the meta-classifiers. Also BC-SBCOS takes more time because it needs to compute all values between all the error elements that are in the queues. This computation takes lees than Majority Vote but it also takes a long time. The fastest method is BC-SBCOS with average but as it can be observed form Figure 4.5 the accuracy of results is not so good. FC-SBCOS takes more time then BC-SBCOS with average but the accuracy increases significantly after 14 steps in comparison with the other method.

In Figure 4.7 we present comparative results between all presented methods used to build the meta-classifier. From SBED we selected results obtained with FC-SBED and for SBCOS we selected results obtained with BC-SBCOS. Also those results were presented in[Mor06_4].

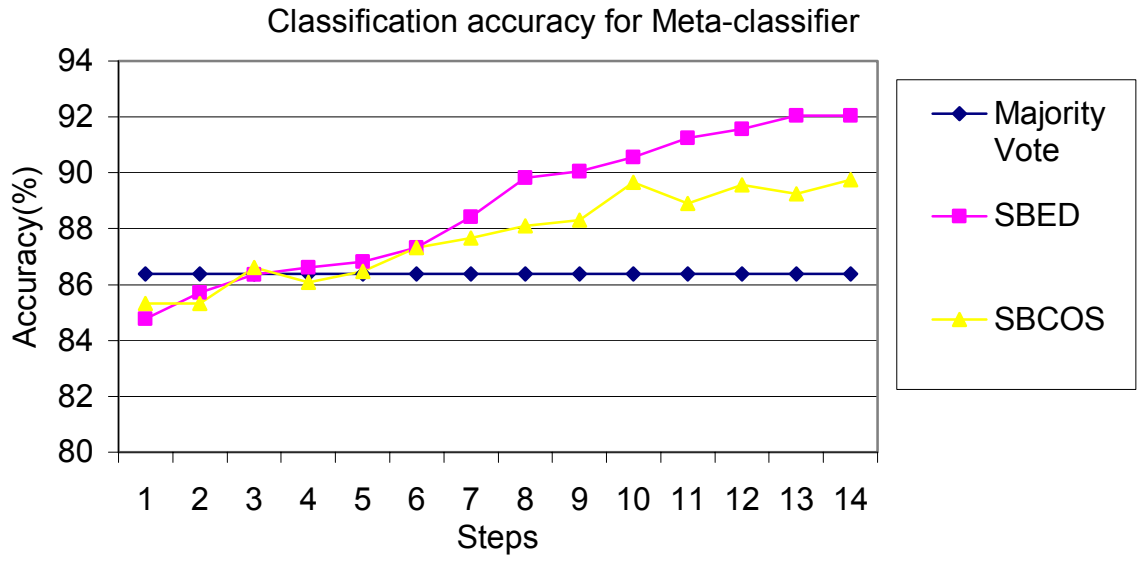


**Figure 4.7 – Classification Accuracy for our meta-classifier**

With Majority Vote the accuracy of classification that was obtained with this meta-classifier is 86.38%. This result is with 0.73% smaller than the maximum individual value of one classifier but it is greater than the average over all classifiers. Comparing SBED with SBCOS methods, the second method has a better starting point (85.33%). After 14 steps the accuracy increases only to 89.75% in comparison to SBED that obtains a final accuracy of 92.04%.

In Figure 4.8 we present all response times obtained for the methods presented above. For Majority Vote we put more times the same response time value. For the other methods we present in this figure the average of the response times obtained for the method based on best classifier selection and the method based on first classifier selection for each of the two methods to compute the similarity.

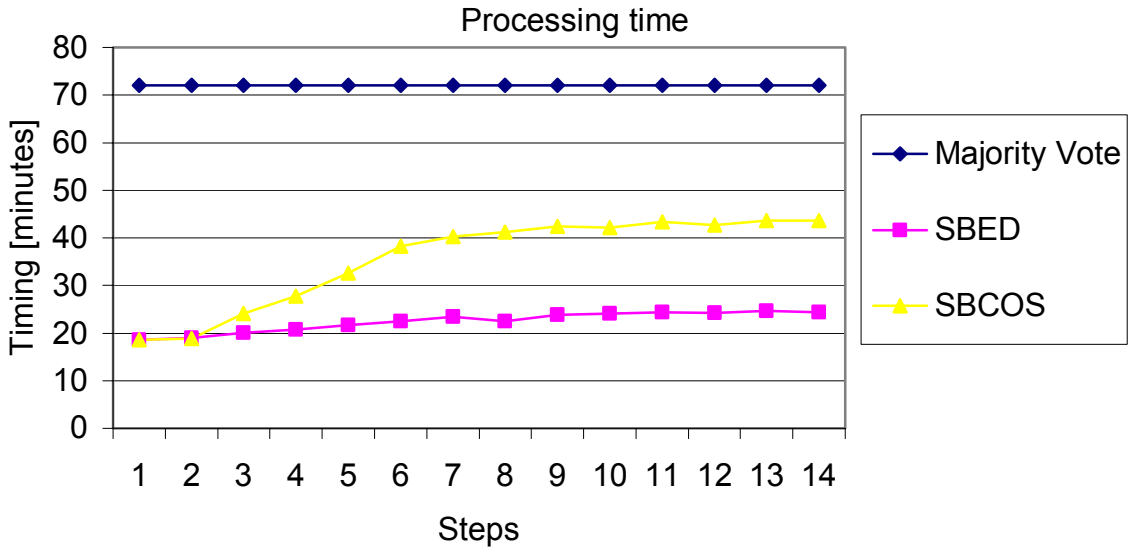


**Figure 4.8 – Processing time – comparison between SBED and SBCOS**

As a comparison we can see that the fastest method is the one that uses Euclidean distance to compute the similarity. Sometimes we obtained a difference by up to 20 minutes during the last steps. Comparing the last two figures we can see that the SBED is faster and obtains better results than SBCOS. The Majority Vote obtains powerless results with the greatest computation cost, as we expected.

# 5 Initial data set scalability

Up to this moment I presented a number of techniques for text classification and laborious comments on their features, strengths, and weaknesses. Given a typical IR system based on vector-space similarity, it is easy to build a classifier that simply indexes all the training set documents, remembering their class labels. A test document is submitted as a query to the IR system, and the distribution of labels on the training documents most similar to it are used to make a decision. The vector-space model assigns large weights to rare terms, without regard to the frequency with which terms occur across documents from different classes.

The process of feature selection removes terms in the training documents that are statistically uncorrelated with the class labels [Gun03], leaving behind a reduced subset of terms to be used for classification. Feature selection can improve both speed and accuracy.

There are several criteria to evaluate classification systems according to [Cha03]

1. Accuracy, the ability to predict the correct class labels most of the time. This is based on comparing the classifier-assigned labels with human-assigned labels.
2. Speed and scalability for training and applying/testing in batch mode.
3. Simplicity, speed, and scalability for document insertion, deletion, and modification, as well as moving large sets of documents from one class to another.
4. Ease of diagnosis, interpretation of results, and adding human judgment and feedback to improve the classifier.

Ideally, we would like to compare classifiers regarding all of these criteria, but simplicity and ease of use are subjective factors, and speed and scalability change with evolving hardware. Up to this moment we focused on the issue of accuracy and speed, with some comments on performance where appropriate. Now, in this chapter we want to focus on scalability for training documents and applying results.

In the last years the available text data becomes larger and larger and a lot of algorithm was proposed to work with them [Kan02] [Ord03]. It is quite straightforward to transform the algorithms so that they are able to deal with larger data [Bei02] (i. e., the scalability of the algorithms). Scalability has always been a major concern for IR algorithms [Ber02].

In this chapter we want to see if there are some huge influences when our algorithm works with more input vectors. In order to do this I developed a strategy in three steps that allows us to work with a greater dimension of the training set, reducing the learning time needed if we work with the entire set at a time. We focused on the training part when the quantity of presented data for learning has a great influence on the capability of the learning system to make good classifications. In designing this strategy we were inspired by a strategy presented in [Yu03] which uses a tree structure to group similar data in databases on levels. This strategy though was not recommended by the authors to be used on text documents and thus we modified it in to a single level grouping.

For having comparable results on classification accuracies for this new strategy of scalability we use as training and testing set the same Reuters set of 7083 samples. This set was generated as the previously used set with samples having a total of 19038 features and 24 topics.

The original learning step that uses the Support Vector Machine technique to classify documents is split in three steps. In the first step all training data is grouped based on its similarity, using "winner-take-all" method [Hun03]. The number of groups that can be created in this step is unlimited and depends on the similarity threshold and data dimension. From each created group a representative vector is computed. With those vectors we create a new training dataset that will be used in the next step of classification. This step had been the representative vector classification step. After the classification, besides the classification results we also obtain the elements that have effective contribution to the classification (called support vectors). This idea can be applied because we use as a classification method the Support Vector Machine techniques. Taking only the selected representative vectors (support vectors) we make a new reduced learning dataset. In the last step, the classification step, we will use a reduced dimension of the training set made only from a small part of input vectors that can have a real importance in finding the decision function.

A group is represented as a vector, which is computed as an arithmetic average over all elements that are included in that group, and a value that represents the total number of samples from that group. In the first step for each new sample we compute the similarity between this sample and all the representative vectors. If there is at least a similarity smaller that the predefined threshold we will put the current sample in that group for which we obtain the smallest value and recompute the representative vector.

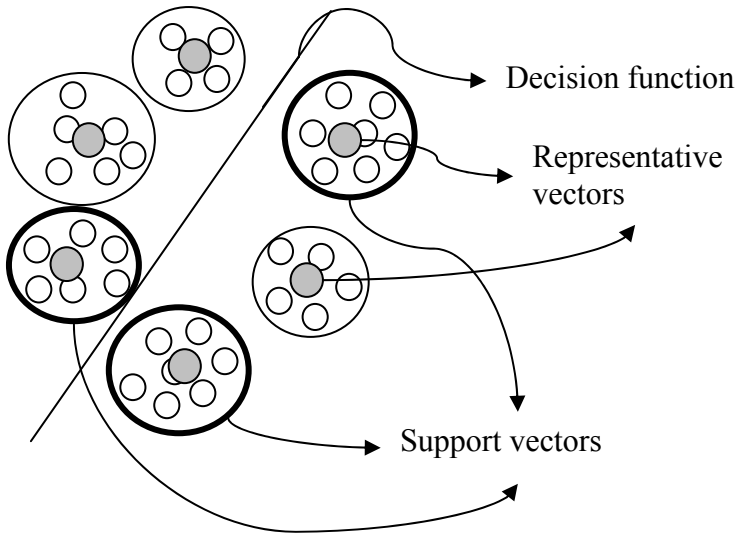


**Figure 5.1 – Selecting of support vectors**

The entire process is presented in the next 7 steps:

1. We normalize each input vector in order to have the sum of elements equal to 1, using the following formula:

$$TF(d,t) = \frac{n(d,t)}{\sum_{\tau=0}^{19038} n(d,\tau)} \tag{5.1}$$

where *TF(d,t)* is the term frequency, *n(d,t)* is the number of times that term *t* occurs in document *d*, and the denominator represent the sum of terms that occur in the entire document *d*.

2. After normalization we compute the Euclidian distance between each input vector and each representative vector (Figure 5.1 the gray small circles) that was created up to this moment. This makes our strategy slower when we have more groups. The formula for Euclidian distance is presented in equation 4.1. We prefer to choose for similarity the Euclidean distance method because as we showed in section 4.3.2.2 the dot product method, usually used in the literature, sometimes can lead to bad results.

$$E(t, c_i) = \sqrt{\sum_{k=0}^{19038} (t_k - c_{ik})^2} \tag{5.2}$$

Where $t_i$ represent the terms from the input vectors and $c_i$ represent term from the representative vector. Thus, we compute each distance and we keep the smallest obtained distance. If that obtained distance is smaller than a predefined threshold we will introduce the current sample in the winner group, if not we will make a new group.

3. After this grouping (all large circles from Figure 5.1), we will make a new training dataset with all representative vectors. This set will be used in the first classification step. In this step we are not interested in the accuracy of classification because the input vectors are not the original vectors. Here we are interested only in selecting only relevant vectors from this new set. Because in this step we use a classification method that needs a topic for each created group we need to specify a topic. This topic is specified automatically as being the most frequent topic that occurs in that group.
4. On this reduced set, having 19038 features, we make a feature selection step. For this step we prefer to use SFM_FS method presented in our second PhD report. After computing all weights we select only 1309 features because as we showed they offer good results.
5. The resulted smaller vectors are used in a learning step. For this step I use polynomial kernel with degree equal to 1 and nominal data representation. I use polynomial kernel because it usually obtains a small number of support vectors in comparison with Gaussian kernel. I use the degree of the kernel equal to 1 because in almost all cases I obtained better results in the previous testes.
6. After SVM learning step I chose only those vectors that are support vectors (have parameter α greater that 0 – Lagrange multipliers in Figure 5.1 the large circles with a thick line). I chose all groups that are represented by those selected vectors. With these groups we make a new training set that contains the original input vectors that were included in the grouping step in the selected classes (step 2).
7. This set will now be used in the feature selection and classification steps as the original step but having a smaller dimension and containing only elements that can really contribute at the decision making.

In our presented results we start with an initial set of 7083 vectors. After the grouping step we reduce this dimension at 4474 representative vectors that means 63% from initial set. For this reduction we use a threshold equal with 0.2. Following, we take the feature selection step and after that a classification step for selecting the relevant vectors. After the classification the algorithm returns a number of 874 support vectors. Taking those relevant vectors we create a dataset that contains only 4256 samples that means approximately 60% of the initial data. We make a feature selection step using SVM_FS method and select only 1309 features. This set was splat in the training set having 2555 samples and in the testing set having 1701 samples. We make only one test because we were interested to see if the accuracy will go down excessively when we apply a method to select only a small part from the entire set.

In the Figure 5.2 we present comparative results obtained for Polynomial kernel and nominal data representation.

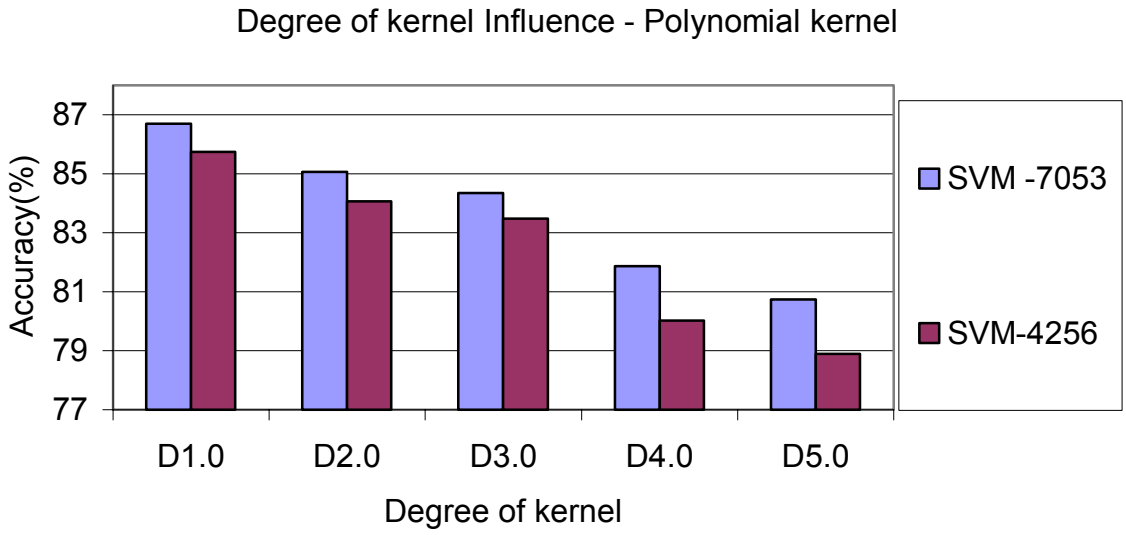


**Figure 5.2 – Comparative results for different set dimensions – polynomial kernel**

As it can be observed the difference in the accuracy obtained is on average equal to1.3% for all kernel degrees. When we work with a small dimension of the kernel degree the difference is smaller than 1. For example the difference for degree 1 is 0.94%. When the degree of the kernel increase the difference between these two sets increases to 1.87%. In Figure 5.3 are presented results obtained for Gaussian kernel and Cornell Smart data representation. For this kernel the average difference between those two sets is greater than in polynomial kernel case, being of 1.89%. The smallest difference was obtained of a value C equal with 1.8, value for which in all previous tests we obtained the best results.

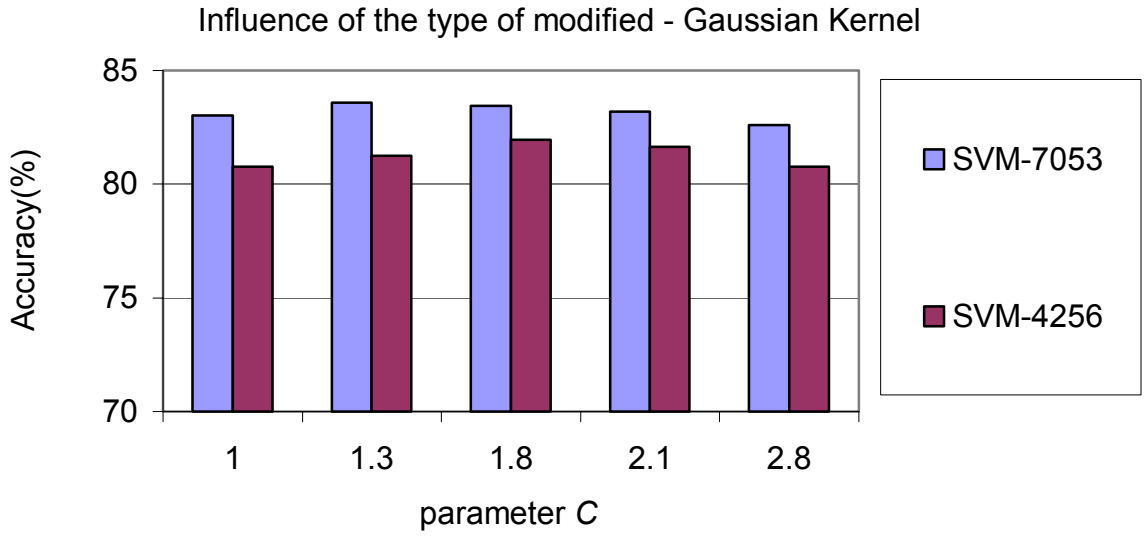


**Figure 5.3 – Comparative results for different set dimensions – Gaussian kernel**

As we expected the accuracy of classification decreases but not so much. We reduced the data in the first step at 63% from in initial set and in the second step at 60% from the initial set and the lose in accuracy was about 1% for polynomial kernel and about 1.8% for Gaussian kernel. It is an interesting observation that for values of parameters (degree or *C*) for which we usually obtain the best results with the normal set the difference when work with the reduced set is the smallest.

Obviously, the time needed for training on small number of samples decreases. For example for polynomial kernel with degree 1 and Nominal representation need 620 seconds to learn using all dataset and 194 second to learn a new smallest dataset. At this 194 second we need to add also the time needed to select support vectors that was of 209 seconds and the time needed for grouping data (21 seconds). The last two times occurs only one time for all tests with polynomial and Gaussian kernel. The total time for polynomial kernel and degree 1 is 424 seconds. To compute those times in both cases we don't take into consideration the time needed for feature selection. All the time before feature selection we have 19038 features but in the second case we have a reduced dimension of the set. Also some of these times are considerably smaller that the first times. Those values were given using a Pentium IV at 3.2MHhz and 1GB DRAM memory.

In Figure 5.4 is presented comparatively times needed for training the original set with times needed for training the reduced set for both binary and Cornell Smart data representation. As it can be seen the discrepancy between those two sets is great. As an example, for Gaussian kernel the training time decreases from 2554 seconds to 569 seconds for training and 209 seconds for finding support vectors for C=1.8 and Cornel Smart input data representation (and 21 seconds for grouping data). Also we remind that for the reduced time we need 2 feature selection steps, both of them with a smaller dimension in comparison with the original set but all the time with a vector dimension equal to 19038 features.

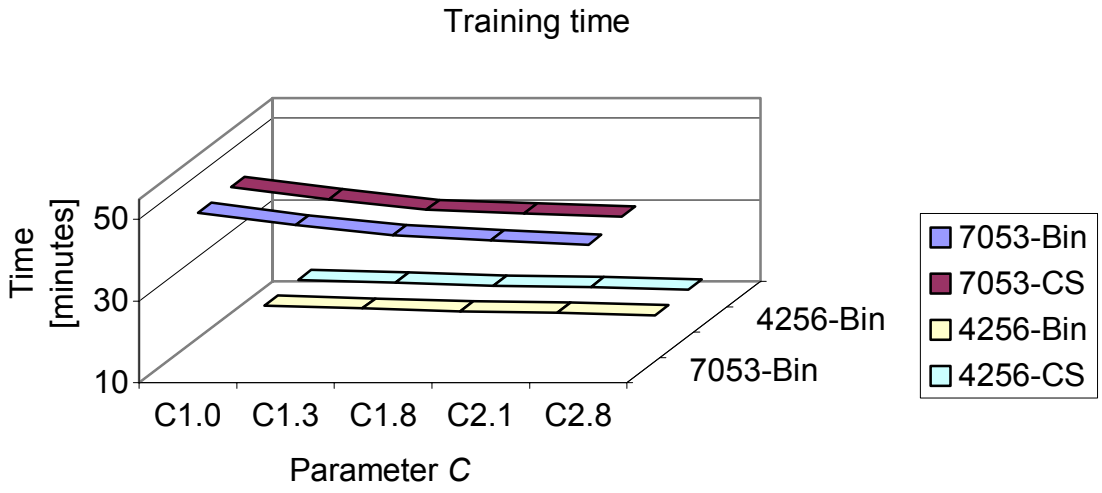


**Figure 5.4 – Comparative training time for Gaussian Kernel**

In Table 5.1 we presented the average difference between the accuracy obtained with the original set and the accuracy obtained with the reduced set. In other words, we present here lose obtained for each type of data representation and for each kernel when we work with a reduced set. For Gaussian kernel and Binary data representation we obtain the greatest lose from all the tests. For polynomial kernel lose is on average with 1.5% in accuracy and also the training time doesn't decrease so much, only with 3 minutes.

| Kernel type | Data representation | Average accuracy |
|---|---|---|
| Polynomial kernel | BINARY | 1.51% |
| | NOMINAL | 1.3% |
| | CONNEL SMART | 1.78% |
| Gaussian kernel | BINARY | 3.81% |
| | CONNEL SMART | 1.89% |

**Table 5.1 – Decrease in average accuracy for all data representation**

Maybe when we will make the tests with entire Reuter's data set, learning with all data will be impossible due to the huge time and memory needed. Those modest results obtained will be useful in choosing the adequate parameters for the learning step. We don't make those tests with all databases because this is not our interest for this PhD and those tests usually take a lot of time. For example, for the entire Reuters database we only started the first step of feature extraction and after this step we had obtained 806791 vectors, each of them having 310033 dimensions (features) and a total of 103 topics.

# 6 Methods for splits the training and testing data set

Our experiments are performed on the Reuters-2000 collection[Reu00]. From all documents we selected the documents having the industry code "System software". We obtained 7083 files that are represented using 19038 features and 68 topics. From these 68 topics we have eliminated those topics that are poorly or excessively represented. Thus we eliminated those topics that contain less than 1% documents from all 7083 documents in the entire set. We also eliminated topics that contain more than 99% samples from the entire set, as being excessively represented. After doing so we obtained 24 different topics and 7053 documents. In all results presented in the second and this PhD report this set of documents was split randomly in a training set ( having 4702 samples) and a testing set (having 2351 samples).

In this chapter will try to see if our algorithm was optimized only for this data set or this optimization is valid for any data set. In order to answer this question we also created different sets that have the same dimension as the first sets but having other random method to split documents into training and testing sets. First random method chose almost randomly where the current document is put, in the training or in the testing set. We said almost randomly because there is one restriction, the training set needs to be larger then the testing set. Usually this method assures that two samples are put in the training set and one sample in the testing set. The second method usually puts a number of samples (say $n$) in the testing set and a number of samples ($2*n$) in the training set. For this method we use $n$ with a value greater than 1. This assures in the end that the testing and training sets have the same dimension as the training and testing set obtained with first method but they contain different samples.

We don't repeat all experiments with these new sets. We chose to repeat only experiments with SVM_FS method for different dimensions of the feature space. Into the Figure 6.1 we show a comparison of results for Polynomial kernel and 475 features. We present here the average over all three methods of representing the input data.

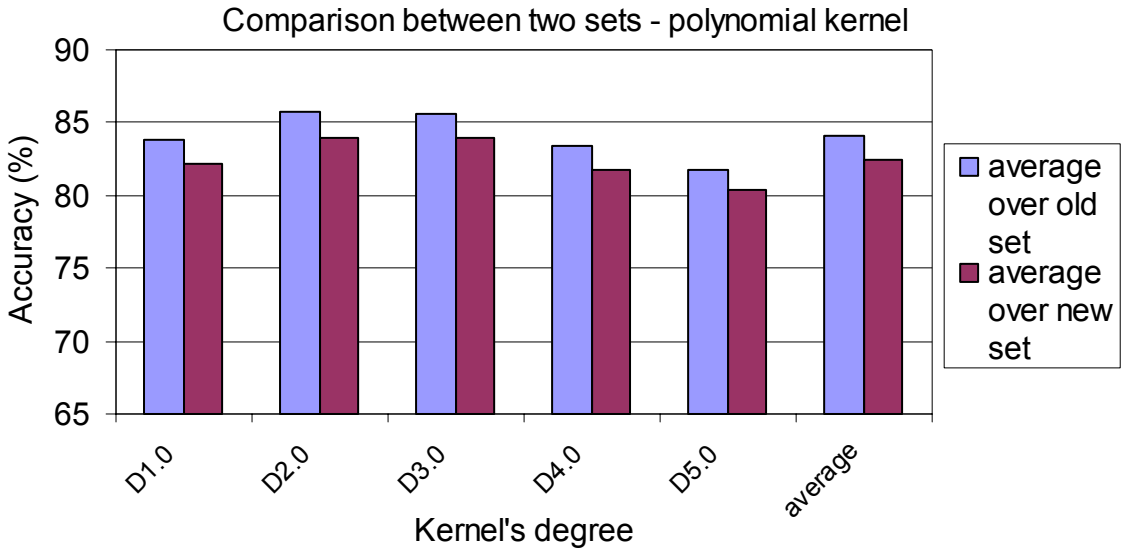


**Figure 6.1 – Comparative results for a set with 475 features**

The last column represents an average over all degrees of the kernel. As it can be observed the results obtained for the old selected set are with 1.61% better in comparison with the newly selected set. We obtain an average of 84.04% for the old sets and 82.43% for the new sets. For this dimension of the feature set the old set obtains better results for polynomial kernel.

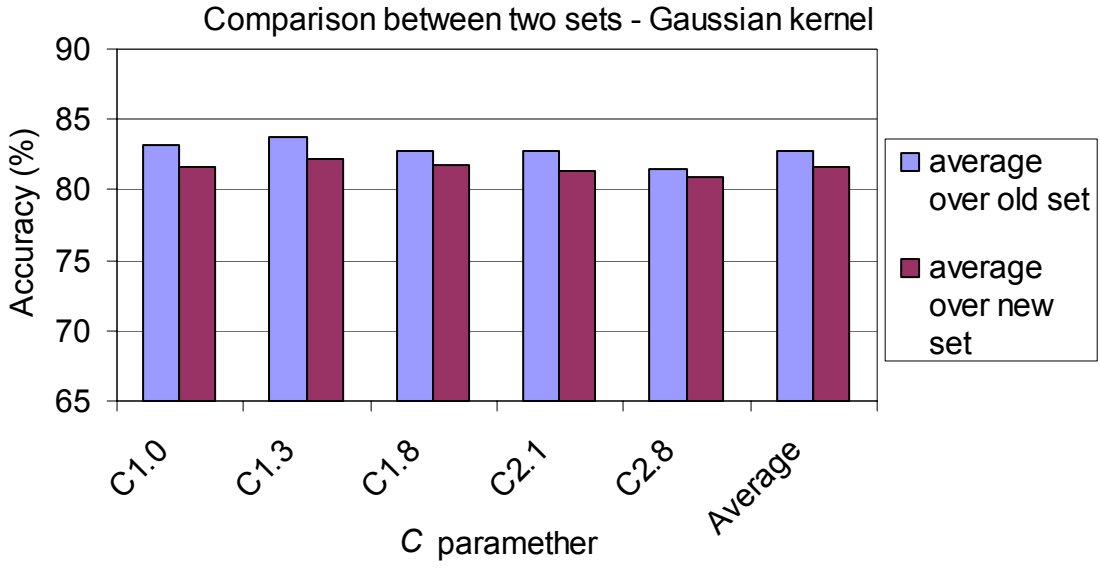


**Figure 6.2 – Comparative results for a set with 475 features**

When using the same dimension of the feature set but with the Gaussian kernel we obtain better results for the old set - 82.77% (1.2% more than with the new data set – 81.57%). Generally speaking for a dimension of 475 features the first selected set returns better results. Even if we have the same number of features those features are different as we use a different set which leads to finding different relevant features. Because we splat the data differently our decision functions (in the classification step) change due to different importance of features (the weight of the features changes) compared to the old set. Those features can also have influence on the accuracy of classifying. As follows we will present results obtained for a dimension of 1309 features.

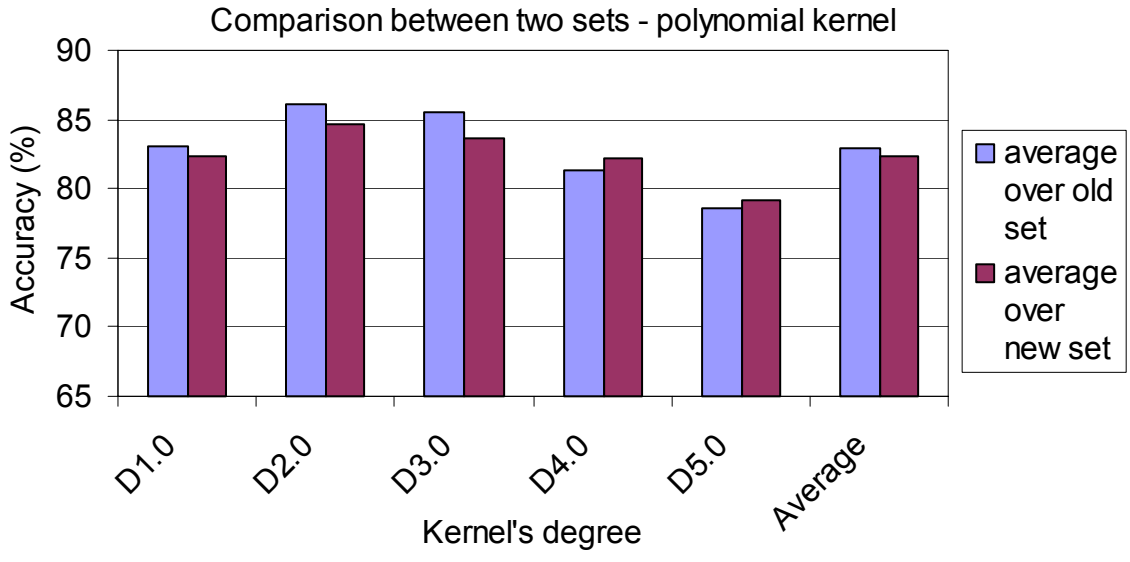


**Figure 6.3 – Comparative results for a set with 1309 features**

For this dimension we obtain a difference between the presented sets of only 0.33% (82.82% for the first selected sets and 82.49% for the newer sets).

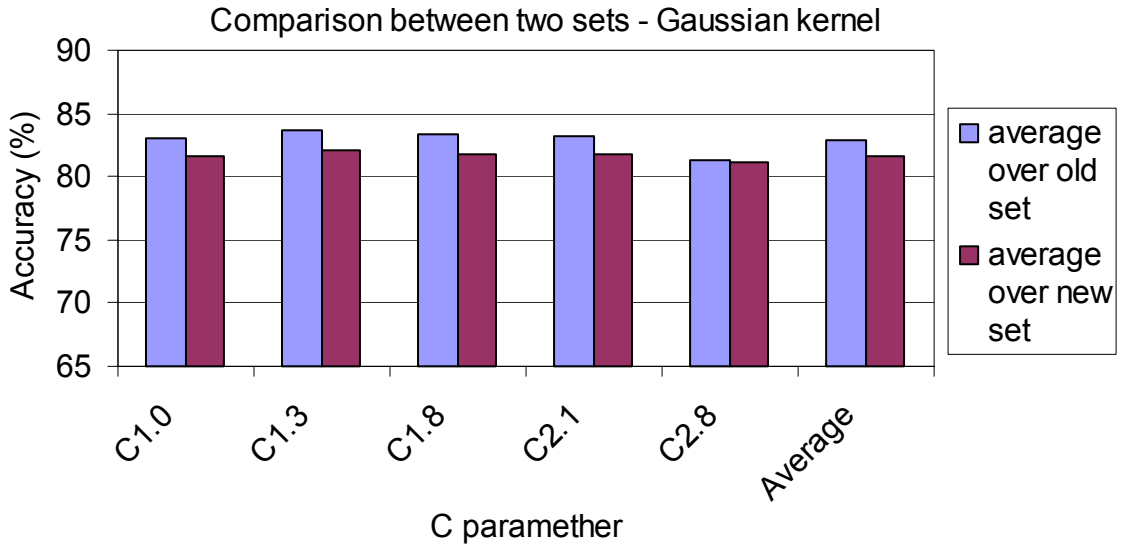


**Figure 6.4 – Comparative results for a dimension of the feature space of 1309 features**

For Gaussian kernel the difference is of 0.81%. Globally for a dimension set of 1309 features also the first selected set obtains better results.

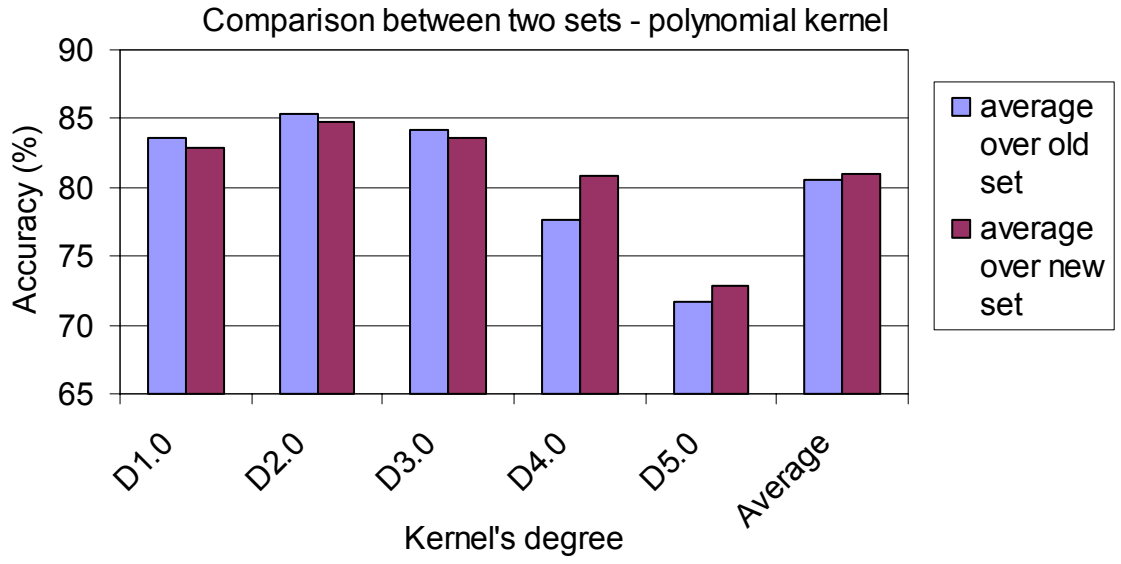


**Figure 6.5 – Comparative results for a set with 2488 features**

For a dimension of the feature space of 2488 features the newer sets obtains better results. It obtains 80.97%, with 0.48% more than the first sets (80.48%). This tendency, surprising, is kept for Gaussian kernel too. The difference is 0.27% more for newer sets (80.97%) in comparison with the first sets.

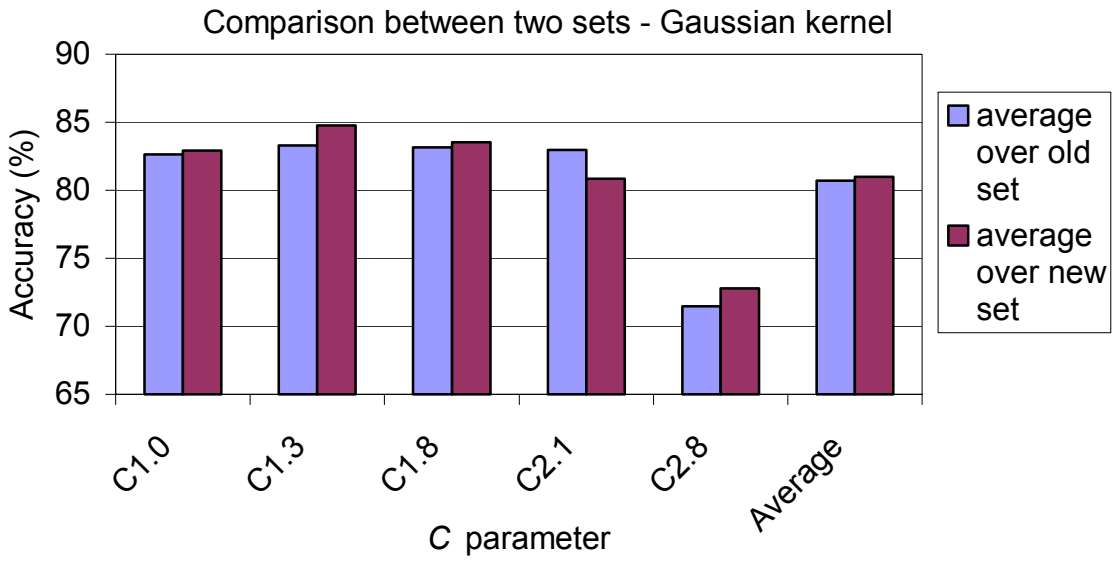


**Figure 6.6 – Comparative results for a dimension of the feature space of 2488 features**

The next two figures present results obtained for a dimension of 8000 features. As it can be observed from these charts the discrepancies between the two sets are greater.

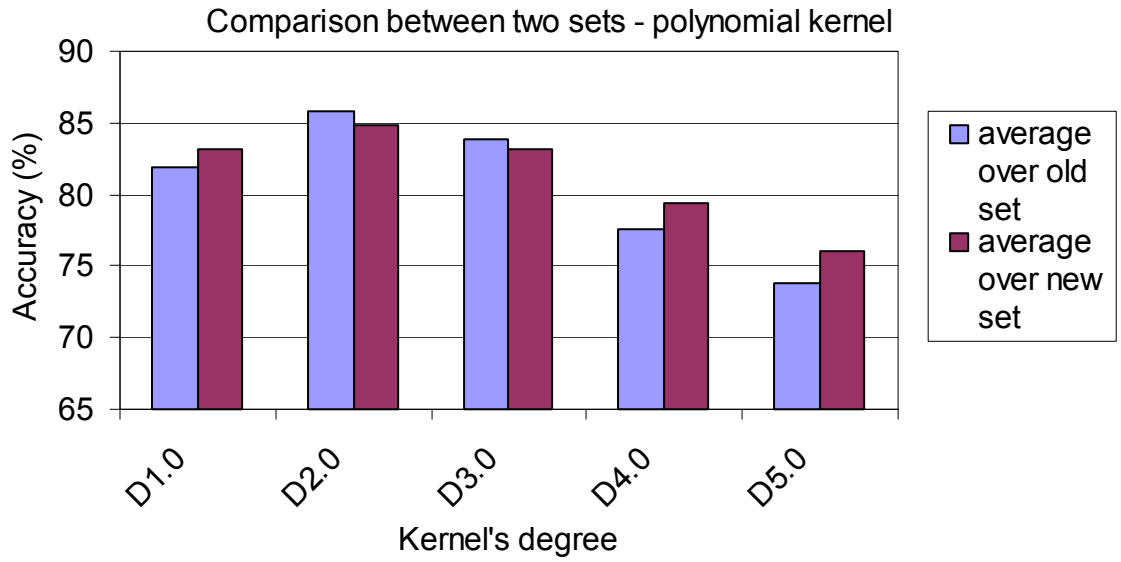


**Figure 6.7 – Comparative results for a set with 8000 features**

For the polynomial kernel with dimension of the feature space equal with 8000 features the difference between these two sets is of 0.72% for the newer sets. On average for all types of input data representation (Binary, Nominal or Cornell Smart representation) and all kernel degrees we obtain with newly selected sets better results. For the Gaussian kernel, represented in Figure 6.8, these tendencies are not kept. Thus the results are greater with 0.58% for the first selected sets in comparison with the new selected sets. On average for the first sets we obtain 82.01%, and 81.59% for the second sets.

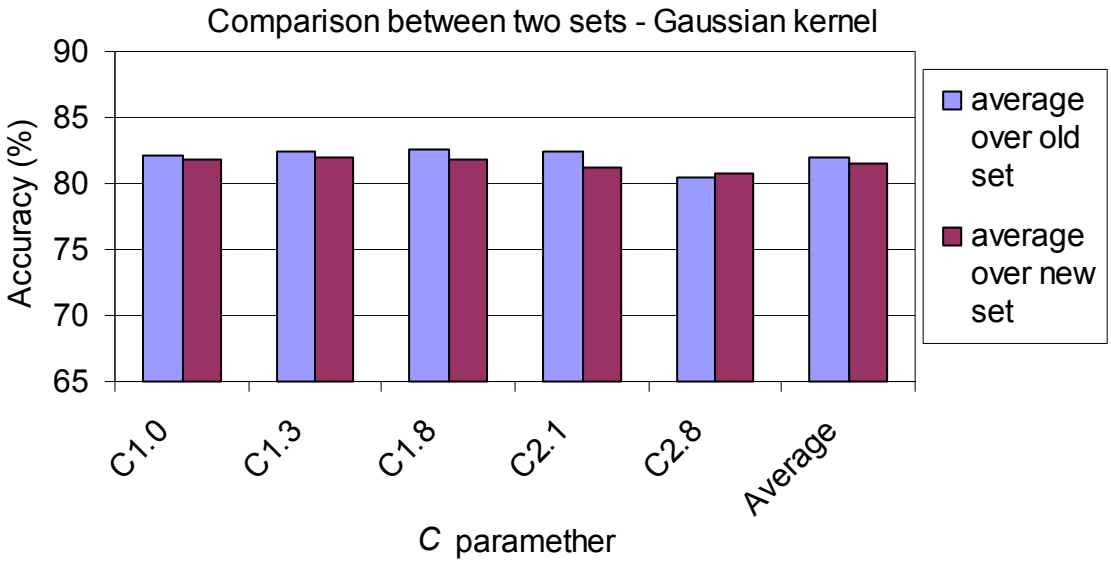


**Figure 6.8 – Comparative results for a dimension of the feature space of 8000 features**

Here we presents results obtained using a features selection method based on Support Vector Machine and after selecting different feature dimension sets we test the accuracy of classification with our classifier also based on Support Vector Machine. A conclusion of these tests is that the results obtained on both sets are equivalent. Sometimes the first sets obtain better results with 1%, sometimes the new sets obtain better results. All the time the difference between these sets is not so great, it never crosses 2%. As a conclusion the results presented in our entire work are not influenced on the selected sets. With other sets our features selection methods and our classification algorithm obtains comparable results. This conclusion encourages us to use our classification in real life to classify web documents.

The most credible results would be the average over each value obtained with each of the sets. In order to have this kind of credibility we would have had to run all our previous tests on different groups of training and testing sets obtain by splitting the initial set. Unfortunately this kind of testing is very time-consuming and we resumed our experiments only to two different groups.

# 7 Conclusions and Further Work

At the beginning of this PhD report we have proposed a new method to better correlate kernel's parameters. The method correlates the degree of the Polynomial kernel with the bias, respectively correlates the constant from the Gaussian kernel with a value that represents the number of distinct features that occurs into the currently used vectors and having weights greater than 0.

In the polynomial kernel case there are more values for which we obtain the best results but our proposed formula assures to hit in almost all cases the best results without having to make more tests to find the good parameter for the bias. Thus we propose that **the bias of the kernel to be correlated with the degree of the kernel (*b=2*d*)**.

For Gaussian kernel our proposed formula always assures to obtain the best results. We also showed that for text classification problems it is not a good idea to use parameter C equal to the number of features, as it is usually used in the literature. Using our proposed formula we obtained **on average results with 3% better for polynomial kernel and results with 15% better for Gaussian kernel**. As far as we know, I am the first author that propos a correlation between these two parameters for both polynomial and Gaussian kernels.

In Chapter 2 of this report, I investigated whether feature selection methods can improve the accuracy of document classification. Here we present a new feature selection method based on Genetic Algorithm (GA_SVM) with a fitness function that uses Support Vector Machine technique. This method is a completion for those three feature selection methods presented into our second PhD report (Random, Information Gain and Support Vector Machine for feature selection SVM_FS). In this report we compare this new method only with SVM_FS method that obtains best results in comparison with the other two methods. We also tested here the influence of the representation of the input data (Binary, Nominal or Cornel Smart representation).

**The best results were obtained when we chose a small (but relevant) dimension of the data set**. We showed that by using between 2.5% to 7% from the total number of features the classification accuracy is significantly better (with a maximum of 86.64% for SVM_FS method, polynomial kernel and nominal data representation). **If we further increase the number of features to more than 10%, the accuracy does not improve or even decreases** (to 86.52% for 2488 and 85.36% for 8000 features). When we used SVM_FS, better classification accuracy was obtained using a small number of features (accuracy of 85.28% for 475 features, representing about 3% from the total number of features) - needing also small training time. Generally speaking, the SVM_FS and GA_SVM methods we obtained comparable results.

We have also observed that the **polynomial kernel obtains better results when we used a nominal data representation** and the **Gaussian kernel obtains better results when we used Cornell Smart data representation**. **The best accuracy was obtained by the Polynomial kernel with a degree one, nominal representation and SVM_FS** (86.64%) in comparison with Gaussian kernel that obtained only 84.85% accuracy for C=1.3, Cornell Smart representation and GA_SVM, but for a greater number of features (2488). We also showed that the training classification time increases only by 3 minutes, as the number of features increases from 485 to 1309 and increases by 32 minutes when number of features increases from 1309 to 2488.

Another problem propose into this report is to develop a meta-classifier with our presented classifiers in order to improve the classification accuracy. So, in the next chapter, we investigated three approaches to build an efficient meta-classifier. Based on our previous work we selected 8 different SVM classifiers. For each of the classifiers we modified the kernel, the degree of the kernel and input data representation. Based on these selected classifiers we computed **the upper limit of our meta-classifier that is 94.21%**. We compared one simple static method based on Majority Vote with two adaptive methods.

With Majority Vote the classification accuracy was 86.38%. As we expected, the documents that are correctly classified by only one classifier can't be correctly classified by this method.

**The SBED method obtains best results**, growing up to 92.04% after 14 learning steps with 2.17% smaller than the upper limit. This method is also the fastest because it chooses first acceptable classifier that might be used. The last method (SBCOS) tries to be the most rigorous because it finds the best component classifier. As a consequence, the training time for SBCOS is greater on average with 21 minutes comparatively with SBED that give the response only in 21 minutes.

In the end of our paper we present the possibility of using more input data. Because the dimension of each vector used is great, when we want to try to learn greater sets we have memory and time problems. In order to do this we implement a way to reduce in the first step the number of vectors from the input set and make two learning steps in order to consider the learning step finished. We noticed that **the classification accuracy decreases with only 1% when we reduce the dataset at 60%** from entire dataset.

In the final chapter we showed that our work is not significantly influenced by the selected training and testing sets. So that we select other training and testing sets and repeated the tests that we made for SVM_FS method. We observed that the results obtained with the new grouped data in the sets are quite equivalent with the first grouped data in sets. Sometimes the first sets obtain better results with 1%, sometimes the new sets obtain better results. All the time **the difference between these sets is not so great, it never crosses 2%**. As a conclusion **the results presented in our entire work are not influenced on the selected sets**. With other sets our features selection methods and our classifier algorithm obtain comparable results. This conclusion encourages us to use our classification in real life to classify web documents.

The next goal of our ongoing work is to adaptation of our work for online Web mining applications, in order to extract and categorized online news. We consider this as a natural extension of our algorithm. We also are interested in classifying larger text data sets (the complete Reuters database or a lot of part of documents from the web), removing infrequent features and improving the existing methods of extracting features for working faster and accurate with more data. Another interesting problem for extracting features is the synonym problem and the polysemy problem that can reduce the size of the vector space. Maybe we could also find association rules between words and eliminate those words that occur together.

In further experiments I will try to combine the classification method with clustering method in order to use labeled and unlabeled data into a hybrid classification algorithm. Our idea is to change the primal step from clustering, when we chose an initial value of data and initialize the Lagrange multipliers $\alpha_i$ (initial hyperplane), into the classification step. In this step a small number of labeled data are presented to a classification algorithm to obtain the $\alpha_i$ coefficients that are used as initial values for clustering process. This offers us the possibility to use in the training part more unlabeled data.

A major issue that occurs in all classification and clustering algorithms is that they are reluctant to fitting in real spaces. For instance they have a problem when they have to deal with new documents for which none of the features are in the previous feature set (the product between the new features set and the previous feature set is the empty set). There definitely are methods of updating the algorithm. The newly obtained algorithm will somehow classify the documents by creating a merge between the feature sets. This problem occurs as the training set can not contain the roots of all existing words. Feature selection methods choose only interesting features. As we see in this report the algorithm obtains better results when using fewer relevant features. Thus training on few features increases the number of documents that can not be then classified. As further development I will test on families of words and use as features only a representative of each family. By doing so the number of features will be significantly reduced and thus we can increase the number of files that can be classified further on. This method could increase the classification accuracy as it is used as a feature selection method. In order to achieve this we could use the [WordNet] database which contains the families of words for the English language.

# 8 Glossary

*Classification accuracy* - the ability to predict the correct class labels, percent of correct classified document

*Clustering* – an unsupervised learning process

*Data scalability* – a system is considered scalable if when the number of input data is enlarged 10 times, its takes no more than 10 times to execute the same process.

*Data set* – the set of all documents took into consideration for training and testing

*Decision function* – in the classification process it represents the solution of the learning problem

*Feature* – here it represents the root of the word that was extracted from the documents

*Feature selection* – according to [Gue00] is the process of selecting a subset of features which maximizes the classification performance of a given procedure over all possible subsets.

*Feature set* – the set of all word roots that occur in all documents from the data set.

*Hyperplane* – a decision boundary and represents a set of points from the training set that meets a constraint expressed as a linear equation.

*Kernel* – a function that computes the inner product in the feature space directly as a function of the original input points, it becomes possible to merge the two steps needed to build a non-linear learning machine.

*Learning with kernel* – systems for efficient training the learning machine in the kernel-induced feature spaces (Support Vector Machine)

*Relevant* – in our context represent a term that has a good influence in obtaining better classification results

*Meta-classification* – a technique for combining the predictions obtained from the base-level models in order to improve the classification accuracy

*Stemming* – the process of extracting the root of a word, in our case only for English language.

*Stop-words* – a set of words that are deemed as “irrelevant” or they appear frequently.

*Support vectors* – in our context they represent a subset of the training patterns that has a great influence in define the decision function.

# 9 References


[Mor05_1] **Morariu, D.**, *Web Information retrieval*, 1st PhD Report, University „Lucian Blaga“ of Sibiu, February, 2005, http://webspace.ulbsibiu.ro/ daniel.morariu/html/Docs/Report1.pdf

[Pla99] Platt, J., *First training of support vector machines using sequential minimal optimization*. In B. Scholkopf, C.J.C. Burges, and A. J. Smola, editors, Advances in Kernel Methods – Support Vector Learning, pages 185-208, Cambridge, MA, 1999, MIT Press.

[Sch02] Schoslkopf B., Smola A., *Learning with kernels. Support Vector Machines*, MIT Press, London, 2002.

[Nel00] Nello C., Swawe-Taylor J., *An introduction to Support Vector Machines*, Cambridge University Press, 2000

[Vap95] Vapnik V., *The nature of Statistical learning Theory*, Springer New York, 1995

[Pla99] Platt J., *First training of support vector machine using sequential minimal optimization*. In B. Scholkof, C.J.C. Burges, and A. J. Smola, editors, Advance in Kernel Methods – Support Vector Learning, pages 185-208, Cambridge, MIT Press, 1999

[Lib] http://www.csie.ntu.edu.tw/~cjlin/libsvm

[Mor06_1] **Morariu, D.**, Vintan, L., *A Better Correlation of the SVM Kernel's Parameters*, Proceedings of the 5th RoEduNet IEEE International Conference, Sibiu, June, 2006

[Kim00] Kim G., Kim S., *Feature Selection Using Genetic Algorithms for Handwritten Character Recognition*, Proceedings of the Seventh International Workshop on Frontiers in Handwriting Recognition, Amsterdam, 2000

[Gue00] Guerra-Salcedo, C., Chen S., Whitley D., Smith S., *Fast and Accurate Feature Selection Using Hybrid Genetic Strategies*, CEC00, Proceedings of the Congress on Evolutionary Computation, CEC00, July 2000

[Jeb00] Jebara, T., and Jaakkola, T., *Feature selection and dualities in maximum entropy discrimination*, In Uncertainty in Artificial Intelligence 16, 2000

[Jeb04] Jebara T., *Multi Task Feature and Kernel Selection for SVMs*, Proceedings of the 21st International Conference on Machine Learning, Banff, Canada, 2004

[Gol89] Goldberg G., *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison Wesley, New York, 1989

[Bal97] Ballard D., *An Introduction to Neural Computation*, the MIT Press, 1997

[Lug98] Luger G. F., Stubblefield W. A., *Artificial Intelligence*, Addison Wesley Longman, Third Edition, 1998

[Hol75] Holland, J., *Adaptation in Natural and Artificial Systems*. University of Michigan Press, 1975

[Whi94] Whitely, D., *A genetic Algorithm Tutorial, Foundations of Genetic Algorithms*, ed. Morgan Kaufmann

[Mor05] **Morariu, D.**, *Classification and Clustering using SVM*, 2nd PhD Report, University „Lucian Blaga“ of Sibiu, September, 2005, http://webspace.ulbsibiu.ro/daniel.morariu/html/Docs/Report2.pdf

[Brz94] Brazdil, P. B. Gama, J., and Henery, B., *Characterizing the applicability of classification algorithms using meta-level learning*. Proceedings of the 7th European Conference on Machine Learning (ECML-94)(83-102), 1994

[Lin02] W.-H. Lin , A. Houptmann, *News Video Classification Using SVM-based Multimodal Classifier and Combination Strategies*, International Workshop on Knowledge Discovery in Multimedia and Complex Data, Taiwan, 2002

[Dim00] Dimitrova, N., Agnihotri, L., Wei, G., *Video Classification Based on HMM Using Text and Face*, Proceedings of the European Conference on Signal Processing, Finland, 2000

[Siy01] Siyang, G., Quingrui, L., Lin, M., *Meta-classifier in Text Classification*, http://www.comp.nus.edu.sg/~zhouyong/papers/cs5228project.pdf, a research group, 2001

[Lin02_2] W.-H. Lin , R. Jin, A. Houptmann, *A Meta-classification of Multimedia Classifiers,* International Workshop on Knowledge Discovery in Multimedia and Complex Data, Taiwan, May 2002

[Kit98] Kittler, J., Hatef, M., Durin, R.P.W. and Mates, J., *On Combining Classifiers*, IEEE Transactions and Pattern Analysis and Machine Intelligence, 1998.

[Mor06_2] **Morariu, D.**, Vintan, L., Tresp, V., *Feature Selection Method for an Improved SVM Classifier*, Proceedings of the 3rd International Conference of Intelligent Systems, ISSN 1305-5313, pp. 83-89, Prague, August, 2006

[Mor06_3] **Morariu, D.**, Vintan, L., Tresp, V., *Evolutionary Feature Selection for Text Documents using the SVM*, Accepted at 3rd International Conference on Neural Network and Pattern Recognition NNPR'06, Barcelona, Spain, October, 2006

[Mor06_4] **Morariu, D.**, Vintan, L., Tresp, V., *Meta-classification using SVM classifier for Text Document*, Accepted at 3rd International Conference on Neural Network and Pattern Recognition NNPR'06, Barcelona, Spain, October, 2006

[Gun03] Gunnain, R., Menzies, T., Appukutty, K., Srinivasan, A., *Feature Subset Selection with TAR2less*, Available from http://menzies.us/pdf/03tar2less.pdf, 2003

[Cha03] Chakrabarti, S. – *Mining the Web - Discovering Knowledge from hypertext data*, Morgan Kaufmann Press, 2003

[Kan02] Kanungo, T., Mount, D.M., Netanyahu, N.S., Pitko, C.D., Wu,A., *An Efficient* k-*Means Clustering Algorithm: Analysis and Implementation*, IEEE Transactions on Pattern Analysis and Machine Intelligence, col. 24, no. 7, July 2002

[Ord03] Ordones, C., *Clustering Binary Data Streams with K-means*, Conference of Data Mining and Knowledge Discovery, San Diego, 2003

[Bei02] Beil, F., Ester, M., Xu, X., *Frequent Term-Based Text Clustering*, In SIGKDD02 Exploration: Newsletter on the Special Interest Group on Knowledge Discovery & Data Mining, ACM Press, Alberta Canada, 2002

[Ber02] Berendt, B., Hitho, A., Syumme, G., *Towards Semantic Web Mining*, Springer-Verlag Berlin Heidelberg, ISWC 2002, pp. 264-278, Berlin, 2002

[Yu03] Yu, H., yang, J., Han, J., *Classifying Large Data Sets Using SVMs with Hierarchical Clusters*, In SIGKDD03 Exploration: Newsletter on the Special Interest Group on Knowledge Discovery & Data Mining, ACM Press, Washington, DC, USA, 2003

[Hun03] Hung, C., Wermter, S., Smith, P., *Hybrid Neural Document Clustering Using Guided Self-Organization and WordNet*, Published by the IEEE Computer Society, IEEE 2003

[Reu00] Reuters Corpus: http://about.reuters.com/researchandstandards/corpus. Released in November 2000

[WordNet] http:\\www.cogsci.princeton.edu\wn2.0 – online lexical system