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

Contributions to Automatic Knowledge Extraction from Unstructured Data

PhD Thesis

Author:
Ionel Daniel MORARIU, MSc

PhD Supervisor:
Professor Lucian N. VINȚAN, PhD

SIBIU, 2007

Contents

"Anyone who has never made a mistake
has never tried anything new".

Albert Einstein

1 Introduction and Main Objectives

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 continually grows 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 the documents contain, 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 challenge. 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 fulfilled in order to solve the problem. One of those requirements has a high influence on the final accuracy of classification. Actually, believe that it is impossible to approach this entire problem even in a PhD thesis. In the last years a lot of research efforts are centered on automatically document classification. This PhD thesis brings contributions in **developing, enlarging and improving a powerful Support Vector Machine Classifier and some feature selection methods also for text and Web documents**.

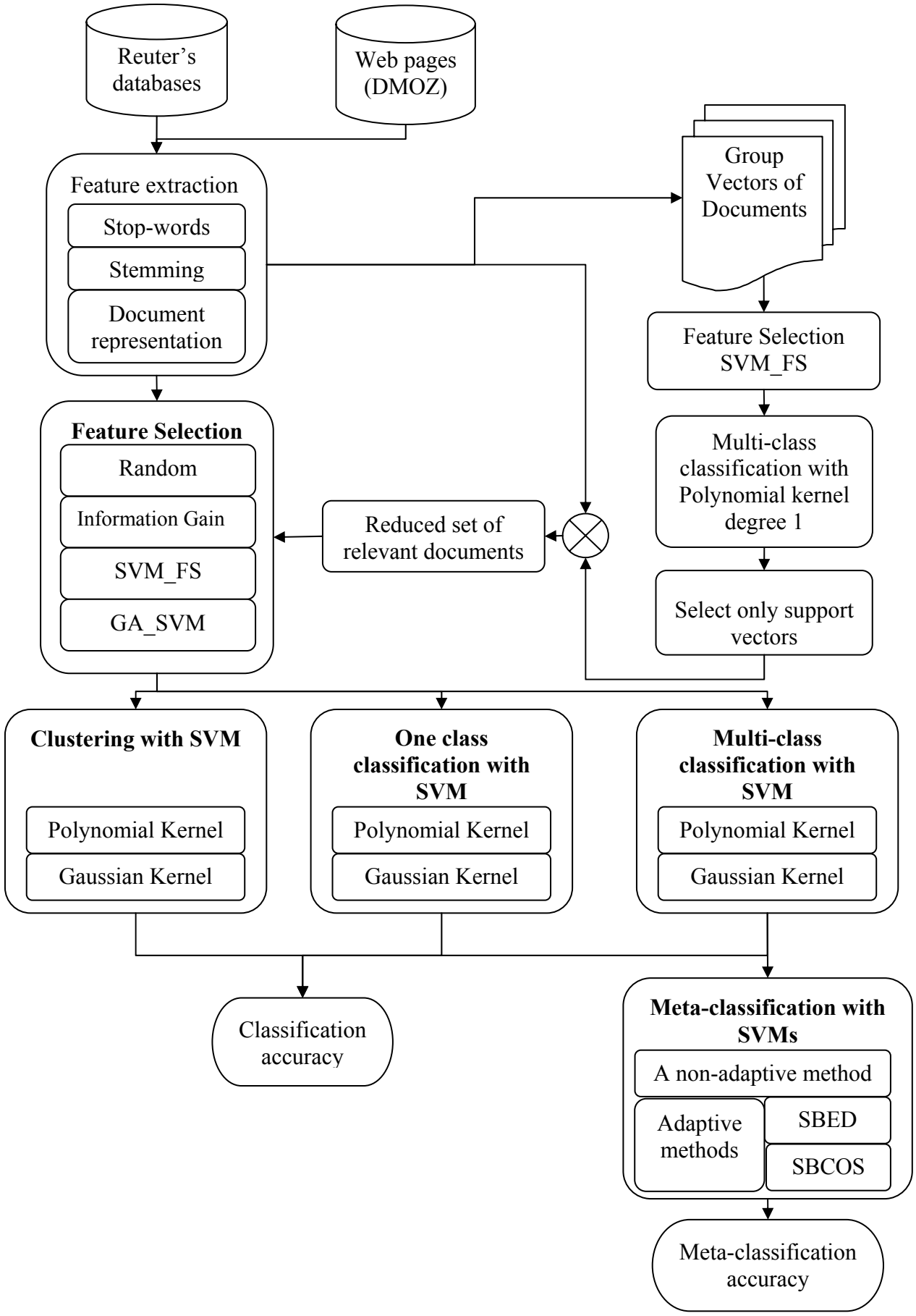

Figure 1.1 – Documents classification flowchart. My view.

I considered the process of automatically document classification as a flowchart where each part receives some information, process it and then further transfer it, as showed in Figure 1.1. Each part of the flowchart 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.

Regarding to the thesis' structure I chose to present, grouped by domains, the state of the art and my research and contributions to this area.

Thus, chapter 2 contains some prerequisites for the work that I presented in this PhD thesis. I presented some techniques for preprocessing the text documents, especially preprocessing of the Reuters 2000 database and a database created using Web documents extracted from DMOZ Web directory. In the first step, using as input a set of text documents (text files or Web pages) I represent them as a form of feature vectors. These are frequency vectors of the words that occur into the document. This representation is closer to the one understood by computer. Due to the huge dimensionality of resulting vectors a selection of relevant features is necessary. Thus, chapter 4 contains four developed methods of features selection: Random selection, Information Gain selection, Support Vector Machine selection and Genetic algorithm with SVM fitness function. In chapter 3 I present in details the algorithm used for classification based on Support Vector Machine (SVM) technique [Pla99], focusing on the SVM as a classification process. My original contribution consists in **a method of correlation of kernel's parameters to improve the classification accuracy**. In chapter 5 I present the experiments that lead to the choice of the kernel correlations and its improvement in comparison with LibSvm implementation.

The flowchart ends with the **development of a meta-classifier in order to improve classification accuracy** that is presented in chapter 6 and tries to explore the classifiers' synergism. This flowchart also contains some contributions that improve the process of classification by making it more reliable, as presented in chapter 7. Therefore I present a methodology that **makes my 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 two antagonist main objectives – more data in the set and smaller response time with good accuracy.

This thesis ends with a chapter pointing out my original contributions and proposes some further perspectives of development in this field, a glossary and the references that have been used.

Acknowledgments

In writing my thesis I have been fortunate to be assisted by technical experts and to have the support of family and friends. I would like to thank all of them.

I would like to express my sincere gratitude to my PhD supervisor Professor Lucian VINTAN, PhD, for his valuable guidance, support and responsible scientific coordination during the PhD stage. He constantly spent his time to review the PhD developed work and by sharing his insights with me in discussions that helped improving my researches.

At the same time, I would like to thank SIEMENS AG, CT IC MUNICH, Germany, especially Vice-President Dr. h. c. mat. Hartmut RAFFLER, for the 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 and for his valuable guidance in this wide interesting domain of research. I also want to thank Dr. Kai Yu for his inputs that help in developing my ideas.

I am also grateful to my colleagues at Computer Science Department from "Lucian Blaga" University of Sibiu for their support, professional suggestions and for creating a good working environment that allowed me to develop my research.

I would also like to express my grateful thoughts for the PhD reviewers, thanking them for the considerable efforts they put in reviewing this work.

Last but not least I would like to thank my family and friends for their continuous encouragements and for gently accepting the fact that I wasn't around them as much as I should have been.

"To be conscious that you are ignorant
is a great step to knowledge."

Benjamin Disraeli

2 Methods for Knowledge Discovery in Data

In fact a substantial fraction of the available information are stored in text or document database which consist of a large collection of documents from various sources such as news articles, research papers, books, Web pages, etc. Data stored in text format are considered semi-structured data in that they are neither completely unstructured nor completely structured. A document may however contain a few structured fields such as title, authors, publication, data, etc. Unfortunately those fields usually are not filled in; the majority of people don't loose time to complete these fields. Some researchers suggest that the information needs to be organized during the creation time, using some planning rules. This is pointless because most of people will not respect them. This is considered a characteristic of unordered egalitarianism of Internet [Der00]. Any attempt to apply the same organizing rules will determine the users to leave. The result for the time being is that most information needs to be organized after it was generated and searching and organizing tools need to work together.

As more and more information is available, effective information retrieval is a challenge without summarization and indexing of the documents content. Categorizing documents is one solution for this problem and recent tendencies are to combine them with user's profile information or semantic analysis. In order to perform categorization we face the task of classifying natural language data into predefined categories. Many methods of classification and machine learning techniques have been used in the last years to classify documents. But unfortunately most of the available information is unlabeled and there have been a lot of attempts to use clustering methods or combine them with classification methods.

Impressive amounts of information potentially relevant to the user are contained in the Web, this information being very chaotic at the moment. The Web is becoming a huge repository of information, and it is built in uncoordinated manner but yet restrictive. The Web is an environment that grows continuously, it is populated and it implies a growth in participants, without having a coordinated manner of building. These characteristics have as a result both pluses and minuses. One of the pluses is the increase in the variety of the content. Information found on the Web is undoubtedly larger than the one available trough news, radio and television. This is due to a considerably large number of Web developers. The lack of organization and the heterogeneity of the data represent one of the minuses when trying to find information. Exploring the Web is the only way to find relevant information, this method being unfortunately time consuming. Queries that contain words are the simplest method of searching information on the Web – method that usually doesn't return the expected results. Inexperienced users don't know how to formulate complex queries (like using preposition in the queries) and need more time to find information.

The growth of the Web in diversity and dimension gives it an inestimable value that makes it an active repository of information. For the first time we can speak about an environment that has almost as many authors as it have readers. The growing Web content makes it more and more

difficult to estimate real values from whole content. Its unsupervised progress makes it contain a large number of redundant information.

The difficulty in finding and organizing relevant information grew exponentially with the growth of information on the Internet and Intranets. Methods for searching relevant information for the user were developed at the same time. Thus, a series of "search engines" were developed. Search engines research grew rapidly in the past years, in areas such as algorithms, strategies and architecture, increasing both effectiveness and quality of results. People want to find relevant information quickly. Searching information on the Web can be a frustrating activity when a search engine returns thousands of documents for a given query. When searching, a user inputs a simple keyword query (two or three words usually). The query response is usually a list of pages ranked based on their similarity to the query. Search tools today have the following problems [Cha00]:

- *Low precision* - due to the irrelevancy of many of the search results – this leading to difficulties in finding relevant information;
- *Low recall* - due to the inability to index all the information available on the Web – this leads to difficulties in finding relevant information in un-indexed documents in the search engine database.

Most search engines are constantly looking on the net for new Web pages and they use the page's summary or some reported keywords, to index the pages in a great database. When a user inputs some specific keywords the searching engine returns the addresses of all documents indexed in the database on those keywords. In order to raise the result's quality, the search engine applies various functions in order to assign relevance to a page (e.g.: page rank, similarity, back link and mixed approaches) and gives precedence to pages with high weight, supposedly indicating greater relevance.

2.1 Genesis: Data Mining in Databases

Data mining [Jai01] refers to extracting or "mining" knowledge from large amounts of data. It is the short term for "knowledge mining from data". Many people treat data mining as a synonymous for another popular used term, Knowledge Discovery in Databases, others view data mining simply as an essential step in the process of knowledge discovery in databases. This step is about solving problems by analyzing data presented (usually) in databases and trying to discover such characteristics that can be used to organize massive databases. Thus data mining can be defined as the process of discovering patterns in data and relationships between attributes from data. This process must be automatic or (more usual) semi-automatic. Thus data mining represents the process of discovering interesting knowledge from large amounts of data stored either in databases, data warehouses, or other information repositories. Typically data mining systems have the following major components:

- *Databases, data warehouses, or other information repositories*: Databases are created to perform On-Line Transaction and query Processing (OLTP), covering most of day-to-day operations and being detailed to be easily used for decision making. Data warehouse systems offer help, for the specialists, in data analysis and decision making, collective referred to as On-Line Analytical Processing (OLAP), providing facilities for summarization, aggregation, storing and managing information at different levels of granularity. Information repositories can also be flat files.
- *Databases or data warehouses server* – is responsible for fetching the relevant data, based on the user's data mining request.

- *Knowledge base* – includes the domain knowledge that is used to guide the search or to evaluate the interestingness of resulting patterns. The knowledge can include hierarchical concepts, used to organize attributes or attribute values into different levels of abstraction. The knowledge can be used to assess a pattern's interestingness based on its unexpectedness or additional interestingness constraints or thresholds and metadata.
- *Data mining engine* – is the essential component of the data mining system and ideally consists of a set of functional modules for tasks such as characterization, association, classification, cluster analysis, and evolution and derivation analysis.
- *Pattern evaluation modules* – component typically employing interestingness measures and interacting with the data mining modules so as to focus the search towards interesting patterns. It can used interestingness thresholds to filter out discovered patterns. Alternatively, the pattern evaluation module may be integrated with the mining module, depending on the implementation of the data mining method used. Usually the evaluation of pattern interestingness is put in the mining process so as to confirm only the interesting parents searched.
- *Graphical user interface* – is the module for interaction between users and the data mining systems, allowing the user to interact with the system by specifying a data mining query or task, providing information to help focus the search, and performing exploratory data mining based on the intermediate data mining results. This component allows the user to browse databases and data warehouses schemas or data structures, evaluate mined patterns, and visualize the patterns in different forms.

The process of knowledge discovery in database has more steps, data mining being one of these steps:

- Preprocessing data
- Data mining
- Pattern evaluation
- Knowledge presentation

2.1.1 Preprocessing Data

This is an important step in the process of knowledge discovery. Today real-world database or repository data are highly susceptible to noise, incomplete and inconsistent data due to their typically huge size. Its aim is to prepare data for analyzing. There are a number of data preprocessing steps:

- Data cleaning – to remove noise (a random error or variance in a specific data), and to correct inconsistencies (the same object appears with two different attribute values) in the data (for example a given concept may have different names in different databases, or the same person can be registered as "Bill" in one database, but "William" in another);
- Data integration – to merge data from multiple sources into an appropriate form for mining. It combines data from multiple sources into a coherent data store. It refers to the homogeneity of data;
- Data selection – to select relevant information for analysis. Information is selected as relevant from user's point of view;
- Data transformation – to prepare data for analysis. It contains operations like normalizing data (scaling the values of the attributes so that they fall within a specified range) that can improve the accuracy and efficiency of mining algorithms;

- Data reduction – to reduce data size by aggregation, eliminating redundant features, or clustering.

2.1.1.1 Cleaning Data

C*leaning data* is the first sub-step and prepares data for processing. Because vast amount of data are involved in the data mining process, these data are usually incomplete, noisy, and inconsistent. Data cleaning routine attempts to fill the missing values, or smooth out noise while identifying outliers, and to correct inconsistencies in the data. Outliers are considered values out of range or noise in data. Different methods can be used in the process of filling missing values:

- The "*ignore the samples*" method is not very effective, unless the samples contain several attributes with missing values. This method is especially poor when the percentage of missing values per attribute varies considerably;
- Another method is "*fill the missing value manually*", where the great disadvantage of this method is that it is time consuming and may not be feasible when it is given a large data set with many missing values;
- "*Use a global constant to fill in the missing values*" is the method by which all missing attribute values are replaced with the same constant. This method is simple but it is not recommendable because the mining process may mistakenly think that they form an interesting concept;
- Another method is "*use the attributes mean to fill in the missing value*" where all missing values are replaced with the average of the values for the same attribute in the database;
- "*Use the most probable value to fill in the missing value*" is a good method but it is time consuming because the probable value is determined by the use of regression, inference-based tools using Bayesian formalism, or decision tree induction.

Due to usually noisy data the cleaning data process also uses techniques for smoothing noise in data. The noise is a random error or variance in a specific variable. There are some techniques for smoothing data [Jai01]. Usually these methods try to eliminate those values that occur sporadically but without any assurance that those data are not novelty in data. As a result these techniques usually use a variable threshold for eliminating noise at different levels to assure that novelty is not being eliminated. Some of these techniques are:

- *Binning* methods smooth sorted data values by consulting its "neighborhood", that is, the values around it. The sorted values are distributed into a number of "buckets" called *bins*. Because binning methods consult the neighborhood of values, they perform local smoothing. In this smoothing by bin means method, each value in a bin is replaced by the mean value of the bin. In smoothing by bin boundaries, the minimum and maximum values in a given bin are identified as the bin boundaries and each bin value is then replaced by the closest boundary value.
- Another method is *Clustering* where outliers may be detected by clustering. Similar values are organized into same groups, or "clusters" and values that fall outside of the set of clusters may be considered outliers.
- *Regression* is a method where data can be smoothened by fitting the data through a function such as the regression. *Linear regression* can be used in finding the best line to fit two variables, so that one variable can be used to predict the other variable. *Multiple linear regressions* are an extension of the linear regression in which more than two variables are involved and the data is fitted in a multidimensional surface. Using regression to find a

mathematical equation to fit the data helps smooth out the noise. Usually outliers are identified by using a combination between computer and human inspection. Many methods for data smoothing are also methods for data reduction involving discretization.

Techniques of data preprocessing want to improve the data quality, thereby help to improve the accuracy and efficiency of the subsequent mining process. Preprocessing is an important step in the knowledge discovery process because a quality decision must be based on quality data.

Data recorded in transactions can produce inconsistencies in the data. There may also be inconsistencies due to data integration, where a given attribute can have different names in different databases. Inconsistencies can also be redundancies. Some of these redundancies can be corrected manually or using methods for reducing noise.

2.1.1.2 Data Integration and Transformation

The process of data mining often requests merging data from multiple data stores. Thus in the ***data integration*** steps data from multiple sources are combined into a coherent data store. The source may include multiple databases, data cubes or flat files. Data cube consists of a lattice of cuboids, each corresponding to a different degree of summarization of the given multidimensional data. Partial materialization refers to the selective computation of a subset of the cuboids in the lattice and full materialization refers to the computation of all of the cuboids in the lattice. There are a number of issues to consider during data integration. *Scheme integration* is the method that merges different kinds of input data (e.g. different file formats, different structure names, a. o.). Other important issues can be *Redundancy* where an attribute is considered redundant if it can be "derived" from another attribute. Inconsistencies in attributes dimension or naming can also cause redundancies in the resulting data set. Redundancies can be detected using correlation analysis, where analysis can measure if one attribute implies the other, based on the available date. The correlation between two attributes can be measured by [Jai01]:

$$r_{A,B} = \frac{\sum_{k=1}^{n}(A_k - \overline{A})(B_k - \overline{B})}{(n-1)\sigma_A \sigma_B} \tag{2.1}$$

where A and B are the attributes, *n* represents the number of samples, $\overline{A}$ *and* $\overline{B}$ are the respective mean values of A and B, and σ_A *and* σ_B are the standard deviation of A and B computed thus:

$$\sigma_A = \sqrt{\frac{\sum_{k=1}^{n}(A_k - \overline{A})^2}{n-1}} \tag{2.2}$$

The correlation coefficient ranges from 1 (perfectly correlated), through 0 (no correlation) to -1 (perfectly invert correlated). If the resulting of $r_{A,B}$ are greater then 0, then A and B are positively correlated, and if the value of A increases then the value of B increases. When $r_{A,B}$ tend to 0 then there is no correlation between A and B. Hence, a high value may indicate that A (or B) can be removed as a redundancy. If the resulting is equal with 0, then A and B are independent and there is no correlation between them. If the resulting value is less than 0, then A, and B are negatively correlated (when the values of the one attribute increase as the values of the other attribute decrease). When redundancies between attributes are detected, duplication at the sample level is detected. The above formulas fail when $A_k = \overline{A}$ (or / and $B_k = \overline{B}$) $\forall k = \overline{1,n}$, when the values are constants, leading to nonsense values (in this particular case the attribute need to be eliminated). Another important issue in data integration is *detection and resolution of data value conflict*. This is due to differences in representation, scaling or encoding. For example some databases can use

the British unit measure and also can use American system unit. Careful data integration from multiple sources can help reduce and avoid redundancies and inconsistencies in the resulting data set.

Data transformation is used to transform or consolidate data into forms appropriate for mining. Data transformation can involve some methods like *smoothing* (for noise removal method presented in 2.1.1.1), *aggregation* (for summarizing the data), *generalization* (for replacing the low level data with a high level concept), *normalization* (rescaling attributes to fit in the specified range) and *attribute construction* (construction and adding of new attributes to help the mining process). The attribute is normalized by scaling its values so that they fit to a small specified range.

Normalization is particularly useful for classification algorithms involving, for example, neural networks or distance measurements. There are many methods for data normalization. Method min-max normalization performs a linear transformation on the original data. If that min_A and max_A are the minimum and maximum values of an attribute A, this method maps a value *v* of A to *v'* in the range [new_min_A, new_max_A], where new_min_A and new_max_A are the limits of the new range, by computing:

$$v' = \frac{v - \min_A}{\max_A - \min_A}(new\_\max_A - new\_\min_A) + new\_\min_A \qquad (2.3)$$

This method preserves the relationships between the original data values. The second normalization method is *z-score normalization* where the values for an attribute A are normalized based on the mean and standard deviation of A. Thus a value *v* is normalized to v' by computing:

$$v' = \frac{v - \overline{A}}{\sigma_A} \qquad (2.4)$$

where $\overline{A}$ and σ_A are the mean and the standard deviation for the attribute A (equation 2.2). This method is used when the minimum and the maximum for the attribute A are unknown, or when are outliers dominate. Another method is *normalization by decimal scaling* where the A's attribute values decimal point is moved. The number of decimal points moved depends on the maximum absolute value of A. A value *v* of A is normalized to *v'* by computing:

$$v' = \frac{v}{10^j} \qquad (2.5)$$

where *j* is the smallest integer such that max(|v'|)<1. Obviously, normalization can significantly modify the original data but the data profile (data distribution) will be kept.

In **attribute construction**, the new attributes are constructed from the given attributes and added in order to help improve the accuracy and understanding of the structure in the high dimensional data. Attribute construction can help ease the fragmentation problem when decision tree algorithms are used for classification. By combining attributes, attribute construction can discover missing information about the relationship between data attributes that can be useful for knowledge discovery.

2.1.1.3 Data Reduction

Complex data analysis and data mining on huge amounts of data may take a very long time, making such analysis impractical or infeasible. Data reduction is a technique that can be applied to obtain a reduced representation of the data set that is much smaller in volume, and closely maintains the integrity of the original data. That is, mining on reduced data set should be more

efficient yet produce the same (or almost the same) analytical results. There are some techniques for data reduction (for details see [Jai01]):

- *Data cube aggregation* – some aggregation operations are applied to the data in the construction of the data cube. Data cubes store multidimensional aggregated information. Each cell holds an aggregated data value, corresponding to the data point in multidimensional space. Concept hierarchies may exist for each attribute, allowing the analysis of data at multiple levels of abstraction. Data cubes provide fast access to pre-computed, summarized data, thereby benefiting on-line analytical processing as well as data mining. The cube created at the lowest level of abstraction is referred to as the *base cuboid.* A cube for the highest level of abstraction is the *apex cuboid.* Data cube created for various levels of abstraction are often referred to as *cuboids*. Because data cube provide fast access to pre-computed data, they should be used when possible to reply to queries regarding aggregated information.
- *Dimension reduction* – irrelevant or redundant attributes or dimensions may be detected and removed. Typically, attribute subset selection methods are applied, where the goal is to find a minimum set of attributes such that the resulting probability distribution of the data classes is as close as possible to the original distribution obtained using all attributes. The "best" or "worst" attribute are determined using tests of statistical significance, which assume that the attributes are independent of one another. Many other attribute evaluation measures can be used, such as the information gain and the mutual information (presented in the next paragraph).
- *Data compression* – encoding mechanisms are used to reduce or "compress" the data set size. There are two popular methods of lossy data compression (when we can reconstruct only an approximation of the original data, with losses). First there is the *wavelet transformation* method where the linear signal processing techniques are applied to the data vector, and transform it into a numerically different vector of wavelet coefficients. The two vectors, the original and the wavelet, are of the same length, but the wavelet vector can be truncated and can be retained by storing only a small fraction of the strongest coefficients. The second method is the *principal component analysis* where the idea is to search for an orthogonal vector that can best be used to represent the data and the dimension is smaller than the original dimension. The original data are thus projected onto a much smaller space.
- *Numerosity reduction* – the data are replaced or estimated by alternative, smaller data representation. These techniques may be parametric (where the model is used to estimate the data, so that typically only the data parameter needs to be stored, instead of the actual data), or nonparametric (where histograms, clustering and samples are used for storing reduced representation of the data).
- *Discretization and concept hierarchy generation* – is used to reduce the number of values for a given continuous attribute by dividing the range of the attribute into intervals. Interval labels can then be used to replace actual data values reducing the number of values for an attribute. A concept hierarchy allows the mining of data at multiple levels of abstraction and is a powerful tool for data mining. There are different methods for numeric concept hierarchy generation like binning, histogram analysis, cluster analysis and entropy based discretization (the first three presented earlier) [Jai01].

2.1.2 Data Mining

Data mining is an essential step in process of knowledge discovery data where AI methods are applied in order to extract patterns (rules) from data. In the data mining step the user

communicates with the data mining system using a set of *data mining primitives* designed in order to facilitate efficient and fruitful knowledge discovery. Those primitives include the database portion or the data set specifications in which the user is interested and the kinds of knowledge to be mined. A *data mining query language* is designed to incorporate these primitives, allowing users to flexibly interact with the data mining systems. A data mining task can be specified in the form of a data mining query, which is the input of the data mining system.

Data mining can be classified into *descriptive data mining* and *predictive data mining*. Concept description is based on descriptive data mining and describes a given set of task relevant data in a concise and summative manner, presenting interesting general properties of the data. Concept description consists in characterization and comparison or discrimination. There are two general approaches to concept characterization: the data cube on-line analytical processing approach and the attribute oriented induction approach. Both are approaches based on attribute or dimension.

For a data mining system there are some primitives like task relevant data, the kind of knowledge to be mined, background knowledge, interestingness measures, and presentation and visualization of discovered patterns [Jai01], [Cha00].

Task relevant data – specifies a portion of the database that is investigated, because usually the user is interested only in a subset of the dataset. The difficult task for this step is to specify the relevant attributes or dimensions involved in the problem. Users have only a rough idea of what the interesting attributes for exploration might be.

The kind of knowledge to be mined – specifies the data mining functions to be performed, such as characterization, discrimination, association, classification, clustering, and evolution analysis. In this step the user can specify and provide more pattern templates that all discovered patterns must match. These templates can be used to guide the discovery process.

Background knowledge – some knowledge about the domain to be mined is specified for guiding the knowledge process of discovery. A powerful form of background knowledge is known as *concept hierarchies*. They allow the discovery of knowledge at multiple levels of abstraction. This is represented like a set of nodes organized in a tree, where each node represents a concept.

Interestingness measure – is used to separate uninteresting pattern from knowledge. It is used to guide the process of mining or to evaluate the discovered patterns. This can reduce substantially the number of patterns, typically only a small fraction of these patterns will actually be interesting to the given user. There are some objective measures of pattern interestingness. Some are based on the structure of patterns and some are based of the statistic underlying them. Generally, each measure is associated with a threshold that can be controlled by the user. One measure is *simplicity*. This can be viewed as a function of the pattern structure, defined in terms of pattern size in bits, or the number of attributes or operations appearing in the pattern. Another measure is *certainty* where each discovered pattern has a measure of certainty associated with it that assesses the validity or "trustworthiness" of the pattern. A measure of certainty for the rules "A=>B", is *confidence*, where A and B are sets of items:

$$\text{confidence}(A \Rightarrow B) = \frac{\#\_\text{tuples\_containing\_both\_A\_and\_B}}{\#\_\text{tuples\_containing\_A}} \tag{2.6}$$

Another measure is *utility.* It measures the potential usefulness or interestingness of the pattern. It can be estimated by a utility function, such as support. The support associated to a pattern refers to the percentage of task-relevant data tuples for which the pattern is true:

$$\text{support}(A \Rightarrow B) = \frac{\#\_\text{tuples\_containing\_both\_A\_and\_B}}{\text{total\_}\#\_\text{of\_tuples}} \tag{2.7}$$

The last measure of interestingness presented here is the *novelty*. The novelty bringing patterns are those that contribute with new information or increase performance to the given pattern set. For example data exception may be considered novelty in that it differs from the data expected based on the statistical model or user belief. Another strategy in detecting novelty is to remove redundant patterns (a discovered rule can be implied by another rule that is already in the knowledge based or in the derived rule set).

Presentation and visualization of discovered patterns – is the form in which the discovered patterns are to be displayed. The visualization of discovered patterns in various forms can help users with different backgrounds to identify patterns of interest and to interact or guide the system in future discovery.

There are some algorithms of data analysis grouped in four categories. They are grouped according to the idea of the algorithm used for mining: *mining association rules, classification, prediction,* and *clustering*.

2.1.3 Mining Association Rules

Mining association rules consists of first finding frequent item-set (set of items such as A or B, that satisfy a minimum support threshold, or percentage of task-relevant tuples), from which strong association rules in the form of A =>B are generated. These rules also satisfy a minimum confidence threshold. Association rules can be classified into several categories based on different criteria. The association rules can be classified by the type of values handled in the rule into Boolean or quantitative. The association rules can also be classified by dimension into single dimension or multidimensional. By the level of abstraction the association rules can be classified into single level or multilevel (where different attributed are at different levels of abstraction). For mining association rules there are some classical algorithms like *Apriori, Frequent Pattern Growth, Multilevel Association Rules,* and *Constraint Based Rule Mining* (for details see [Jai01]). A lot of these algorithms are explained in details in my first PhD report [Mor05_1].

2.1.4 Classification and Prediction

Classification and prediction are two forms of supervised data analysis that can be used to extract models describing important data classes or to predict future data trends, being also an important step in data mining process. While classification predicts categorical labels, the prediction models continuous valued functions. Data classification is a two-step process. In the first step a model is built, based on a predetermined set of data classes or concepts, by analyzing database samples described by the attributes. Each sample is assumed to belong to a predefined class. Since the class label of each training sample is provided this step is also known as supervised learning. The learned model is represented as classification rules, decision trees, et al. In the second step the model is used for classification. First the predictive accuracy of the model is estimated. The accuracy of a model on a given test set is the percentage of test set samples that are correctly classified by the model. If the accuracy of the model is considered acceptable, the model can be used to classify future data samples for which the class label is not known. Prediction can be viewed as the construction and usage of a model to assess the class of an unlabeled sample, or to assess the value or value ranges of an attribute that a given sample is likely to have. A commonly accepted view in data mining is to refer to the use of prediction to predicted class labels as *classification,* and the use of prediction to predict continuous values as *prediction*. There are five criteria for the evaluation of classification and prediction methods like *predictive accuracy* (refers to the ability of the model to correctly predict the class label of new or previously unseen data),

computational speed (refers to the computational costs involved in generating and using the model), *robustness* (the ability of the model to make correct prediction given noisy data or data with missing values), *scalability* (refers to the ability to construct the model efficiently given large amounts of data) and *interpretability* (refers to the level of understanding and insight that is provided by the model). There are some common algorithms used for data classification and prediction [Ian00].

ID3 and C4.5 are the two greedy algorithms for the induction of the decision trees. Each algorithm uses theoretically an information measure to select the attribute tested for each non-leaf node in the tree. Pruning algorithms attempt to improve accuracy by removing tree branches reflecting noise in the data. Decision trees can be easily converted to classification IF-THEN rules.

Naive Bayesian classification and Bayesian belief networks - are based on Bayesian theorem of posterior probability. Like in naive Bayesian classification (which assume class conditional independence), Bayesian belief networks allow class conditional independencies to be defined between subset of variables.

Backpropagation [Mit97] is a neural network algorithm for classification using supervised learning that employs a method of gradient descent. It searches for a set of weights that can model the data so as to minimize the mean squares distance between the network's class prediction and the actual class label of data samples. Rules may be extracted from trained neural networks in order to help improve the interpretability of the learned network.

Nearest neighbor classifier and case based reasoning classifier are methods based on instances of classification in that they store all of the training samples in the pattern space. In genetic algorithms, populations of rules "evolve" via operations of crossover and mutation until all rules within a population satisfy classes that are not distinguishable based on the available attributes. Fuzzy set approaches replace "brittle" threshold cutoffs for continuous valued attributes with degree of membership functions.

Linear, nonlinear, and generalized linear models of regression can be used only for prediction. In linear regression, data are modeled using a straight line. Linear regression models a random variable, Y (called a response variable), as a linear function on another random variable, X (called a predictor variable). Many nonlinear problems can be converted to linear problems by performing transformations on the predictor variables.

Stratified k fold cross-validation is a recommended method for estimating classifier accuracy. Bagging and boosting methods can be used to increase overall classification accuracy by learning and combining a series of individual classifiers. Sensitivity, specificity, and precision are useful alternatives to the accuracy measure, particularly when the main class of interest is in the minority.

2.1.5 Clustering

Clustering is the unsupervised process of grouping the data into classes (or clusters) so that objects within a class (cluster) have high similarity, but are very dissimilar in comparison with the objects from other classes (clusters). Dissimilarity is assessed based on the attribute values describing the objects. Cluster analysis can be used as a standalone data mining tool to gain insight into the data distribution, or serve as a preprocessing step for other data mining algorithms in the detection of clusters. Clustering is a dynamic field of research in data mining. Many clustering algorithms have been developed. These can be categorized into *partitioning methods, hierarchical methods, density based methods, grid-based methods and model-based methods* [Jai01].

A *partitioning method* first creates an initial set of *k* partitions, where *k* is the arbitrary number of partitions to be constructed; then it uses an iterative relocation technique that attempts to improve

the partitioning by moving objects from one group to another. On the other hand the *hierarchical method* creates a hierarchical decomposition of given set of data. The method can be classified as being either agglomerative (button-up) or divisive (top-down), based on how the hierarchical decomposition is formed. To compensate for the rigidity of merge or split, the quality of hierarchical agglomeration can be improved by analyzing object linkages at each hierarchical partition or by integrating other clustering techniques, such as iterative relocation. A *density based method* clusters objects is based on the notion of density. It either grows clusters according to the density of the neighborhood objects or according to some density function. A *grid based method* first quantifies the object space into a finite number of cells that form a grid structure, and then performs clustering on the grid structure. A *model based method* creates a model for each of the clusters and finds the best fit of the data to that model. Typically model based methods involve statistical approaches or neural network approaches (such as competitive learning and self organizing maps – Kohonen maps [Koh95]). For more details related to there clustering methods also can see [Jai01], [Ian00].

2.2 Text Mining

As I presented by now data mining is about looking for patterns in a database that is considered to be structured data. In reality a substantial portion of the available information is stored in text, which consists of large collections of documents from various sources, such as news articles, research papers, books, digital libraries, e-mail messages and Web pages. Data stored in text format is considered semi-structured data in that they are neither completely unstructured nor completely structured, because a document may contain a few structured fields such as title, authors, publication, data, etc. Unfortunately these fields usually are not filled in. The majority of people don't loose time to complete these fields. Some researchers suggest that the information needs to be organized during the creation time, using some planning rules. This is pointless because most people will not respect them. This is considered a characteristic of unordered egalitarianism of Internet [Der00]. Any attempt to apply the same organizing rules will determine the users to leave. The result for the time being is that most information needs to be organized after it was generated. For this, searching and organizing tools need to work together.

From enormous quantity of information, only a small fraction will be relevant to the user. Thus, users need tools to be able to compare different documents, rank the importance and relevance of the documents, or find patterns and trends across multiple documents. Without knowing what could be in the documents, it is difficult to formulate effective queries for analyzing and extracting useful information from data. Thus, text mining has become an increasingly popular and essential theme in data mining.

2.2.1 Analyzing Text Data and Information Retrieval

Information retrieval (IR) is a field developed in parallel with database systems. Information retrieval is concerned with the organization and retrieval of information from a large number of text-based documents. A typical information retrieval problem is to locate relevant documents based on user input, such as keywords or example documents. Usually information retrieval systems include on-line library catalog systems and on-line document management systems. Since information retrieval and database systems each handle different kinds of data, there are some database system problems that are usually not present in information retrieval systems such as concurrency control, recovery, transaction and management. There are also some common information retrieval problems that are usually not encountered in traditional database systems,

such as unstructured documents, approximate search based on keywords and the notion of relevance.

2.2.1.1 Basic Measures for Text Retrieval

May [Relevant] be the set of documents relevant to a query and [Retrieved] be the set of documents retrieved. The set of documents that are both relevant and retrieved is denoted by $[\text{Relevant}] \cap [\text{Retrieved}]$. There are tow basic measures for assessing the quality of text retrieval:

- *Precision*: is the percentage of retrieved documents that are in fact relevant to a query. It is defined as follows:

$$precision = \frac{|\{\text{Relevant}\} \cap \{\text{Retrieved}\}|}{|\{\text{Retrieved}\}|} \tag{2.8}$$

- *Recall*: is the percentage of documents that are relevant to the query and were in fact retrieved.

$$recall = \frac{|\{\text{Relevant}\} \cap \{\text{Retrieved}\}|}{|\{\text{Relevant}\}|} \tag{2.9}$$

Precision ranges from 1 (all retrieved documents are relevant) to 0 (none of relevant document is retrieved). *Recall* range from 1 (all relevant documents are retrieved) to 0 (none of retrieved document is relevant). In fact *precision* represents a quantitative measure of the information retrieval system while *recall* represents a qualitative measure of this system.

2.2.1.2 Keyword-Based and Similarity-Based Retrieval

Most information retrieval systems support *keyword-based* and *similarity-based* retrieval. In keyword-based information retrieval, a document is represented by a string, which can be identified by a set of keywords. A user provides a keyword or an expression formed out of a set of keywords, such as "car and repair shop". A good information retrieval system needs to consider synonyms when answering such query. This is a simple model that can encounter two difficulties: (1) the *synonyms* problem, keywords may not appear in the document, even though the document is closely related to the keywords; (2) the *polysemy* problem: the same keyword may mean different things in different contexts.

The information retrieval system based on similarity finds similar documents based on a set of common keywords. The output for this system is based on the degree of relevance measured based on using keywords closeness and the relative frequency of the keywords. In some cases it is difficult to give a precise measure of the relevance between keyword sets. In modern information retrieval systems, keywords for document representation are automatically extracted from the document. This system often associates a stoplist with the set of documents. A stoplist is a set of words that are deemed "irrelevant" and can vary when the document set varies. Another problem that appears is *stemming*. A group of different words may share the same word stem. A text retrieval system needs to identify groups of words where the words in a group are small syntactic variants of one another, and collect only the common word stem per group.

Let's consider a set of d documents and a set of t terms for modeling information retrieval. We can model each of the documents as a vector v in the t dimensional space $\mathbb{R}^t$. The i^{th} coordinate of $v()$ is a number that measures the association of the i^{th} term with respect to the given document: it is generally defined as 0 if the document does not contain the term, and nonzero otherwise. The element from a $v()$ vector, v_i can indicate the frequency of the term in the document and there are a

lot of methods to define frequency of the terms. Similar documents are expected to have similar relative term frequency, and we can measure the similarity among a set of documents or between a document and a query. There are many metrics for measuring the document similarity. The used one is the Euclidean distance but the most used is cosine similarity defined as:

$$sim(v_1, v_2) = \frac{v_1 \cdot v_2}{\|v_1\| \|v_2\|} \tag{2.10}$$

where $v_1 \cdot v_2$ is the standard dot products defined as $\sum_{i=1}^{t} v_{1i} v_{2i}$ and ||v1|| is defined as $\|v_1\| = \sqrt{v_1 \cdot v_1}$

The similarity ranges from 1 (perfectly similar) by 0 (orthogonal) to -1 (dissimilar). Great values of similarity represent a small angle between vectors and therefore the vectors (the documents) are similar.

For defining the frequency of the terms (term-weighing) for the nonzero entries in such a vector are also many methods used [Mit97]. For example, it can be simply defined $v_i = 1$ as long as the i^{th} term occurs in the document, or let v_i be the term frequency, or normalized term frequency. Term frequency is the number of occurrences of the i^{th} term in the document. Normalized term frequency is term frequency divided by the total number of occurrences of all terms in the document. I have tested various term-weights in my application. Once the terms representing the documents and their weights are determined, we can form a document-term matrix for the entire set of documents. This matrix can be used to compute the pair-wise dependences of terms. A reasonable measure to compute those dependences is *odds ration* [Mla99], *information gain*, *mutual information* [Mla98] [Mla99], or *support vector machine* [Mla04]. There are some problems in determining term dependences based on document representation [Bha00]:

- *Too many terms in description.* The total number of distinct terms is quite large even for a small collection. A large ratio of these terms, when intellectually evaluated, seems irrelevant for the description of the document;
- *Inability to expand query with good quality terms.* It is well known that good quality terms are those that are more prevalent in relevant documents. Dependence base on all documents may not help in adding good quality terms to the query;
- *Mismatch between query and document terms.* Usually users' vocabulary differs from that of authors or indexes. This leads in some query terms not being assigned to any of the documents. Such terms will not form a node in the dependence tree construct by the earlier studies.

Each document from the set can be represented as a vector. Using similarity we can construct a similarity-based index, and when a query occurs, it is represented like a vector. This vector is used to search for their nearest neighbors in a document collection. The problem in this representation is that usually the dimension of the set of documents and the number of terms is quite large and leads to the problem of inefficient computation. On the other hand high dimensionality leads to a very sparse vector and increases the difficulty to detecting relationships between terms. A method called *latent semantic indexing* was developed to reduce the size of the frequency table for analysis.

2.2.1.3 Latent Semantic Indexing

It is used for reducing the size of frequency matrix. This method uses *singular value decomposition (SVD)*, a well-known technique in matrix theory. Given a t × d term frequency matrix representing *t* terms and *d* documents, the SVD method removes columns to reduce the

matrix to size k × d, where *k* is usually taken to be around a few hundred for large document collections. To minimize the amount of information loss, only the least significant parts of the frequency matrix are omitted. The latent semantic indexing method consists of the following basic steps:

- Create a term frequency matrix, called frequency-matrix. Compute singular value decomposition of *frequency-matrix* by splitting the matrix into three smaller matrices, *U, S, V*, where U and V are orthogonal matrices ($U^TU = I$ and $V^TV = I$), and S is a diagonal matrix of singular values. Matrix U have size *t* x *k*, the V matrix have size *k* x *d*, and matrix S is of size *k* ×*k*;
- For each document *d*, replace its original document vector by a new one that excludes the terms eliminated during SVD;
- Store the set of all vectors, and create indexes for them using advanced multidimensional indexing techniques.

Most matrixes that represent a set of documents are sparse matrixes, having most of elements equal to 0. A better representation in the memory of those matrixes is to keep only the values there are greater than 0 and the position of those values for each line of the matrix. This allows us to store the frequency-matrix without lose (for details see [SparseMatrix]).

There are other techniques for text retrieval like invert index or signature files. The *invert index* technique is an index structure that maintains two hash indexed or B+ tree indexed tables: *document-table* and *term-table*. The document-table contains a set of document records, each containing two fields: *doc-id* and *posting-list* (list of terms) that occur in the document, sorted according to some relevance measure. The term-table contains a set of term records, each containing two fields: *term-id* and *posting-list* (specifies a list of document identifiers in which the term appears).

The *signature file* is a file that stores a signature record for each document in the database. Each signature has a fixed size of *b* bits representing terms. Each bit of a document signature is initialized to 0. A bit is set to 1 if the term it represents appears in the document. A signature S_1 matches another signature S_2 if each bit that is set in signature S_2 is also set in S_1.

2.2.2 Document Classification Analysis

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.

Usually the key phrases do not come from a fixed vocabulary, but are likely to be phrases that occur in the text of the document itself. Maybe only a smaller training set is available – after all, assigning phrases to documents involves expensive intellectual labor. This technique is inapplicable because it can only form models for the key phrases that are used in the training data. But instead the key phrases could be chosen from the text itself. Given a document, rudimentary lexical techniques based on punctuation and common words could be used to extract a set of candidate key phrases. Then, features could be computed for each phrase, like how often it appears in that document, how close to the beginning of the document it first occurs, how often it has been used as a key phrase in other training documents, whether it occurs in the title, abstract or section headings, whether it occurs in the title of the paper cited in the reference list, and so on.

2.3 Web Mining

Web is a huge service which offers a lot of information [Law99]. The Web also contains a rich and dynamic collection of hyperlink information and Web page access and usage information, providing rich sources for data mining. The mining of Web pages is so different from mining of text documents. The Web is too huge and is still growing rapidly, the information stored on the Web is continuously updated and the Web pages contain far more authoring style and content variations than any set of books or other traditional text based documents. Another problem for Web mining is that the Web serves a diversity of user communities, and the users may have very different backgrounds, interests, and usage purposes. When users want to find something on the Web only a small fraction of the information on the Web is relevant or useful to the current user. These challenges have promoted research into efficient discovery and use of resources on the Internet. There are many index-based Web engines that search the Web, index pages, build and store huge keyword-based indexes. These indexes help locate sets of Web pages containing certain keywords. Thus an experienced user can be able to quickly locate documents by providing a set of keywords and phrases. As I previously pointed out, this type of search engines based on keywords suffers from several deficiencies. One is because the topic can contain a huge number of document entries returned by the search engine. Many of these pages can have low relevance according to the user's needs, or can contain a poor quality. Another deficiency is that many relevant pages for the user's search don't contain the keywords defining term. This is referring to as the polysemy problem, presented earlier in the text mining section. The Web mining is the most challenging task that searches for Web access patterns, Web structures and the regularity and dynamics of Web contents.

2.3.1 Automatic Classification on Web Documents

In the automatic classification of Web documents, each document is assigned to a class label from a set of predefined topic categories, based on a set of examples of pre-classified documents. For example Yahoo taxonomy from the net and its associated documents can be used as training and test sets in order to construct a Web document classification scheme and this scheme can be used after that for classifying new Web documents by assigning categories from the same taxonomy. This method might be useful for supervised learning classification.

The method for classifying documents using words from the document can be used for classifying Web documents. This scheme for classifying based on terms obtains good results for Web

documents classifying. Because hyperlink contain a solution of good quality from the semantic point of view in comparison with the topic of the page, it is better to use this semantic information in order to achieve even better accuracy than pure keywords based classification. However, since the hyperlink surrounding a document may be quite noisy, naive use of terms in a documents hyperlink neighborhood can even degrade accuracy of classifying.

2.3.2 Web Mining Categories

Data mining consist of three steps: *preprocessing data, pattern discovery and pattern analysis*. Similarly, Web mining has three parts which can contain the above three steps:

- *Web content mining* – text mining for the Web page content (the real data from the Web pages) and metadata given by tags in the html files;
- *Web structure mining* - data which describes the organization of the content and of the site;
- *Web usage mining* – data that describes the pattern of link usage on Web pages.

Alternatively, the structure mining of the Web can be treated like pieces of Web's content mining, for this, mining the Web can instead be simply classified in content mining and usage mining.

Web mining is the application of data mining techniques to the content, structure, and usage of Web resources. This can help to discover global, as well as local structure within and between Web pages. Like other data mining applications, Web mining can profit from given structure of data, but it can also be applied to semi-structured or unstructured data like free-form text. This means that Web mining is an invaluable help in the transformation from human understandable content to machine understandable one.

2.3.2.1 Web Content Mining

Web content mining is a form of text mining. The primary resource of the Web is the individual page. Web content mining can take advantage of the semi or structured nature of Web pages text. For this there are more techniques for information retrieval from text documents, like methods for indexing a text that were developed to work with unstructured (semi-structured) documents. Some traditional techniques for information retrieval became inadequate for an extensive amount of data. Without knowing what is in the document it is difficult to formulate querying for analyzing and extracting interesting information. The user needs tools for comparing different documents. Some important tools are rank and relevance of the document or finding patterns and direction for more documents. Web content mining describes the discovery of useful information from the Web contents/data/documents. Some of the Web content data are hidden data, which cannot be indexed. These data are either generated dynamically as a result of queries and reside in the database or are private. The Web already contains many kinds and types of data such as textual, image, audio, video, metadata as well as hyperlinks. Thus we could consider multimedia data mining as an instance of Web content mining. The Web content data consists of unstructured data such as free texts, semi-structured data such as HTML documents, and more structured data such as data in the tables or databases generated by HTML. According to [Ray00], we could differentiate the research done in Web content mining from two different points of view: *information retrieval* and *database* views.

The goal of Web content mining from the information retrieval view is mainly to assist or to improve the information finding or filtering through the users inferred or solicited profiles. Most of the applications use sets of words (in literature referred as "bags of words" [Cha00]) to represent

unstructured documents. The bag of words or vector representation takes single words found in the training set as features. This representation ignores the sequences in which the words occur and is based on the statistics about single words isolation. The feature could be Boolean (a word either occurs or not in a document), or frequency based (frequency of the word in a document). The features could be reduced further by applying some other feature selection techniques, such as information gain, mutual information, cross entropy, odds ratio [Mla99], χ^2 statistic or Term strength [Yan97].

Other preprocessing includes Latent Semantic Indexing as I briefly presented in paragraph 2.2.1.3 [Dee90] [Hof99]. The preprocessing variations are useful for reducing feature set size. In general they are highly effective over different domains for text categorization. Other feature representations are also possible such as: using information about word position in the document, using *n*-grams representation (word sequences of length up to n), using phrases, using document concept categories, using terms, using hypernyms, etc. Currently the term text mining has been used to describe different applications such as text categorization, text clustering, empirical computational linguistic tasks, exploratory data analysis, finding patterns in text databases, finding sequential patterns in texts, and association discovery. Information retrieval for semi-structured documents is due to the additional structural (HTML) information in the hypertext documents. Actually all of the works surveyed use the hyperlinks structure between the documents for document representation. The methods that are used are common data mining methods.

The database techniques on the Web are related to the problems of managing and querying the information on the Web. This was presented in the data mining section.

2.3.2.2 Web Structure Mining

Web structure mining usually operates on the hyperlink structure of Web pages. The primary Web resource that is being mined is the set of pages, ranging from a single Web site to the Web as a whole. Web structure mining explores the additional information that is contained in the structure of hypertext. An important application area is the identification of the relative relevance of different pages that appear equally pertinent when analyzed with respect to their content in isolation. The Web structure mining can be used to find authoritative Web pages and identify hubs (presented summary further in section 2.3.2.2.3) or see details in [Cha03]. According to Cooley, Mobasher and Srivastava [Coo99] there are five types of Web pages:

- *Head* pages that are entry points to a site;
- *navigation* pages that contain many links and poor content in information;
- *Content* pages that contain a small number of links but are rich in information;
- *Look-up* pages that have many incoming links, few outgoing ones and no significant content, such as pages used to provide a definition or acronym expansion;
- *Personal* pages that have diverse characteristics and no significant traffic;

Web structure mining tries to discover the model underlying the link structure of the Web. The model is based on the topology of the hyperlinks with or without the description of the links. The model can be used to categorize Web pages and is useful to generate information such as the similarity and the relationship between different Web sites. Web structure mining could be used to discover authoritative sites regarding the subject and overview sites regarding the subject, sites that point to many authoritative pages (hubs). This line of research is inspired by the study of social networks and citations analysis. With social network analysis we could discover specified types of pages (such hubs, authorities) based on the in-coming and out-coming links. Web structure mining uses the hyperlinks structure of the Web to apply social network analysis to

model the underlying links structure of the Web itself. Some research uses the network analysis to model the network of artificial intelligence research. They use the name of the entry data found in the close proximity of any public Web pages such as the hyperlinks from home pages, co-authorship and citation of pages, exchange of information between individuals found in net-news archives, and organization charts. Some algorithms have been proposed to model the Web topology such as *Hyperlink Induced Topic Search* and *PageRank*. These models are mainly applied as a method for computing the quality rank or relevance of each Web page.

2.3.2.2.1 Mining the Web Links

This is done in order to identify the authoritative Web pages mining relevant and high quality pages related to a given topic. The Web consists not only of pages, but also of hyperlinks pointing from one page to another. These hyperlinks contain an enormous amount of latent human annotation that can help us to automatically infer the notation of authoritative. When an author of a Web page creates a hyperlink pointing to other Web pages, this can be considered the author's endorsement of the other page. The collective endorsement of a given page by different authors on the Web may indicate the importance of the page and may naturally lead to the discovery of authoritative Web pages. Therefore, the tremendous amount of Web linkage information provides rich information about the relevance, the quality, and the structure of the Web's contents, and thus is a rich source for Web mining. The problem is that not every hyperlink represents the endorsement we seek. Some links are creating for other purposes, such as for navigation or for paid advertisement. Yet overall, if the majority of hyperlinks are for endorsement, the collective opinion will still dominate. Other problem is commercial or competitive interests, one authority will seldom have its Web page point to its rival authorities in the same field.

2.3.2.2.2 Page Rank

Page rank is used to discover the most "important" pages on the Web. In PageRank, each page on the Web has a measure of prestige that is independent of any information that we needed or current query. Roughly speaking, the prestige of a page is proportional to the sum of the prestige scores of pages linking to it. In this method all measures are defined recursively: the prestige of a node depends on the prestige of other nodes, and the measure of being a good hub depends on how good neighboring nodes are as authorities (and vice versa). This procedure involves computing eigenvectors for the adjacency matrix, or a matrix derived thereof, of the Web or a suitably relevant sub graph of the Web.

Let's assume for the moment that the Web graph is strongly connected – that is, from any node *u* there is a directed path to a node *v* (ergodic graph). If one wanders on the Web for infinite time, following a random link out of each page with probability *1-p* and jumps to a random Web page with probability *p*, then different pages will be visited at different rates; popular pages with many in-links (link to that page) will tend to be visited more often. This measure of popularity is called PageRank, defined recursively as:

$$PageRank(v) = \frac{p}{N} + (1-p)\sum_{u \to v} \frac{PageRank(u)}{OutDegree(u)} \tag{2.11}$$

where '$\to$' means "link to" and N is the total number of nodes in the Web graph and *OutDegree(u)* is the numbers of out-links from page *u* (links to other pages). The Google search engine simulates such a random walk on the Web graph in order to estimate PageRank, which is used as a score in pre-computed independent of the query. Hence Google can be potentially as fast as any relevance ranking search engine. Note that the PageRank is independent of any query or textual content. When a query is submitted, a text index is used to first make a selection of possible response

pages. Then an undisclosed ranking scheme that combines PageRank with textual match is used to produce a final ordering of response URL's (Uniform Resource Locator). The strongest criticism of PageRank is that it defines prestige via a single random walk uninfluenced by a specific query or by a current user's profile. A related criticism is of the artificial decoupling between relevance and quality, and the ad hoc manner in which the two are brought together at query time, for the sake of efficiency.

2.3.2.2.3 Hubs

A hub is one page or a set of Web pages that provides selections of links to authorities. Hub pages may not be prominent themselves, or there may be few links pointing to them; however, they provide links to a collection of prominent sites on a common topic. Such pages could be lists of recommended links on individual home pages, such as recommended reference sites from a course home page, or professionally assembled resource list on commercial sites. Generally speaking, a good hub is a page that points to many good authorities; a good authority is a page pointed to by many good hubs. To find the authoritative pages there is an algorithm about how to use hubs, called HITS (Hyperlink-Induced Topic Search) which is presented below.

First it collects a starting set of, say, 200 pages from an index-based search engine. These pages form the root set. Since many of these pages are presumably relevant to the searched topic, some of them should contain links to most of the prominent authorities. Therefore, the root set can be expanded into a base set which includes all of the pages that the root-set pages link to, and all of the pages that link to a page in the root set, up to a designates size cutoff, such as 1000 to 5000 pages (to be included in the base set). After that a weight-propagation phase is initiated. This is an iterative process that determines numerical estimators of a hub and authority weights. Here the links between two pages with the same Web domain (e.g. sharing the same first level in their URL's) often serves as a navigation function and thus does not confer authority, such links are excluded from the weight-propagation analysis.

We first associate a nonnegative authority weight a_p and a nonnegative hub weight h_p with each page p in the base set, and initialize all a and h values with a uniform constant. The weight is normalized and an invariant is maintained that the squares of all weights sum to 1. The authority and hub weights are updated based on the following equations:

$$a_p = \sum_{(q \text{ such that } q \to p)} h_q \quad and \quad h_p = \sum_{(q \text{ such that } q \leftarrow p)} a_q \tag{2.12}$$

where "$\rightarrow$" means "link to".

The first equation implies that if a page is pointed to by many good hubs, its authority weight should increase. The second equation implies that if a page is pointing to many good authorities, its hub weight should increase. We write these equations in matrix form as follows. Let us number the pages {1, 2… n} and define their adjacency matrix A to be an $n \times n$ matrix where A(i,j) is 1 if page i links to page j, or 0 otherwise. Similarly, we define the authority weight vector $a=(a_1, a_2, \ldots, a_n)$ and the hub weight vector $h=(h_1, h_2, \ldots, h_n)$. Thus we have:

$$\begin{aligned} h &= A \cdot a \\ a &= A^T \cdot h \end{aligned} \tag{2.13}$$

where A^T is the transposition of matrix A. Unfolding these two equations k times, we have

$$\begin{aligned} h &= A \cdot a = AA^T h = (AA^T)h = (AA^T)^2 h = \ldots = (AA^T)^k h \\ a &= A^T \cdot h = A^T Aa = (A^T A)a = (A^T A)^2 a = \ldots = (A^T A)^k a \end{aligned} \tag{2.14}$$

According to linear algebra, these two sequences of iterations, when normalized, converge to the main eigenvectors of AA^T and A^TA, respectively. This also proves that the authority and hub weights are intrinsic features of linked pages collected, and are not influenced by the initial weight settings.

The Hyperlink Induced Topic Search (HITS) algorithm generates a list of pages with large hub weights and also list of pages with large authority weights for a given search topic. Many experiments have shown that Hyperlink Induced Topic Search provides surprisingly good search results for a wide range of queries. Some problems occur when hubs contain multiple topics, because this algorithm ignores textual contexts. Or also cause "topic hijacking" when many pages from a single Web site point to the same single popular site, giving the site too large a share of the authority weight. Such problems can be overcome by replacing the sum from the first equations with weighted sums, scaling down the weights of multiple links from within the same site. This is done using anchor text (the text surrounding hyperlink definition in Web pages).

2.3.2.3 Web Usage Mining

Web usage mining [Str00] mines the data derived from users' interaction with the Web. The Web usage data includes the data from Web server access logs, proxy server logs, browser logs, user profiles, registration data, user sessions or transactions, cookies, user queries, bookmark data, mouse clicks and scrolls, and any other data as the results of interactions. Web usage mining focused on techniques that could predict user behavior while the user interacts with the Web, mining the log files (IP address), cookies or path analysis. In Web usage mining resources that are mined are records of the requests made by visitors to a Web site, most often collected in Web server logs. The content and structure of Web pages, and in particular those of one Web site, reflect the intentions of those who have authored and designed those pages, and their underlying information architecture. The idea is to find a relationship that can be induced by usage where no particular structure was designed. Mining the visitors to that site, however, one may find that most of those users who were interested in product A were also interested in product B. Here "*interested*" may be measured by request for product description pages. The idea is to identify association rules between products (Web pages in this case). When a new user shows interest in product "A", he will receive also a recommendation for product "B".

The Web usage mining process could be classified into two commonly used approaches. The first approach maps the usage data of the Web server into relational tables before an adapted data mining technique is performed. The second approach uses the log data directly by using special pre-processing techniques. As this is true for typical data mining applications, here the issues of data quality and preprocessing are also very important. The typical problem is distinguishing among unique users and server sessions. In general typical data mining methods could be used to mine the log data after the data has been preprocessed to the desired form.

The applications of Web usage mining could be classified into two main categories: learning a user profile or user modeling in adaptive interfaces and learning user navigation patterns. Web usage mining would be interested in techniques that could learn necessary information about the needs and preferences. These are used to model the user's profile and combine it with Web content mining. On the other hand, information providers would be interested in techniques that could improve the effectiveness of the information on their Web sites by adapting the Web site design or by biasing the user's behavior towards satisfying the goals of the site. In other words, they are interested in learning user navigation patterns. Then the learned knowledge could be used for applications such as personalization (at a Web site level), system improvement, site modification, business intelligence and usage characterization.

2.3.2.3.1 Mining Log Files

This type of mining processes log records for extracting user's accessing pattern to Web pages. By analyzing and exploring regularity Web log records one can identify potential customers for e-commerce, increase quality and deliver information services to end users and increase performance of server system from the Web. Web server usually records every login (Weblog), for each access to Web pages. This contains URL, original IP from user log and request. This Weblog provides rich information about the dynamic of Web. This is important to develop new sophisticated techniques for mining the Weblog.

To develop techniques for mining the Web, we can consider the following. Firstly, it is very encouraging and exciting to imagine various applicable possibilities of Weblog file analysis. It is important to know if the success of each application depends on how much knowledge is discovered from the data log. Data logs need to be preprocessed and transformed in order to recover and analyze significant and useful information. Secondly, besides available URL's, time and user's IP, the information about Web pages content is important. A multidimensional visualization can be built using Weblog and multidimensional analysis. On-Line Analytical Processing (OLAP) can be performed to find the most accessed N pages, the latest most frequently accessed and anything else to help us discover relevant data. Thirdly, data mining can be used in Weblog records to find association patterns or sequential and oriented access to Web. Analyzing detailed Weblogs offers us the opportunity to take the necessary measures to obtain additional information from users. This additional information can include sequences about Web pages visited by the user from the buffer. Data Weblog provides us information about what user groups access what page groups. Weblog can be grouped with Web contents mining and link structure mining to help us mine the Web, classify Web documents, and build a multi-level structure of basic information.

An interesting problem from the producer's point of view is trying to replace nowadays Internet advertising with a new one, addressed only to users interested in these commercials and with low costs. This onset is already applied in advertising where the specific catalogue is created centered by a specific field that is sent only to interested people. For example if a user buys a software package from a company, he will receive new proposals from that company afterwards. Thus the problem for the entrepreneurs is to identify interested customer for their products. The products can be in these cases a very general item (a new job, software, merchandise). There are some ways to approach this problem. One way is to search on the Internet to find user's presentation pages and analyze those pages from the semantic point of view to identify if the user could be interested in that product. In case that the user could be interested he can be informed by sending the commercial for that product. However this approach is limited because in this moment there are not so many people that have complete personal pages on the net, and usually these pages are not updated. Another approach is trying to identify users based on what they do on the Web (inspecting queries, cookies, and server log files). Thus analyzing queries can give us information about categories of interest, at the moment, for the user. Analyzing server log files can give us the information about the IP of the users, pages visited on the server, and interests in those pages (by analyzing the time spent on the pages). This information can be mined to identify users that are now interested in the product.

2.3.3 Resource Discovery Systems

Even though interface and Web content designers [And04] included user behavior and some skills in some search engines users may still have difficulties performing Web searches:

- The user is unable to formulate the right query and restrict the result set. Using phrases with many words often produces no result. Users then prefer to specify only one or two words, which generates large sets of results.
- The user interface can be difficult to use or inaccessible for the unskilled or disabled user.
- The ranking function is applied statically; the user is not able to select the criteria most appropriated to him. Some options are present in advances searches, but are rarely applied by users.
- The information on the Internet is rarely structured and organized for fast retrieval by search engines. Web "designers" don't apply meta-tags such as description of keywords correctly and don't use meaningful filenames, titles, link descriptions and alternative texts. In addition, an inappropriate use of metadata produces the phenomena called "search engine spam", aimed at deviating search engine results. For this reason most search engine ignores or only partially uses metadata.

Besides the results of search engines, another important aspect is the usefulness of search tools. This aspect is often neglected. It is important to make this very accessible and usable by anyone, regardless of their physical condition or environments. Accessibility guarantees use to all, understandable and navigable content. Usability renders a more efficient and satisfactory Internet navigation. Many factories compose the user interface, like presented in [And04]:

- *Arrangement of components:* This point is very relevant because value-enhancing features are more "visible" when positioned in an area rapidly encountered by eye movement and do not require page scrolling. For example, the refinement functions of Google [Google], which allows searching into results it is not very obvious thanks to its position and font (size and color): it is found at the end of the results page, so inexperienced users may not benefit from them.
- *Number of elements:* Simplicity helps unskilled users navigate the interface easily. Web directories are organized according to categories of goods and services offered. Depending on the type of search it may be more appropriate to use a search engine or directory. On the other hand, their interface is quite full and can create confusion in an unskilled user who wishes to formulate a search query.
- *Expressive power:* A graphical representation can communicate certain kinds of information much more rapidly and effectively than other methods.
- *Functions*: A user typically performs a simple search and specifies one or more words, obtaining a large set of results. Further criteria selection can be specified to restrict search in the results. Preference and commands, although very powerful, are rarely used, even by skilled persons.
- *Clustering of results* allows users to explore results grouped by categories; in this way users can navigate a single branch of results more efficiently.

In the system presented earlier the search engine indexes Web pages into a very large database, and returns as response all matching links found in the database for a user query. This is a viable solution and it is used by most search engines, but it is not such a good solution because it needs, from the user's point of view, more time and more searching effort. This is due to the fact that the results to the query are hundreds or thousands of pages and it becomes very difficult for the user to search and find relevant information in the enormous amounts of information available. Another disadvantage for this type of search is that the results are ordered using page ranking that is applied statically, without any influence from the current user, and these criteria usually may not reflect the needs of the user. This situation occurs when a new, interesting and relevant document for the

current user is not placed in among the most interesting documents because it has a poor page rating. This situation makes it too difficult for the user to find it, or too time consuming for the user to find it. Another disadvantage is due to the fact that these types of search don't provide any information about the influence the keyword has in the query results and how does this keyword influence the joining of documents.

All search engines have provided the possibility to customize search, as "Advanced search" and "Preferences". For example the possibility to select the language of the searched pages, to specify exactly the words that need to be in the documents and the words that should not be in the documents, documents format and the date when they were created. The disadvantage is that they have a rigid structure and a fixed number of possibilities.

To solve the problems presented earlier this field of research has grown rapidly in areas such as algorithms, strategies and architectures. Hundreds of ideas were tried, some were implemented and some are only prototypes. Some of these ideas are:

- Organizing documents in predetermined categories. Usually these are named *Web directories* and have a static structure that is very difficult to update. (e.g. yahoo [Yahoo], aol [Aol], and excite [Excite])
- An upgrade of the existing search engines was tried by improving the way the results are presented. This was done by different methods of organizing the results:
  - *hierarchical representation* obtained by performing one or two levels on-fly clustering of the results returned by a classical search engine (e.g. WebCrawler [WebCr] for one-level representation and vivisimo [Vivisimo] for two-level representation)
  - *graphical representation* by a sequence of interactive maps of the results obtained from the classical search engine (e.g. Kartoo [Kartoo], and SurfWax [Surfwax])
- Programs (agents) that allow *monitoring a specified page* from Web and report when changes occur (e.g. DICA, Syskill & Weber and others).
- Programs (agents) that silently *observe the user's browser behavior* during the day and then use these training examples to learn user's profiles, and during the night searches for new pages that are likely to fit the learnt profile.
- Search engines that help users by suggesting synonyms or idioms of the words presented in the query and try to develop a more refined query to obtain better results. Literature calls this "*Query refinement*".
- Creating programs (agents) that work between the results of the search engine and the user by filtering the results presented by the former using the *user profile* learnt from the latter.

2.3.3.1 Web Directories

We can easily see that hyperlinks don't create a topic-oriented structure on the Web. To solve this problem and group the documents in some way there were created topic-structured directories which are called topic directories (Web directories). Some of these are *Open Directory Project* [Dmoz] and *Yahoo* [Yahoo]. This structure has been constructed through human effort, and resulted in a giant taxonomy of topic directories. Each directory contains a collection of hyperlinks relevant to the topic that are often the most popular or authoritative sites related to this specific topic. One may model such as tree-structured hierarchies which are considering the relations between topics and their generality such as specific topics or general topic. Following this idea a page is assigned to the best fitting topic for the content of that page. Although topic taxonomies

are a special case of semi-structured data, they are important and frequent enough to be mentioned separately. For example, the *dmoz* [Dmoz] directory that categorizes Web sites is created manually by about 52 thousand editors, according to Kummamuru [Kum04], and unfortunately covers less than 5% of the Web. At the end of 2006 DMOZ have 74719 editors and 590000 categories in more than 10 languages.

These directories are created by users and have a fixed structure and new documents are fitted in this structure making creation of new topics hierarchy difficult. This is a good idea because the structure of these directories is very eloquent and intuitive for human users but it is very difficult to implement and growth. Another disadvantage is that these types contain a limited number of pages in directories because the process of assigning new pages is difficult. This type of server accepts requests from users to put new pages in its structure. These pages can be put only into the existing hierarchical structure. This is based on users registering a page address and a short description of this page and server catalogues and puts this page somewhere in its hierarchical structure. Users can search in those directories or can open specific categories from this structure and they can access the documents from this structure. This was founded in the spirit of Open Source movement, and it permits each user, after a simple registration, to create and manage a subtopics category. This modification was made to simplify the development of Web categories and increase the coverage of the Web.

The search engines use agents to find new Web sites or Web sites that were modified. Web directories will list a site only if this site was registered by the Webmaster. Web directories are generally manually created by analyzing the received addresses of sites and than evaluates and catalogues these sites in their structure. Because the descriptions have a limited dimension, in almost all cases these contain only a short description of the content.

In comparison with agents of the search engines that analyze the title, meta tags, head and content of the site, Web directories evaluate only the title, short description and categories proposed for a site and don't take into account the content of the site and its meta tags. So, the Webmaster description for a specific page has a great influence on the content of the categories for Web directory.

Web directories are divided in categories and subcategories, ordered usually alphabetically for an easy scan. However the update of new sites is too difficult and needs much time for evaluating the structure and categorize it. The categories are influenced by the point of view of the team that developed them.

Many sites are based on Web directories. Some special sites, very used, are Yahoo (but this also has a search engine) and "search.aol" [Aol] that has many categories and subcategories and many pages indexed in those category. Another interesting site is SurfMind [Surfmind], even if it has only a small number of registered documents (2610 documents, this number grow in 2006 with more than 1800 documents) it has a different type of representation of the search results. Thus it can have a list with new sites and a list with most popular sites, or a list with all categories. For a specified category you can see all the articles and a short description for each document. You can also open the article directly in its own Web page with differed dimensions of representation without being necessary to open a new Web page. This useful idea of opening the article directly in its own Web page it's now used also in the more recently developed search engine like Vivisimo or Power-Search.

2.3.3.2 Search Results Representation

On important problem for almost all search engines is the presentation of the results. Usually when search engines return results they return more pages that contain links to the results. Some of these links are interesting for the user and some are not. It is very difficult for the user to rapidly

distinguish between the useful and the irrelevant pages. Another disadvantage is that the search engine returns a ranked list of Web pages as response to the user's search request. These Web pages with different topics and different aspects are mixed together in the returned list. The user has to shift through a long list to locate pages of interest. This method is highly inefficient since the number of retrieved search results can be in the thousands for typical queries. Most users just view the top results and therefore might miss relevant information. Moreover, the criteria used for ranking may not reflect the needs of the user. It is not specified in the returned list why this document is relevant to the user's query and what is the relationship between the returned documents. Because Web directory creation, as presented earlier, is very difficult and very time consuming from a human point of view, it is better to create and populate categories dynamically. After receiving results from the search engines, another program (agent) tries to analyze all the results and tries to classify them in dynamically generated categories. Then it presents this result to the user. Organizing Web search results into a hierarchy of topics and subtopics facilitates browsing the collection and locating more easily the interesting results. Almost all sites from this category do this automatically or "on-the-fly". These systems contain two main components: a text classifier that categorizes Web pages and a user interface that presents the Web pages within the category structure and allows the user to manipulate the structure view. These systems differ in the algorithms used for text classification and the types of user interfaces.

As far as the results are represented there are two main methods: hierarchical representation and graphical representation.

2.3.3.2.1 Hierarchical Representation of Search Results

There are systems that create a hierarchical structure for classifying a large, heterogeneous collection of Web content. Organizing search results in this format allows users to have a general image of the document categories and focuses on items in categories of interests rather than having to browse through all the results sequentially. This is one of the most used methods for representing the results of the search engine. In this category there are some sites already implemented (e.g. Vivisimo [Vivisimo], WebCrawler [WebCr]) and some are still in research because of the problem of text categorization is very difficult and very time consuming.

S. Dumais et al. in [Dum00] present an algorithm for automatic representation of the search engine results in two-level categories. This method wants to supplement human effort in creating structured knowledge for Web directories. The basic idea in their work is to use classification techniques to automatically organize search results into existing hierarchical structures. The model for classification was learned offline using a training set of human-labeled documents and Web categories. For classification the authors work with just short summary descriptions of the Web pages. This short description is generated automatically for each page and is used for evaluating the classification algorithm. The summaries consist of title, keyword and the description tag if it exists or the first 40 words from the body otherwise. With this technique the authors create a binary vector for each page that indicates whether or not a term appears in the page. For the training and tasting part of text classification the authors use Support Vector Machine [Sch02] algorithm because it is very fast and effective for text classification problems. The authors used pre-classified Web pages, taken from "LookSmart" [LookSmart] Web directory, for training. Because SVM algorithm solves the problem for two class classification, for multi-class classification the authors implement one versus the rest algorithm. For each test example, the authors compute the probability of being in each of the 13 first level categories and then in each of the 150 second level categories. To solve this problem the authors use a hierarchical decomposition of a classification problem that allows an efficient learning and representation data. For the training part the authors used a set of pre-classified Web pages that can be classified in one or more classes. Once the categories are learned, the results from any user query can be classified.

At query time, each page summary returned by the search engine is compared to the 13 first level category models. A page is placed into one or more categories, if it exceeds a pre-determined threshold for category membership. Pages are classified into second-level categories only for the first level categories that they belong to. This is done using the same procedure.

In another article Hao Chen et al. [Che00] present the application implemented using the algorithm presented earlier and studies the influence of representing the results of the search engine hierarchically versus typically ranked list interface. This study is done using predefined categories from LookSmart Web directory. In the training part the authors make an offline text classification with representative labeled samples of Web pages. At query time, new search results are quickly classified on the fly into the learned category structure.

In this PhD thesis I am presenting also a modality of classifying Web documents based on Support Vector Machine algorithm. In comparison with Dumais idea I will take in consideration all content of the Web document and the title of the document. I present results based on Web documents took from DMOZ Web categories and with more than two subcategories, but for a small number of categories. I analyzed Web pages only from feature selection influence point of view, not necessary for creating the Web categories. Obtained results are presented in section 4.6.4.

The categories can be ordered either in a static alphabetical order, or dynamically according to some importance scores. The advantage of dynamic ranking is to present the most likely category first. The disadvantage is that it prevents the user from establishing a mental model of the relative position of each category in the browser window. In the dynamical experiment the importance was determined by the number of pages in the category. The category with the most items in it was shown first, and so on. Into the category the pages are sorted by the probability that the current page is in this category. The categories used in their experiment were designed to cover the full range of Web content. Nonetheless, not all user queries will match the category structure to the same extent, some query may fall entirely. In this case the category interface is like in the list interface and has a "NotCategorized" group. The study showed that the category interface is superior both objectively and subjectively. Users liked the category interface much better than the list interface and they were 50% faster at finding information that was organized into categories. Organizing search results allows users to focus on items in categories of interest rather than having to browse through all the results sequentially.

Kummamuru et al. [Kum04] present an idea to build a hierarchical topic for a collection of search results retrieved as response to a query. At each level of the hierarchy the algorithm, named "DisCover", progressively identified topics in a way that maximized the coverage while maintaining distinctiveness of the topics. For evaluating the quality of the topic the authors used some objective measures like "coverage" and "reach time". The authors used this like a method for comparing this algorithm with another two monothetic algorithms for clustering. Automatic taxonomy generation algorithm based on how the documents are assigned to different cluster can be categorized into two classes:

- *Monothetic algorithms* in which a document is assigned to a cluster based on a single feature;
- *Polythetic algorithms* in which a document is assigned to the cluster based on multiple features;

The authors used monothetic algorithms because each cluster is described by a single feature or concept and all the documents presented in a cluster contain this feature. This leads to easier understanding of the cluster by users. The algorithm progressively identifies clusters and it tries to maximize the distinctiveness of the monothetic features describing the cluster while at the same time it maximizes the number of documents that can be described or covered by the monothetic features.

During this work the authors proposed some properties of taxonomies generated from a corpus of documents:

- *Document coverage* – ideally all the documents in the collection should be covered by the taxonomy. A document is said to be covered by taxonomy if it lies in at least one of the clusters in the top level of the taxonomy. This implies that the top level clusters should be chosen in such a way that they cover most of the documents in the corpus. It will be considered a good taxonomy generation algorithm, the algorithm that covers the most number of documents.
- *Compactness* – because the main purpose of the taxonomy is to summarize and provide a better browser experience, the taxonomy needs to be as compact as possible. The taxonomy can be too wide or too deep because the basic purpose may not be served.
- *Sibling cluster distinctiveness* – each of the nodes, especially those at the top level, represent a concept presented in the search results. At any level of the hierarchy this should be as different as possible from each other to maximize generality while browsing the documents in the hierarchy.
- *Node Label Productiveness* – a good taxonomy should help the user to find documents of interest with minimum effort. The labels of the node guide the user in locating a document in the taxonomy. Thus the node labels should be chosen such that they are good indicators of the documents contains.
- *Reach time* – is the average time needed to locate interesting search results. It is an important criterion for taxonomies that need to be quickly able to locate search results of interest within the hierarchy.
- *General to specific* - the node labels in the hierarchy are actually the concepts associated with the query. In the hierarchy, the most general concept should be associated with the root, and the node label of any node should be more specific (less general) than that of its parent and at the same time it should be less specific than those of its children. The generality or specificity of the node labels needs to be considered within the context of the query.

In the presented article the authors take into consideration only the first three proprieties. It is assumed that each document in the collection can be represented by a set of words. Each node in the hierarchy is associated with a concept and all documents under a node contain that concept. Of course, each of these documents will contain several other concepts as well. Monothetic clustering algorithm involves selecting a subset of concepts from all the concepts, optimal in some sense, and associating a child node with each of them. Thus the authors create a compact hierarchy that initially has a limited number of child nodes, but provides to the user the ability to progressively increase the number of child nodes of any node of interest.

For comparing and evaluating this hierarchy the authors performed a users study. This can demonstrate whether the hierarchies generated are actually helpful to real users for browsing. The authors performed the study using 17 volunteers, who were both technical and non-technical employees of the IBM Indian Research Laboratory, and 50 queries (25 ambiguous queries and 25 popular queries). Each volunteer needed to evaluate 3 queries except for one volunteer who evaluated 2 queries. The voluntaries needed to give 5 responses to each query, with a total of 25 responses for ambiguous queries and 25 responses for popular queries. Users were provided with a very short introduction to autonomic taxonomy generation algorithm and the motivation behind creating taxonomy from a document collection. The users were not specified which hierarchy corresponded to which algorithm. Each user needed to fill up an online questionnaire for each query. The questionnaire contained 6 questions. The users need to evaluate the obtained hierarchy using a scale from 1 to 10. These 6 questions were broadly divided into 3 groups. The first groups

of 3 questions pertain to the top level nodes of the hierarchy. The second group of 2 questions pertains to the second level nodes, and in the final question the user needed to give an overall rating to the hierarchy. For comparisons, the authors used the proposed algorithm "DisCover" and two others algorithms for hierarchical representations CAARD (Clustering Algorithm for Asymmetrically Related Data [Kum01]) and DSP (Dominating Set Problem [Law01]). For evaluation the authors used only 4 properties: document coverage, compactness, reach time and computational complexity. The results obtained show that "DisCover" is superior to DSP in almost every aspect and in comparison with CAARD is better in terms of summarizations for both popular and ambiguous queries.

2.3.3.2.2 Interactive Map for Representing the Search Results

An alternative to the representation of the search results is graphical user interface that allows easier navigation and exploring of these results. Kunz et al. present in [Chr02] an interactive matrix for showing the hyperlinks of retrieved sites. In this representation the user can see the results using one category resulting in a List Browser and two categories at the same time with the so called Matrix Browser. To represent search results the authors use a familiar and well known interactive tree widget. The tree widget provides the ability to display multiple items in a tabular format. The tree widget is used to display hierarchically organized data. It is a vertical container for widgets of tree type items. The tree widget provides geometry management of arbitrary widgets arranged in a directed, acyclic graph.

In general visualization and exploring search results structure still constitutes a major problem for user interface design, in terms of minimizing search results, viewing and supporting user's understanding and providing efficient interaction for exploring the search results. In the presented paper the authors combine the two ways of searching and exploring into the information space in a new graphical user interface for the search engine. Thus the results of the search engine are displayed in one interactive category tree or in an adjacency like system with two interactive trees as axis of a matrix. The first overview generated for the graphical user interface shows how many hits are founded in the hierarchically ordered categories. In the List Browser representation the results set is listed in a window, like a tree widget which allows the user to expand and collapse the categories and subcategories by clicking on the interactive symbols in front of the bar and the category name. The size of the bar represents the consolidated amount of search hits in the category and all subcategories. So the user gets an overview in which categories the search results are located, and thus can explore the results in a flexible manner. In the Matrix Browser the users have the possibility to navigate within these trees and explore the structured search results. They have different kinds of interactive symbols inside the matrix visualization. Circles represent how many sites are found in two categories and squares represent the site references itself. In the hierarchical categories system the users can increase or reduce the displayed amount of information and refine their query without input of any text data and they have the opportunity to view more details of the metadata structure together with the local sites.

Another type of representation of the search results is a graphical representation where all information is placed intro tree structure with connections between documents. For example the site "www.kartoo.com" is very interesting because of the way it represents the search results, and because all other things have total graphic representation (they present results in a interactive map). As many search engines it uses other search engines from the Web. It only graphically represents all the results. It allows the user to choose what search engine he wishes to be used for the search. The disadvantage is that they provide only a small number of results at the same time (in the graphical representation). The interesting part is that the authors represent the names of the documents and their connections together with the keywords that connect them, but this connection is not so relevant. The same application, have also a separate section for monitoring

specified sites (this part is explained in the next section). This site is interesting as a representation mode of the search results but information was not available regarding the techniques used for implementation because it is a commercial product in a developing stage.

In [Wiz04] Wiza et al. presets an innovating system for building the visualization of the search results in a 3D representation. The authors use for representation the X-VRML system (X - Virtual Realities Modeling Language). The authors use an interface selection which receives the search results from the most popular search engine and selects the best representation for those results. This idea is interesting as it can model the results in the most appropriate manner. This idea has not been found in the works presented above. The system, called Periscope [Periscope], can choose to represent the data into a 2D or 3D space; because some 2D maps representation can provide reasonable good results. The 2D representation is presented with a flat panel, limited in size, where information can be presented in patch forms. Sometimes it uses different textures and it is used when the number of object presented is low. For creating the 3D representation the authors take in consideration 3 elements:

- *User interaction* - navigation in the space and interaction with its contents;
- *User cognition* - the spatial representation data is closer manner to the humans perceive the surrounding world and permit the user to change the viewpoint to improve perception and understanding of observed data;
- *Information capacity* - the system can provide information in different shapes, colors, textures, positions, sizes, orientations and even behavior.

In comparison with 2D representation the 3D representation is not limited in size and space but only in user perception.

2.3.3.3 Monitoring Specified Pages

There are situations when users find the right place where the interesting information is placed but the information is not available for the moment and they need to revisit the site periodically to find new information. According to this in [Ack97_1] and [Ack97_2] is presented an agent that uses machine learning for detecting “interesting” changes in Web pages previously marked by the user to be relevant. Because this agent is based on the changes on known pages rather than finding new pages, it increases the likelihood that the founded information will be interesting. The authors called this agent DICA (Do I Care Agent). This agent has some major functions:

- Periodically it visits a user defined list of target pages;
- Identifies any changes since the last time it was visited;
- Analyzes pages and decides if the changes are interesting;
- Notifies the user if the changes are interesting;
- Accepts relevance feedback on the interestingness of the change;
- Facilitates information sharing and collaboration between individuals and groups;

Because the user provides feedback on whether change reasons are interesting or not and there is a limited list of Web pages the precision is greatly improved. For this the author use two features for improving the precision: percentage of retrieved items that are “interesting” and the percentage of potentially interesting items that are actually retrieved. The analysis of changes in the monitored pages is based on the fact that changes are usually incremental and new things are usually added in ways that are consistent with what has been done in the past. In order to detect whether a change is interesting or not the authors use Bayesian classification process and extract a set of compatible attributes and compute the probability that the object belongs to each of the known set of

categories. The agent notifies the user when the computed probability exceeds a configurable threshold. For classification the authors create a classification profile that is a list of feature values and the corresponding probability that documents having those value belong to one or more classes.

All interesting changes are sent to the user by e-mail and all changes are also listed in the agent's associated Web pages which users can browse at their leisure. Such as in these pages attribute values that define an interesting change are listed. These pages are also used for relevance feedback. In these pages the changes found by the agent in the monitored pages are sorted according to how interesting they are likely to be, with more interesting changes on top. All changes are initially marked "Unrated" and the user can change this to "I Care" or "I don't Care" and after that the agent learns again.

Because different pages may be interesting for different reasons, users may maintain multiple DICA agents specialized on particular topics or kinds of Web pages. For example, the criteria for an interesting journal announcement may be different from a news item. An agent can receive changes from another agent and can decide if those changes are interesting or not from its point of view. For more details on Agent's Paradigm and detailed information on Agents see [Bar02].

Other two agents that learn user profiles are presented in [Ack07_2]. First "Syskill & Webert" is an agent that is designed to help users with long term information seeking goals, such as finding previously unvisited Web sites on a particular technical topic. The authors use a simple Bayesian classifier to create a user profile from the user's preferences. The profile is used afterwards to compute the probability that a Web page is interesting to the user or not. For learning the user profile the authors use feedback on the interestingness and usefulness of the Web pages. The authors developed an interface where they present the results and the user can click on "Hot" or "Cold" buttons to specify if he is or isn't interested in this document. When creating a query for the search engine the agent uses two types of words. The first query contains the words that occur in the most number of pages that have been rated "hot" and the second contains words whose presence helps discriminate pages that are rather hot then cold using mutual information. Because the authors use a Lycos search engine that doesn't accept a very long query they use the seven most informative words that are found in a higher proportion of hot pages and the seven most commonly words occurring as cold. The agent retrieves each returned page and analyzes the HTML files to produce its recommendations. The agent displays its recommendations by annotating links that indicate whether the user is likely to be interested in the page and a number indicating the probability that a page is interesting.

The second agent presented by Ackerman [Ack97_1], [Ack97_3] is named "GrantLearner" that is an agent that notifies an individual of new research grant opportunities that meet the learned profile of the individual's research interests. This agent learns to distinguish between interesting and uninteresting funding opportunities based a user's rating of these descriptions. This agent is connected to a grant database maintained at UCI (University of California, Irvine) and provides a user interface to browse it. The system displays a list with all grant subjects and a description of the grant. Using "hot" and "cold" buttons the user can indicate whether a grant is related to his interests. After that when the user provides information about 10 interesting grants the agent creates the profile of the user interests and then processes all database using this profile providing a new list sorted by relevance for the user. For this the agent uses the same Bayesian classification method as for learning the user's probabilistic profile.

2.3.3.4 User's Browser Behavior

In the last years a common idea used for increasing the quality of the search results is creating a user's profile based on user's interests. Then, this profile is used to improve search results or to

develop the query to obtain better results. In this sense Goecks and Shavilk in [Geo99] present an agent that learns a user browser behavior and tries to establish a user profile using this information. Almost all types of research was focused on learning a user profile based on observing the hyperlinks clicked or the amount of scrolling performed. The authors present here an agent that combines many techniques. Thus when the user navigates on a new page, the agent records (a) the text of the HTML file; (b) the number of hyperlinks the user clicked; (c) the amount of user scrolling activity; and (d) the amount of user mouse activity. Based of this information, (a) creates the input and (b)-(d) constitute the outputs of its training examples. When the agent observes that the user performs a large number of actions on the page, it can label that page as a positive instance of the user's interests. The agent *sums* the actions of the user on each page visited over a finite period of time. It represents the information recorded (HTML text) like a vector of word frequencies and uses a neural network (Backpropagation algorithm) to learn the user's profile. The authors use the HTML markup tags to distinguish the context of the word and for different places of the word it gives different weights. The experiment shows that the error for predicting mouse activity was much larger than the error of the other output units. These results could indicate that the user's mouse activity on a page does not correlate with the user's interest in a page.

This idea can be used to define an agent that silently observe the user's browsing behavior and then uses these training examples to learn different functions and gather pages that are likely to be of interest to the user.

2.3.3.5 Refining Search Based on User's Profile

Single keywords are usually ambiguous, or too general for the search engine. Moreover, they can occur in vast quantities of documents, thus making the search return hundreds of results, most of them being irrelevant. Giving additional keywords can refine the search and provide considerable improvement in the retrieved results. Good refinement words must have meaning that helps disambiguate or make more specific the original search word. Providing the refinement words would help a search engine to prune out documents where the word is used with any of its other meanings. There are three ways to expand the query: manual query expansion, semi-manual query expansion, and automatic query expansion. In manual query expansion, the user knows the intended meaning of the keywords he used. In semi-manual query expansion the system provides some synonyms for the keywords and the user selects relevant synonyms. In the automatic query expansion agents are used to find and use best extra keywords using the user's profile to obtain better results.

Intelligent software agents are being developed to deal with these issues. One of these agents called "WebMate" is presented in [Che98] and can be found at [WebMate]. It is a personal agent for browsing and searching the Web. It does roughly two things: (1) learning user interests incrementally, with continuous update and automatically providing documents that match the user interests (e.g. a personalized newspaper), and (2) helping the user to refine search so as to increase retrieval of relevant documents. The agent is composed of a stand-alone proxy that monitors user's actions to provide information for learning and refinement of the search, and an interface used for interaction with the user. In the filtering task the agent judges whether an article is relevant or irrelevant to the user, based on the user's profile, in a probabilistic environment. In contrast with other systems that learn a user's profile and after that use it statically to determine relevant documents, WebMate learns the user profile incrementally and continuously. The system extracts and combines relevant keywords from the relevant pages provided by the user and uses them for keyword refinement. In WebMate agent, the context of the searched keywords in the "relevant" pages is used to refine the search because they think that the context of the search keywords is more informative than the content of the pages. They do this by considering what pages were marked by the user as relevant.

There are at least two methods of specifying the user's profile. In the first method the profile is specified explicitly as the users provides information about him (her) self. Usually users don't update and don't modify over time their profile, and the profile is almost always out of date. In the second method the profile is specified implicitly as the profile is created based on the users' actions observation.

Albanese et al in [Alb04] presented an algorithm for customizing the content and the structure of Web sites in order to provide users with the information they are interested in. As far as personalizing the content of Web site is concerned, the novelty of the structure is that they use a two phase classification approach rather than a single phase approach. The authors take into account both users provided data and browsing patterns and they classify both users and contents.

The algorithm has two phases. In the first phase a pattern analysis and classification is performed by means of an unsupervised clustering algorithm, using the registration information provided by the users. In the second phases a reclassification is iteratively repeated until a suitable convergence is reached. Also the authors use reclassification to overcome the inaccuracy of the registration information and they use observations based on users' navigation behavior. The authors use a clustering procedure for partitioning the feature space built upon the user provided data into a certain number of clusters that group together users appearing to be similar. The reclassification phase is based on the interaction (query, navigation, searching among directories) of each user with the Web site.

Barbu and Marin in [Bar03] present a new algorithm for learning the user's profile. Because search engines return many pages for a given query there are various methods to prune irrelevant documents. Some methods proposed in recent years use the user's profile for filtering the information from search engines. This user's profile can be used to filter the documents returned from search engines or can be used to reformulate the query based on the user's interests. The authors try to keep track of the user's interests by building an individual user's profile and modify this profile over the time. The authors want to identify what part of the user's profile is relevant with the current search. Thus the authors proposed a scheme that learns the user's profile continuously and incrementally using the initial user's profile, actions of the user and a semantic interpretation of the query. The authors take into account the user's current interests and the interest decrease in time if the interests change. From this, results a model called *time-word vector hyperspace* that computes the dynamics of the user's profile. Thus it is created a vector that has word component computed using TD-IDF (Terms Documents – Inverse Document Frequency) technique and the temporal dimension set to zero. Queries are represented as features vectors but in addition to the TF-IDF weights and the temporal dimension set to an initial value that decays in time. This allows that some specific user interests could decrease with time or if the user is interested in that category it can be maintained / increased. For the semantic analysis of the query the authors used WordNet. WordNet [WordNet] is an online lexical system developed by the Cognitive Science Laboratory at Princeton University that is inspired by current psycholinguistic theories of human lexical memory and it is organized into synonym sets, each set representing one underlying lexical concept. The WordNet provides various meanings for a given word.

The user can either input an explicit query or he can narrow his search process when he clicks on the links of the displayed Web pages. Contextual relevant information improves the search performance by filtering the retrieved documents by a relevance score computed as the cosine similarity between the User's Recent Profile feature vector and the feature vectors of the documents retrieved by a classical search engine. The preliminary result shows that the quality of the search result is improved. Thus the authors show that for a classical search engine, from the results of a query only 6 documents from the first 10 are relevant with what the user wants. When they use a standard profile filtering this number is increased to 8 out of 10 and when they use a dynamic profile filtering all presented articles (10 out of 10) are relevant with what the user wants.

Another interesting algorithm to improve the user's profile used for filtering the result of the classical search engines is presented by Tresp and Kai in [Kai02] where the authors combine collaborative filtering and content based filtering into probabilistic framework called by them *Collaborative Ensemble Learning*. Content Based Filter (CBF) represents a system that analyzes the content of a set of items together with a rating provided by the individual user from which it deduces unseen items that may be interesting for the current user. In contrast Collaborative Filter (CF) creates a database of ratings of items taken from a large set of users. The prediction for the current user is based on the rating provided by all other users. The difficulty problem for CBF systems is the gap between low level content features and high level user interface (human perception can't be formalized just based on the content analysis). Human understanding can be formalized when discussing text analysis but formalizing human perception of an image or a sound would only lead to irrelevant details regarding the ensemble. The CF systems memorize the user's preferences without incorporating the actual context of items. This system can not recommend a new page for which there isn't a previous rating given by a user. The algorithm uses a probabilistic model of Support Vector Machine to represent each user's profiles (like CBF) and in the prediction phase, it is combined with user's profile groups (from CF).

For modeling the profile of the user it is used the Support Vector Machine (SVM) [Nel00]. SVM is a classification technique that was applied with great success in many classification problems. The authors extend the standard SVM, that doesn't have a measure of confidence, with a probabilistic extension suggested by Platt [Pla99_2] (called PSV) where the authors associate a probability of class membership to each user. In the collaborative ensemble learning the authors combine current user's preferences, described by the user's model using PSVM, and other users' preferences, described by PSVM, and does the prediction for current user.

Unseen items are described by their feature vectors. To estimate these items the authors use the user's rating (if a user likes or dislikes this item). The authors put all items into a probabilistic framework under four assumptions. The first assumption is that every user is described by a profile that is generated from apriori distribution. The second is that the distribution of the actual items x is independent of the user's profiles. Thirdly, the user has a given rating based on his profile, and fourthly, given a user profile the opinions for individual items are mutually independent. The authors interpreted the results of predicting a current user preference as a combination of user's profiles society.

For modeling the user's profile using support vector machines the authors used a linear kernel for the part of text retrieval problem and a radial basis function kernel for the part of art image retrieval problem. The authors assume that each user is interested in exactly one category.

In another paper [Kai03] Tresp and Kai improve the above algorithm by using a hierarchical Bayes framework for information filtering. The authors model the user's profile using content based probabilistic filtering that are combined with a user's preference society (like in collaborative filtering). The combination scheme can be interpreted as a hierarchical Bayesian approach in which a common apriori distribution is learned from related experiments. In hierarchical Bayes framework the authors assume that preference model for each user has been generated as a simple form of an apriori distribution.

In the experimental part collaborative ensemble learning is compared with two methods of information filtering, (1) one pure collaborative filtering using Pearson correlation, and (2) a pure content based filtering using SVM with a linear kernel. The results were reported based on two data sets, the Reuters text data set (that has 36 categories covering 10034 articles) and a data base of user opinions on art images (642 images).

For text data experiment the authors assumed that each user is interested in exactly one category, and presented results for 360 users by choosing a random set of examples. The authors presented

results for two scenarios: first with only 5 examples for each user (insufficient information about users) and second with 30 examples for each user. Collaborative filtering performs worst in both cases, content based filtering using SVM provides reasonable good results and collaborative ensemble learning outperforms both other methods.

For the data base of user's opinions on art images the authors collected user's preferences for art images in a Web-based survey, choosing a total of 642 images and asked for their opinions (like/ dislike/ not sure). The authors collected data from 190 users, at average each of them had rated 89 images. To describe the images content they used 256 features, 10 features based on wavelet texture and 9 features on color moment, obtaining a 275 dimensional feature vector for each image. The authors evaluated the performance for 2, 5, 10, 20, 50 and 100 rated images, chosen at random from the collected data for this particular user. Content based filtering gave a very poor performance, since the low level image features are not indicative of the image content. Collaborative filtering performs well, once the number of rated images for the text user is suitably large. Collaborative ensemble learning achieved excellent performance when very few examples are given for a text user.

2.3.4 Semantic Web and Ontologies

The definition of Semantic Web according to Tim Berners-Lee, the inventor of World Wide Web is: "The extension of current Web in which information is given well-defined meaning, better enabling computers and humans to work in cooperation."[Ber99] tell us the actual tendency in this huge domain and the challenge that the computers can understand and offer us more and better information faster than now.

The majority of today's World Wide Web content is designed for human to read and understand, not for machine and computer programs to manipulate meaningfully. Computers can adeptly parse Web pages for layout and routine processing but, in general, machines have no reliable way to process the semantics. Technology developed in the last decade, including of course the development of World Wide Web give us the access to a huge quantity of information, more than we can understand and manage. A recent studies [More05] found that "the average knowledge worker in a Fortune 1000 company sends and receives 178 messages daily", whilst an academic study has shown that the quantity of information in the public Web tripled between 2000 and 2003 [Lym05]. We urgently need techniques that will help us give a sense at all quantity of available information in order to quickly find what we want and ignore the rest. Those techniques will extract and "summarize" what it is important in this narrow domain and will help us understand the connection between obtained information.

In the last years the knowledge management becomes a very important objective. The components that help in a significant mode the peoples that regularly work with that knowledge, improving the efficiency of the work, will have a great influence in representing of knowledge. Integration is also the exchange key for information management. For example information integration is necessary for "reach a better understanding of business through its data" [Cha05], for obtaining a common point of view for all data types and for understanding relation between them.

Semantic Web is the abstract representation of data on the WWW, based on the RDF (Resource Description Framework) standard [Rdf] and other standards used to defined it. This is being developed by the WWW Consortium with participation from academic researchers and industrial partners. Data can be defined and linked in such a way so discovery, automation, integration and reuse across different applications are more effective. The Semantic Web will bring structure to the meaningful content of Web pages, where software agents roaming from page to page or from site to site can readily carry out automated sophisticated task for users.

The semantic Web and its technologies offer us a new onset for information and process management, main principle of developing being the use of a semantic metadata. The metadata can have two levels. At the first level the metadata can describe a document, for example a Web pages or a part of a document (a paragraph). At the second level the metadata can be use to describe entities from document, for example a person or a company. In each case the main think is that the metadata is semantic, that is what tell us about the content of the document (for example the subject matter or relation with the others documents) or about entities from document. This new metadata is in contrast with the existed metadata from the current Web that is codified in the HTML and summary describe the format in which the information will be presented to the user. In the actual format using metadata we can specified that the string is displayed using bold and red fonts but we can't describe if the string represents the name of the author or the name of the company.

There are a lot of advantages that becomes from representing the metadata in the new format [Dav06]:

- The ability to reason, to draw inferences from the existing knowledge to create new knowledge;
- We can organize and find information using its sense not only the content (text);
- Computers can understand when the words and phrases are equivalents;
- Computers can distinguish when same word is used with different sense;
- We can improve the modality of representing the search results (the information can be clusters based on the meaning);
- We can put together information from all documents and summarizing those better;
- Relationships between key entities in the documents can be represented visually.

The use of semantic metadata is also crucial for integrating information from heterogeneous sources, whether within one organization or across organizations. Different schemes are used for describing and classifying information, and different technologies are used within the information.

The following sub-branches of Artificial Intelligence are being addressed to Semantic Web:

- *Knowledge acquisition and representation* - that is defined as the extraction of knowledge from different sources of information and express that knowledge in a computer tractable form, so that it can be used to help software-agents to perform well;
- *Agent systems* – are a software that perceives its environment through sensors and acts upon that environment through effectors;
- *Ontology* - is a document or file of vocabularies that formally define the relations among terms. The most typical kind of ontology for the Web has concepts and a set of inference rules between concepts.

Ontology represents the heart of all semantic Web applications. A commonly agreed definition of ontology is: "ontology is an explicit and formal specification of a conceptualization of a domain of interest" [Gru93]. Definition has two key points: that the conceptualization is formal and hence permits reasoning by computer; and that a practical ontology is designed for some particular domain of interest. Ontologies were developed in Artificial Intelligence to facilitate reutilization and common using of knowledge, to provide a computer semantic processing of knowledge thus to make communication between different agents (software or human) possible.

Ontologies are used to organize knowledge into a structured way in many areas – from philosophy to knowledge management and the Semantic Web. Usually the ontology is referred as a graph or network structure consisting from [Fen04]:

- A set of concepts – vertices in a graph;
- A set of relationships connecting concepts – edges in a graph;
- A set of instances assigned to a particular concept – data records assigned to concepts or relation.

From the perspective of knowledge discovery, semi-automatic ontology construction can be defined as consisting on the following interrelated phases:

- Domain understanding – the area we are dealing with;
- Data understanding – available data and its relations;
- Task definition – available data and its proprieties;
- Ontology learning – semi-automatic process addressing the task definition;
- Ontology evaluation – estimate the quality of the solution;
- Refinement with human in the loop – perform any transformation needed to improve the ontology.

Ontologies are described using different languages as OIL (Ontology Interchange Language), SHOE (Simple HTML Ontology Extensions), DAML (DARPA Agent Markup Language) and OWL (Ontology Web Language). Those are build using RDF (Resource Description Framework) that is an essential data modeling language. It is based on graph representation, but usually serialized as XML. Essentially, it consists of triplets:

- Subject – is a resource or a instance;
- Predicate – is also a resource that explain what does the subject;
- Object – may be a resource, a blank node or a string of characters.

The construction of ontology can be a time consuming process, needing the services of experts both in ontology engineering and in the domain of interest. In general the build of ontology is in fact the extraction of metadata from the existing data, meaning the ability to automatically extract metadata from large text data and to be able to represent and use a new metadata to annotate the text. The ontology also needs to be able to change when the knowledge are modified and to use this new knowledge.

Researches in direction of Semantic Web are trying to reorganize data from the Web by giving them a meaning and making possible the automatic processing data from a semantically point of view. This is a revolutionary concept because Web data are described through a powerful meta-information system. There are some particular implementation attempts, but right now it is only in the research phase. For this PhD thesis I chose an evolutionary way that approaches the actual Web considering its present day organization. Obviously data mining techniques will still be used also when the Semantic Web will be implemented through particular domain ontology.

2.4 The Data Sets Used

Typically, only a small fraction of many available documents will be relevant to a given individual user. Without knowing what could be in the document, it is difficult to formulate effective queries for analyzing and extracting useful information from the data. Users need tools to compare different documents, rank the importance and relevance of the documents. Thus text data mining has become increasingly important. Text mining goes one step beyond keyword-based and similarity-based information retrieval and discovers knowledge from semi-structured text data using methods such as keyword-based association and document classification. In this context

traditional information retrieval techniques become inadequate for the increasingly vast amounts of text data. Information retrieval is concerned with the organizing and retrieval of information from a large number of text-based documents. Most information retrieval systems accept *keyword-based* and/or *similarity-based* retrieval [Cro99]. Those ideas were presented in detailed in sections 2.1 and 2.2 and also in my first PhD technical report [Mor05_1].

2.4.1 Preprocessing the Used Dataset

My experiments are performed on the Reuters-2000 collection [Reu00], which has 984Mb of newspapers articles in a compressed format. The Reuters collection is commonly used in text categorization research. Collection includes a total of 806791 documents, with all news stories published by Reuters Press covering the period from 20th August 1996 through 19th August 1997. The articles have 9822391 paragraphs and contain 11522874 sentences and 310033 distinct root words after was eliminated the stopwords. Each news is stored as an XML file (*xml version="1.0" encoding="UFT-8").* Files are grouped by date in 365 zip files. Documents are preclassified by Reuters according to three categories. According to the industry criterion the articles are grouped in 870 categories. According to the region the article refers to, there are 366 categories and according to topics there are 126 distinct topics, 23 of them contain no articles. In my experiment I took in consideration the topics proposed by Reuters using them as reference for my classification methods.

The definition of the metadata element of Reuters Experimental NewsXML allows the attachment of systematically organized information about the news' summary. The metadata contains information about date when the article was published, language of the article, title, place, etc. The title is marked using title markups like <title> </title>. Then the text of the article follows marked by <text> and </text>. After the content of the article is a structure that is based on a scheme for "Reuters Web" that are in the following format:

```
<metadata>
<codes class="bip:countries:1.0">…</code>
<codes class="bip:topics:1.0">…</code>
<codes class="bip:industry:1.0">…</code>
<!ENTITY % dc.elements

"(dc.title |

dc.creator.name |
dc.creator.title |
dc.creator.location |
dc.creator.location.city |
dc.creator.location.sublocation |
dc.creator.location.stateOrProvince |
dc.creator.location.country.code |
dc.creator.location.country.name |
dc.creator.phone |
dc.creator.email |
dc.creator.program |

dc.date.created |
dc.date.lastModified |
dc.date.converted |

dc.publisher |
dc.publisher.provider |
dc.publisher.contact.name  |
dc.publisher.contact.email  |
dc.publisher.contact.phone  |
```

```
dc.publisher.contact.title  |
dc.publisher.location |
dc.publisher.graphic  |

dc.coverage.start      |
dc.coverage.end        |
dc.coverage.period     |

dc.relation.obsoletes  |
dc.relation.includes   |
dc.relation.references |

dc.date.published |
dc.date.live |
dc.date.statuschanges |
dc.date.expires |

dc.source |
dc.source.contact   |
dc.source.provider  |
dc.source.location  |
dc.source.graphic        |
dc.source.date.published |
dc.source.identifier  |

dc.contributor.editor.name |
dc.contributor.captionwriter.name )">
</metadata>
</newsitem>
```

In the entries above there is information included on region, topic and industry (with bold in the presented example). This information is included using codes. There are separate files, published by Reuters, where it can be found the correspondence between the codes and the complete name. In my application I used the name of the news stories, the content of the news, the topics proposed by Reuters to classify the document and the industry code to select the document. Thus from each file I extract the words from the title and the content of the news and I create a vector that characterizes that document.

A text retrieval system often associates a stoplist with a set of documents. A stoplist is a set of words that are deemed "irrelevant" for a set of documents [Jai01] [Ian00]. In my application I have used a general stopwords list from the package *ir.jar* [Ir] from Texas University. I wanted to use a general list in order to eliminate the non-relevant words. In the stopword list there are 509 different words included which are considered to be irrelevant for the content of the text. At this list I add the word "Reuters" that also occur in all documents and is not relevant in my experiment.

For each word that remained after the elimination of stopwords I have extracted the root of the word (stemming) using a root extraction implementation from the package ir.jar [Ir] from Texas University. If after stemming I obtained the root with dimension smaller or equal to 2 (the word obtained has only two or one characters) and the original dimension of the word was 2 or 1 I eliminate those words. If the original dimension was greater than three, I kept the word in the original format without stemming. Up to this moment I didn't deal with the case in which two different words have the same root after stemming. Afterwards I counted the occurrences of every remaining root. Thus I have created an array for each document from Reuters with the remaining roots (called later features). This array is the individual characterization of each document.

For example starting from a file that has the following tile, text and codes:

```
<title>USA: Spyglass launches new web software version.</title>
<text>
```

```
<p>Spyglass Inc said Tuesday it launched a new version of its Web Server
Software Development Kit, which it said is the first to be fully compatible
with both Microsoft Corp and UNIX platforms.</p>
<p>The software also provides improved features such as file caching and secure
communications, Spyglass said.</p>
<p>Pricing for the software starts at $25,000, which includes 100 licenses and
a one-year support and maintenance contract.</p>
<p>--Chicago news desk, 312-408-8787</p>
</text>

<codes class="bip:topics:1.0">
  <code code="C22">
    <editdetail attribution="Reuters BIP Coding Group" action="confirmed"
date="1996-08-20"/>
  </code>
  <code code="CCAT">
    <editdetail attribution="Reuters BIP Coding Group" action="confirmed"
date="1996-08-20"/>
  </code>
</codes>
```

I extracted 33 different root word (features) and 2 topics (C22 and CCAT). The roots of the words obtained are:

```
compat, corp, cach, commun, contract, chicago, develop, fulli, featur, file,
improv, includ, kit, lauch, launch, licens, mainten, newsdesk, platform, price,
spyglass, softwar, server, soft, secur, start, support, tuesdai, usa, unix,
version, Web, year
```

Using this obtained order of the features, the vector that characterizes this file can be represented in the following format:

```
0:1 1:1 2:1 3:1 4:1 5:1 6:1 7:1 8:1 9:1 10:1 11:1 12:1 13:1 14:1 15:1 16:1 17:1
18:1 19:1 20:3 21:4 22:1 23:1 24:1 25:1 26:1 27:1 28:1 29:1 30:2 31:2 32:1 #
c22 ccat
```

Vector is represented in a sparse format [SparseMatrix]. In this format are stored only the values that are greater then zero. Each pair of values, delimited by " " contain before ":" symbol the feature number, features been in the order presented above and after ":" symbol the number of occurrences of this feature in the presented document. The "#" symbol at the end of the line introduces Reuters classification topics. In this example Reuters proposes 2 topics. This representation of documents is called in the literature bag-of-words approach.

Afterwards I have created a large frequency vector with all unique tokens and the number of occurrences of each token in all documents (in order to further apply the classification method). I use a vector with all tokens to memorize each document in the set. If a token appears in the document then I stored the number of occurrences in the new vector otherwise I stored nothing for that token (sparse vector [SparseMatrix]). By doing so all vectors have the same size and represent the document in the entire set as a line into an array. This representation can be considered as a signature of the document in the documents set.

2.4.2 Training/Testing Files' Structure

Due to the huge dimensionality of the database I present results obtained using a subset of data. From all 806791 documents I select those documents that are grouped by Reuters in "System Software" (I330202) as industry code. After this selection I obtained 7083 documents that are represented using 19038 features. In the resulting set there are 68 different topics for classifying according to Reuters. For those 68 topics I have eliminated those topics that are poorly or excessively represented. Thus I eliminated those topics that contain less than 1% documents from

all 7083 documents in the entire set. I also eliminated topics that contain more than 99% samples from the entire set, as being excessively represented. After doing so I obtained 24 different topics and 7053 documents. For multiclass classification I used all 24 topics. Those 7053 documents are divided randomly into a training set of 4702 documents and a testing set of 2351 documents (samples).

Most researchers take in consideration only the title and the abstract of the article or only the first 200 words from each piece of news. In my evaluation I take into consideration the entire news and the title of the news in order to create the characteristic vector. I have used the bag-of-words approach for representing the documents as presented in section 2.2.1.2.

For binary classification I chose topic "c152" that means "Comment /Forecasts" according to Reuters codes and this subset will be referred here as Subset-c152. I have chosen topic "c152" as it contains only about 30% of the data belonging to that class so the algorithm can actually learn. For the binary classification the training and testing set contain same number of samples and have the same dimension as for multiclass classification. This set is especially used for presenting the clustering results.

The algorithm that I will present is based on the fact that data is grouped only in two classes. This is in fact a binary classification algorithm where the labels can take only two values. Therefore in this phase I took in consideration only one class of documents and documents that are not belonging to that class are considered to be in another class. For classification in more than two classes I take in consideration all topics and I learn separately for each topic using "one versus the rest" technique. Applying this technique I chose separately for each topic all samples that have a specific topic versus the rest of samples for entire set. Thus, considering M classes, repeated two class classifications for each topic obtaining M decision functions.

In the first phase the developed application process the Reuter's files chosen as above, in order to make the words frequency vectors. The large frequency vectors are stored in four different files after creation. Two of these files contain the data necessary for training and testing classification. The other two files contain data necessary for training and testing multiclass classification. The files structure format is presented further. This format is almost identical with the format presented by Ian in [Ian00] and used by Weka application that can be found at [Weka]. The files have a part containing attributes, a part containing topics and a part containing data. In the attributes part there are specified all attributes (tokens) that characterize the frequency vectors. The attributes are specified using the letterhead *"@attribute*" followed by the name of the attribute. The order of the attribute is important because in the data part the attribute is referred by an order number not by name. In the topic part there are specified all the topics for this set according to Reuter's classification. The topics are specified using the letterhead "@topics" followed by the name of the topic and the number of samples that contain that topic. The data section is marked using "@data" and contains all large frequency vectors stored as sparse vectors in the format presented above. For the output files where we have all the topics (for multi-class classification) the large frequency vectors are ordered by their occurrence in Reuter's files. For the output files where we have only one topic (for classification one versus the rest) the large frequency vectors are ordered by topic. Initially I put the large frequency vectors that belong to the class and after that I put the large frequency vectors that don't belong to the class.

The structure presented below is obtained as a result of text extraction on Reuter's database. In this step I have used some classes taken from the package *ir.jar* [Ir] from Texas University. In order to test the influence of feature selection on classification accuracy I used several levels of threshold for Information Gain, SVM feature selection and Genetic Algorithm. I will present later the exact value of threshold and number of resulting attributes. The training and testing file need to have the same structure. This means that we need to have the same number of attributes and topics. Both

attributes and topics have to be in the same order both in the training and testing file. As follows I will present a sample structure.

```
@attribute compat
@attribute corp
@attribute cach
@attribute commun
@attribute contract
@attribute contract
@attribute chicago
@attribute develop
@attribute fulli
@attribute featur
@attribute file
@attribute improv
@attribute includ
@attribute kit

@topic c22
@topic ccat
@topic c15
@topic c152
@topic c313

@data
0:1 1:1 2:1 3:1 4:1 5:1 6:1 7:1 8:1 9:1 10:1 11:1 12:1 13:1 # c22, ccat
0:3 1:2 2:3 3:2 7:3 8:1 11:8 13:1 # c22, ccat
0:1 1:5 2:1 3:5 5:1 7:9 11:2 13:3 # c22
0:4 1:1 2:1 3:3 5:4 8:1 13:1 # c15
0:2 1:9 2:7 3:2 5:2 7:6 11:11 13:3 # c15, c152
0:3 1:2 3:17 4:3 6:14 9:13 14:1 # 1
2:9 4:5 5:2 6:5 9:1 10:1 11:6 12:8 # c15
0:3 2:2 4:1 5:2 7:2 8:5 9:1 11:2 # c152
0:1 2:2 3:5 4:3 6:3 7:9 8:7 9:2 12:1 # c313
0:2 3:4 4:1 6:6 9:2 12:1 # c152
0:4 2:2 3:1 4:1 6:6 7:2 8:1 9:1 12:4 # c313
0:3 2:1 3:3 4:2 6:2 9:4 12:1 # c313
0:1 3:2 4:2 6:1 7:7 8:1 9:2 12:4 # c152
```

2.4.3 Choosing a Larger Dataset

During the section 7.3 I will use a larger data set for some experiments. In what follows I am presenting the method of obtaining this data set. Thus from entire Reuters Database [Reu00] I have selected those documents that are grouped by Reuters in “Computer Systems and Software” (I33020) by industry code. After this selection there are 16177 documents left to work on. From these documents I extracted a number of 28624 features after eliminating the stopwords and extracting the root of the word (stemming). This new set is greater comparatively with previous selected set of 7053douments and 19038 features. Those documents are pre-classified by Reuters after topic and in the resulting set there were obtained 69 different topics, according to Reuters. From those topics I eliminated the topics that are poorly or excessively represented and only 25 topics remained for multi-class classification. After this process the dimension set was reduced at 16139 samples. Those 16139 samples are divided randomly into a training set of 10757 samples and a training set of 5382 samples.

2.4.4 Preprocessing the Used Web Dataset

It is well known that the World Wide Web may be considered as a huge and global information center. A Web site usually contains great amounts of information distributed through hundreds of

pages. Without proper guidance, a visitor often wanders aimlessly without visiting important pages, loses interest and leaves the site sooner than expected. This consideration is at the basis of the great interest about Web information mining both in the academic and the industrial world.

In this paragraph I want to preset the method used for build the database for Web mining. I am interested at this moment only in Web content mining without being interested here about mining the structure of the Web [Mob96] or mining the usage of the Web [Str00] [Alb04]. In this idea I took a lot of pages from the Web directories *DMOZ* [Dmoz]. I chose this site because all of my research is based on a labeled document classification and on this site, that expanded in the last years, there are a lot of documents that are pre-classified. I selected a lot of categories from the DMOZ sites and I took all documents that were classified in those categories. I selected a general category "computers" and from this category I selected a lot of documents and subcategories. Thus 18 categories were selected. A document is considered classified in the category where it is found and also in the categories from the previous levels. The same document can also be obtained from different subcategories if in that document are information that refer all subcategories. In this case I leave the document in all the subcategories. The following categories were selected:

- Computers
- Algorithms
- Compression
- Computational
- Cryptography
- Digital Signal Processing
- Genetic
- Graphics
- Numerical Analysis
- Pseudo-random numbers
- Sorting
- Artificial Intelligence
- Artificial Life
- CAD
- Data Communications
- Modems
- Telephony Wireless

A number of 745 html files were selected from those categories. Each document is classified in at least 2 categories. This number of files remained after automatically eliminating the pages that are not existing and manually (for this moment) selected the pages that are written in other language than English. This elimination was necessary because for this moment in the next step of feature extraction I use an algorithm of root extraction only for English. After the feature extraction step from 745 html documents were extracted 26998 roots of words. In the feature extraction step were also eliminated the irrelevant words using a list of 509 stopwords. From each document were eliminated the html tags and were used only the document's title and all the words from the document. I represented the document as a vector of words (bag-of-words). The selected documents were split randomly in a training set that contain 445 documents and a testing set of 377 documents. Results about Web mining and classifying categories were presented in section 4.6.4.

2.4.5 Type of Data Representation

Because there are many ways to define the term-weight (features), I represented the input data in different formats in my application, and I try to analyze their influence. I take in consideration three formats for representing the data [Cha03].

- Binary representation – in the input vector I store "0" if the word doesn't occur in the document and "1" if it occurs without considering the number of occurrences. The weight can only be 0 or 1.
- Nominal representation – in the input vector I compute the value of the weight using the formula:

$$TF(d,t) = \frac{n(d,t)}{\max_{\tau} n(d,\tau)} \tag{2.15}$$

where n(d, t) is the number of times that term t occurs in document d, and the denominator represents the value of term that mostly occurs in document d, and $TF(d,t)$ is the term frequency. The weight can take values between 0 and 1.

- Cornell SMART representation – in the input vector I compute the value of the weight using the formula:

$$TF(d,t) = \begin{cases} 0 & if\ n(d,t) = 0 \\ 1 + \log(1 + \log(n(d,t)) & otherwise \end{cases} \tag{2.16}$$

where $n(d,t)$ represent number of times term t occurs in document d. In this case the weight can take value 0 if the word does not exist in the document and between 1 and 2 if it exists. The value of the weight is bordered by 2 for reasonable numbers of appearances of a word in a document (less then 10^9 if the logarithm used is based 10).

The idea of separating these representations is that we can also measure the influence of appearance of the word in the document on classification accuracy. In the first representation I'm interested only if a word appears or not in the document. In the second representation is also kept the number of appearances of the word in the documents but normalized over all documents. In the last case the value of the weight creates a gap between the values which represent that the word doesn't occur in the document and the value of at least one appearance.

Because the above formula (2.16) represents a local normalization (only for one document and not for the entire set of documents) we can also make a global normalization which introduces a measure over all the documents from the set (not all axes from the vector space are equally important). We can compute this global norm by using the IDF (Inverse Document Frequency):

$IDF(t) = \log \frac{1+|D|}{|D_t|}$, where D is the document collection and D_t is the set of documents that contain term t. If $|D_t| << |D|$ the term t will enjoy a large IDF scale factor and vice versa.

TF and IDF are combined into the complete vector-space model in an obvious way: the coordinate of document d in axis t is given by:

$$d_t = TF(d,t)IDF(t)$$

When working with large data sets, the results obtained using a global normalization IDF were usually worst that the ones obtained using only TF and I am not presenting them in this report.

> "Mathematics may be defined as the subject in which we never know what we are talking about, nor whether what we are saying is true."
>
> Bertrand Russell

3 Support Vector Machine. Mathematical Background

In this chapter I am presenting some theoretical aspects referring to Support Vector Machine (SVM) technique and some aspects referring to the implementation of this technique for classifying and clustering text documents. I chose this technique based on kernels from all the existing classification techniques (neural networks, Bayesian networks, Markov chains, etc.) because is relatively new and obtains better performances in image recognition, speech recognition and other difficult problems involving learning. Also I was attracted by its possibility to work with very large vectors dimensions and very large data sets keeping reasonable complexities and costs, from both time and algorithms points of view.

Classification is the task of classify examples into one of a discrete set of possible categories. Classification is a supervised learning technique that uses a labeled dataset for training and tries to find rules that split the best the training data set. Clustering is the unsupervised task of grouping the data into classes (or clusters) so that objects within a class (cluster) have high similarity, but are very dissimilar in comparison with the objects from other classes (clusters).

I start by describing the classifier that forms the basis for SVM, the separating hyperplane. This technique is based on classifying only in two classes. Many practical problems require us to classify the data into more than just two classes. There are some methods used for classification in more than two classes. At the end of this chapter I present some changes that need to be made in the classification algorithm, so that it can work with unlabeled documents (clustering). In this chapter I also present the method chosen for implementing this algorithm for documents classification.

3.1 SVM Technique for Binary Classification

3.1.1 Separating Hyperplane

Support Vector Machine (SVM) is a classification technique based on the statistical learning theory [Sch02], [Nel00]. It was successfully used a few centuries old mathematical discoveries to solve optimization problems on large sets as far as the numbers of articles and their characteristics (features) are concerned. The crucial idea is to use kernels to reduce a complex classification task to one that can be solved with separating hyperplanes.

The purpose of the algorithm is to find a hyperplane (in an n-dimensional space the hyperplane is a space with the dimension n-1) that optimally splits the training set (a practical idea can be found in [Chi03]). Actually the algorithm consists in determining the parameters that determine the general equation of the plane. Looking at the two dimensional problem we actually want to find a line that

"best" separates points in the positive class (points that are into the class) from points in the negative class (the remaining points). Starting from a simple pattern recognition learning algorithm that is arguably one of the simples possible we can obtain a decision function that is based on dot product between samples with positive and negative labels. In the SVM algorithm the separating hyperplane is characterized by a decision function like $f(x) = \text{sgn}(\langle \mathbf{w}, \mathbf{x} \rangle + b)$ where "w" is the weight vector orthogonal to the hyperplane, "b" is a scalar that represents the margin of the hyperplane, "x" is the current sample tested and $\langle \cdot, \cdot \rangle$ representing the dot product. "Sgn" is the sign function that returns 1 if the value is greater than 0 and -1 otherwise. In generally w will be scaled by ||w||. In the training part the algorithm need to find the normal vector "w" that leads to the largest "b" of the hyperplane. For better understanding let's consider that we have data separated into two classes (circles and squares) as in Figure 3.1. The problem that we want to solve consists in finding the optimal line that separates those two classes.

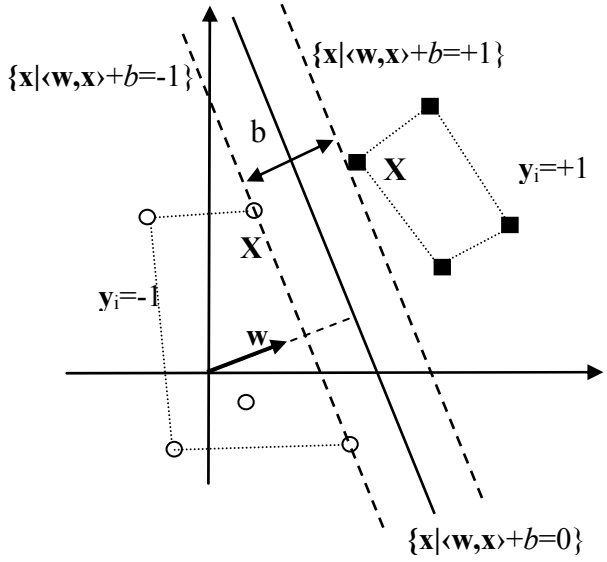

Figure 3.1 - The optimal hyperplane with normal vector w and offset *b*

The problem seems very easy to solve but we have to keep in mind that the optimal classification line should classify correctly all the elements generated by the same given distribution. There are a lot of hyperplanes that meet the classification requirements but the algorithm tries to determine the optimum.

Let $x, y \in \Re_n$ where $x = (x_1, x_2, ..., x_n)$ and $y = (y_1, y_2, ..., y_n)$. We can define the dot product of two vectors as the real value which measures the area determined by the two vectors. We will denote this further on as $\langle x, y \rangle$ and we will compute it:

$$\langle x, y \rangle = x_1 * y_1 + x_2 * y_2 + ... + x_n * y_n \tag{3.1}$$

We say that *x* is orthogonal on *y* if their dot product equals to 0, that is if:

$$\langle x, y \rangle = 0 \tag{3.2}$$

If **w** has the norm equal to 1, then $\langle \mathbf{w}, x \rangle$ is equal to the length of x along the direction of **w**. Generally **w** will be scaled by $\| \mathbf{w} \|$ in order to obtain a unit vector. By $\| \mathbf{w} \|$ we mean the norm of **w**, defined as $\|\mathbf{w}\| = \langle \mathbf{w}, \mathbf{w} \rangle$ and also called Euclid's norm.

This learning algorithm can be performed in a dot product space and for data which is separable by hyperplane, by constructing *f* from empirical data. It is based on two facts. First, among all the hyperplanes separating the data, there is a unique optimal hyperplane, distinguished by the maximum margin of separation between any training point and the hyperplane. Second, the capacity of the hyperplane to separate the classes decreases with the increasing of the margin. These are two antagonistic features which transform the solution of this problem into an exercise of compromises.

$$\underset{w\in\mathcal{H},b\in\Re}{\text{maximize}} \min\left\{\left\|\mathbf{x}-\mathbf{x_i}\right\| \middle| \mathbf{x}\in\mathcal{H}, \left\langle \mathbf{w},\mathbf{x}\right\rangle + b = 0, i = 1,...,m\right\} \tag{3.3}$$

Everything that we have done so far is linear in the data. To allow for much more general decision surface (for example training data which is not separable by a hyperplane in the input space) we can use kernels to nonlinearly transform the input data $x_1,...,x_m \in \Re$ into a higher dimensional feature space, using a map $\Phi : x \mapsto x_i$; we then do a linear separation by construct a separating hyperplane with the maximum margin there. This yields a non-linear decision boundary in the input space. For maximizing the target function and for evaluating the decision function requires the computation of the dot product $\left\langle \Phi(w), \Phi(x_i)\right\rangle$ in a higher-dimensional space. By the use of a positive definite kernel $\left\langle \Phi(\mathbf{w}), \Phi(x)\right\rangle$ it is possible to compute the separating hyperplane without explicitly carrying out the map into the new higher feature space [Sch02]. This computation, which is sometime referred as the kernel trick, was used by Boser [Bos92] to extend the generalized hyperplane classification to nonlinear SVM. The kernel trick can be applied since all vectors only occurred in dot product.

In order to find the optimal hyperplane, distinguished by the maximum margin, we need to solve the following objective function:

$$\begin{aligned} &\underset{\mathbf{w}\in H,b\in\Re}{\text{minimize}}\ \tau(\mathbf{w}) = \frac{1}{2}\|\mathbf{w}\|^2 \\ &\textit{subject to } y_i\left(\left\langle \mathbf{w},\mathbf{x}_i\right\rangle + b\right) \geq 1 \textit{ for all } i = 1,...,m \end{aligned} \tag{3.4}$$

The constraints ensure that $f(x_i)$ will be +1 for $y_1 = +1$ and -1 for $y_1 = -1$. This problem is computationally attractive because it can be constructed by solving a quadratic programming problem for which there are efficient algorithms. Function τ is called objective function (the first feature) with inequality constrains. Together, they form a so-called primal optimization problem. Let's see why we need to minimize the length of **w**. If ||**w**|| would be 1, then the left side of the restriction would equal the distance between x_i and the hyperplane. Generally we need to divide $y_i\left(\left\langle \mathbf{w},\mathbf{x}_i\right\rangle + b\right)$ by ||**w**|| in order to change this into a distance. From now on if we can meet the restriction for all *i=1,..,m* with a minimum length **w** then the total margin will be maximal.

3.1.2 Dual Optimization Problem

Problems like this one are the subject of optimization theory and are in general difficult to solve because they imply great computational costs. To solve this type of problems it is more convenient to deal with the dual problem, which according to mathematical demonstrations leads to the same results. In order to introduce the dual problem we have to introduce the Lagrange multipliers $\alpha_i \geq 0$ and the Lagrangian [Sch02] which lead to the so-called dual optimization problem:

$$L(\mathbf{w},b,\alpha) = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{m}\alpha_i\left(y_i\left(\left\langle \mathbf{x_i},w\right\rangle + b\right) - 1\right) \tag{3.5}$$

with Lagrange multipliers $\alpha_i \geq 0$. The Lagrangian L must be maximized with respect to the dual variables α_i, and minimized with respect to the primal variables **w** and b (in other words, a saddle point has to be found). Note that the restrictions are embedded in the second part of Lagrangian and need not be applied separately.

We will try to give an intuitive solution to the restricted optimization problem. If the restriction is violated then $y_i\left(\langle \mathbf{w}, \mathbf{x}_i \rangle + b\right) - 1 < 0$, in which case L can be increased by increasing the corresponding α_i. At the same time **w** and b need to be modified if L diminishes. In order that $\alpha_i\left(y_i\left(\langle \mathbf{w}, \mathbf{x}_i \rangle + b\right) - 1\right)$ doesn't become an arbitrary large negative number the changes in **w** and b will ensure that for a separable problem the conditions will finally be met. Similarly, for all the restrictions that don't meet the equality (that is for which $y_i\left(\langle \mathbf{w}, \mathbf{x}_i \rangle + b\right) - 1 > 0$) the corresponding α_i needs to be 0. For those α_i the value of L is maximized. The second complementary restriction in the optimization theory is given by Karush-Kuhn-Tucker (also known as the KKT restrictions). At the saddle point the partial derivates of L with respect to the primal variables need to be 0:

$$\begin{aligned} &\mathbf{w} = \sum_{i=0}^{m} \alpha_i y_i x_i \\ &\sum_{i=0}^{m} \alpha_i y_i = 0 \end{aligned} \tag{3.6}$$

3.1.3 Support Vectors

The solution vector thus has an expansion in terms of a subset of training samples, namely those samples with non zero α_i, called Support Vectors. Note that although the solution **w** is unique (due to the strict convexity of primal optimization problem), the coefficients α_i, need not be. According to the Karush-Kuhn-Tucker (KKT) theorem, only the Lagrange multipliers α_i that are non-zero at the saddle point, correspond to constraints which are precisely met. Formally, for all i=1,…,m, we have:

$$\alpha_i\left[y_i\left(\langle \mathbf{w}, \mathbf{x}_i \rangle + b\right) - 1\right] = 0 \ \textit{for all } i = 1,\ldots,m \tag{3.7}$$

The patterns x_i for which $\alpha_i > 0$ are called Support Vectors. This terminology is related to the corresponding terms in the theory of convex sets, related to convex optimization. According to the KKT condition they lie exactly on the margin. All remaining training samples are irrelevant. By eliminating the primal variables **w** and b in the Lagrangian we arrive to the so-called dual optimization problem, which is the problem that one usually solves in practice.

$$\underset{\alpha \in \Re^m}{\text{maximize}}\, W(\alpha) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} \alpha_i \alpha_j y_i y_j \langle x_i, x_j \rangle \tag{3.8}$$

this is called the target function

$$\text{subject to } \alpha_i \geq 0 \ \textit{for all } i = 1,\ldots,m \ \textit{and} \ \sum_{i=1}^{m} \alpha_i y_i = 0 \tag{3.9}$$

Thus the hyperplane can be written in the dual optimization problem as:

$$f(x) = \operatorname{sgn}\left(\sum_{i=1}^{m} y_i \alpha_i \langle x_i, x \rangle + b\right) \tag{3.10}$$

where *b* is computed using KKT conditions.

The optimization problem structure is very similar to the one that occurs in the mechanical Lagrange formulation. In solving the dual problem it is frequently when only a subset of restriction becomes active. For example, if we want to keep a ball in a box then it will usually roll in a corner. The restrictions corresponding to the walls that are not touched by the ball are irrelevant and can be eliminated.

3.1.4 Soft Margin Hyperplanes

In practice, the separating hyperplane may not exist, for instance if the classes are overlapped. To take in consideration the samples that can possibly violate the restrictions we can introduce the slack variables $\xi_i \geq 0$ for all $\mathrm{i} = 1,...,\mathrm{m}$.

Using slack variables the restriction become $y_i\left(\langle \mathbf{w}, \mathbf{x}_i \rangle + b\right) \geq 1 - \xi$ for all *i=1,...,m*. The optimum hyperplane can now be found both by controlling the classification capacity ($\|\mathbf{w}\|$) and by controlling the sum of slack variables $\sum_i \xi_i$. We notice that the second part provides a superior boundary to the number of training errors. This is called soft margin classification. The objective function becomes then:

$$\tau(\mathbf{w}, \xi) = \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{m} \xi_i \tag{3.11}$$

with the same restrictions, where the constant C > 0 determines the exchange between maximizing the margin and minimizing the number of training errors. If we rewrite this in terms of Lagrange multipliers, we get the same maximization problem with an extra restriction:

$$0 \leq \alpha_i \leq C - \xi \;\; for \;\; all \;\; i = 1,...,\; m \;\; and \;\; \sum_{i=1}^{m} \alpha_i y_i = 0 \tag{3.12}$$

3.1.5 Kernel Trick

Everything was formulated in a dot product space. We think of this space as the feature space. On the practical level, changes have to be made to perform the algorithm in a higher-dimensional feature space. The patterns x_i thus need not coincide with the input patterns. They can equally well be the result of mapping the original input patterns x_i into a higher dimensional feature space using Φ function. Maximizing the target function and evaluating the decision function, than requests the computation of dot products $\langle \phi(x), \phi(x) \rangle$ in a higher dimensional feature space. These expensive calculations are reduced significantly by using a positive definite kernel *k*, such that $k(x, x') = \langle x, x' \rangle$. This substitution, which is referred sometimes as the kernel trick was used by Boser [Bos92] to extend the hyperplane classification to nonlinear Support Vector Machines. The kernel trick can be applied since all feature vectors only occur in dot products. The weight vectors than becomes an expression in the feature space, and therefore Φ will be the function through which we represent the input vector in the new space. Thus we obtain the decision function:

$$f(x) = \mathrm{sgn}\left(\sum_{i=1}^{m} y_i \alpha_i k(x, x_i) + b\right) \tag{3.13}$$

The main advantage of this algorithm is that it doesn't require transposing all data into a higher dimensional space. So there is no expensive calculus as in neural networks. We also get a smaller

dimensional set of testing data as in the training phase we will only consider the support vectors which are usually few. Another advantage of this algorithm is that it allows the usage of training data with as many features as needed without increasing the processing time exponentially. This is not true for neural networks. For instance the back propagation algorithm has troubles dealing with a lot of features. The only problem of the support vector algorithm is the resulting number of support vectors. As the number increases the response time increases linearly too.

3.2 Multiclass Classification

Most real life problems require classification on more than two classes. The presented algorithm is about binary classification, where the class labels can only take two values: ±1.There are several methods for dealing with multiple classes that use binary classification as: one versus the rest, pair-wise classification (one versus the one), error-correcting output coding and multiclass objective functions.

One of these methods consists in classifying "one versus the rest" in which elements that belong to a class are differentiated from the others. In this case we calculate an optimal hyperplane that separates each class from the rest of the elements. As the output of the algorithm we will choose the maximum value obtained from all decision functions.

To get a classification in M classes, we construct a set of binary classifiers where each of them is trained separate for one class versus the rest of classes. After that we combine them by doing the multi-class classification according to the maximal output before applying the *sign* function; that is, by taking

$$\arg\max_{j=1,..,M} g^j(x), \quad where\ g^j(x) = \sum_{i=1}^{m} y_i \alpha_i^j k(x, x_i) + b^j \tag{3.14}$$

and

$$f^j(x) = \mathrm{sgn}(g^j(x)) \tag{3.15}$$

where *m* represent number of pattern vectors. This problem has a linear complexity as for M classes we compute M hyperplanes.

Another method for multiclass classification, pair-wise classification, consists in training a classifier for each possible pair of classes. In this method two classes are chose and a hyperplane is computed for them. This is done for each pair of classes from the training set. For M classes, this results in $\frac{(M-1)*M}{2} = C_M^2$ binary classifiers. For M>3 this number is larger than the number of "one versus the rest" classifier having a polynomial complexity. For instance, if M=10, 45 binary classifiers need to be trained rather than 10 as in the method above. Although this suggests large training times, the individual problems that need to be trained are significantly smaller, and if the training algorithm scales super-linearly with the training set size, it is actually possible to save time. In my algorithm the advantage of this method is that usually the resulting hyperplane has a smaller dimension (less support vectors).

In the Error-correcting output coding, presented in [Die95] a binary case can be generated from a multiclass case by separating one class from the rest – digit 0 from digits 1 through 9 – a large number of feature binary cases can be generated by splitting the original set of classes into two subsets. For instance, the even digits can be separated from the odd ones, or digits 0 through 4 can be separated from digits 5 through 9. It is clear that if the set of binary classifiers $f^l,\ldots,f^L$ is

designed in the right way, then the binary response will completely determine the class of a test patterns.

The method multi-class objective functions is arguably one of the most elegant multi-class algorithm, and certainly the closest method aligned at Vapnik's principle of always trying to solve problems *directly*, entails modifying SVM objective function in such a way that it simultaneously allows the computation of a multi-class classifier. For instance [Bla95], equation 3.1 can be modified and used in the following quadratic program:

$$\underset{\mathbf{w}_r \in H, \xi^r \in \Re^m, b_r \in \Re}{\text{minimize}} \frac{1}{2}\sum_{r=1}^{M}\|\mathbf{w}_r\|^2 + \frac{C}{m}\sum_{i=1}^{m}\sum_{r \neq y_i}\xi_i^r \tag{3.16}$$

$$\begin{aligned} \textit{subject to } & \langle w_{y_i}, x_i \rangle + b_{y_i} \geq \langle w_r, x_r \rangle + b_r + 2 - \xi_i^r \\ & \xi_i^r \geq 0 \end{aligned} \tag{3.17}$$

where $m \in \{1,...,M\} \setminus y_i$ *and* $y_i \in \{1,...,M\}$ is the multiclass label for pattern x_i. In terms of accuracy, the results obtained using this approach is comparable to the results obtained with the widely used "one versus the rest" approach.

3.3 Clustering Using Support Vector Machine

Further on we will designate by classification the process of supervised learning on labeled data and we will designate by clustering the process of unsupervised learning on unlabeled data. The algorithm above only uses labeled training data. Vapnik presents in [Vap01] an alteration of the classical algorithm in which are used unlabeled training data. Here finding the hyperplane becomes finding a maximum dimensional sphere of minimum cost that groups the most resembling data (presented also in [Jai00]). This approach will be presented as follows. In [Sch02] we can find a different clustering algorithm based mostly on probabilities.

For document clustering we will use the terms defined above and we will mention the necessary changes for running the algorithm on unlabeled data.

There are more types of the kernels that can be used in the decision function. The most frequently used are the linear kernel, the polynomial kernel, the Gaussian kernel and the sigmoid kernel. We can choose the kernel according to the type of data that we are using. The linear and the polynomial kernel run best when the data is well separated. The Gaussian and the sigmoid kernel work best when data is overlapped but the number of support vectors also increases.

For clustering, the training data will be mapped in a higher dimensional feature space using the Gaussian kernel. In this space we will try to find the smallest sphere that includes the image of the mapped data. This is possible as data is generated by a given distribution and when they are mapped in a higher dimensional feature space they will group in a cluster. After computing the dimensions of the sphere this will be remapped in the original space. The boundary of the sphere will be transformed in one or more boundaries that will contain the classified data. The resulting boundaries can be considered as margins of the clusters in the input space. Points belonging to the same cluster will have the same boundary. As the width parameter of the Gaussian kernel decreases the number of unconnected boundaries increases, too. When the width parameters increases there will be overlapping clusters. We will use the following form for the Gaussian kernel:

$$\Phi(x, x') = e^{-\frac{\|x-x'\|^2}{2\cdot\sigma^2}} \tag{3.18}$$

where σ is the dispersion. Note if σ^2 decreases the exponent increases in absolute value and so the value of the kernel will tend to 0. This will map the data into a smaller dimensional space instead of a higher dimensional space. Mathematically speaking the algorithm try to find "the valleys" of the generating distribution of the training data.

If there are outliers the algorithm above will determine a sphere of very high costs. To avoid excessive costs and misclassification there is a soft margin version of the algorithm similar to that described in section 3.1. The soft margin algorithm uses a constant so that distant points (high cost points) won't be included in the sphere. In other words we choose a limit of the cost to introduce an element in the sphere.

Now, we will present in detail the clustering algorithm. We will also use some mathematical aspects to justify some of the assertions. Let $X \in \Re^n$ be the input space, $\{x_i\} \subseteq X$ the set of N samples to be classified, R the radius of the searched sphere and the Gaussian kernel presented in (3.18). We will consider *a* as being the center of the sphere. Given this the problem becomes:

$$\| \Phi(x_j) - a \|^2 \leq R^2, \forall j = 1...N \tag{3.19}$$

and by replaced the norm with the corresponding dot product we get:

$$< \Phi(x_j) - a, \Phi(x_j) - a > \leq R^2, \quad \forall j = 1...N \tag{3.20}$$

Equation (3.19) represents the primal optimization problem for clustering.

For the soft margin case, we will use the slack variables ξ_i and formulas (3.19) and (3.20) become:

$$\| \Phi(x_j) - a \|^2 \leq R^2 + \xi_j, \quad \forall j = 1...N \tag{3.21}$$

and respectively:

$$< \Phi(x_j) - a, \Phi(x_j) - a > \leq R^2 + \xi_j, \quad \forall j = 1...N \tag{3.22}$$

The equations above are equivalent forms of the same primal optimization problem using soft margin. The primal optimization problem is very difficult, if not impossible, to be solved in practice. This is why we will introduce the Lagrangian and the dual optimization problem. As above the solution of the dual problem is the same to the solution of the primal problem.

$$L = R^2 - \sum_j \left(R^2 + \xi_j - \left\| \Phi(x_j - a) \right\|^2 \right) \cdot \beta_j - \sum_j \xi_j \mu_j + C \sum_j \xi_j \tag{3.23}$$

where $\beta_j \geq 0, \mu_j \geq 0$ are the Lagrange multipliers, C is a constant and $C \sum \xi_j$ is a penalty term. In this case we have two Lagrange multipliers as there are more restrictions to be met.

Considering the partial derivates of L with respect to *R*, *a,* and ξ_j (primal variables) equal to 0 we get:

$$\frac{\partial L}{\partial R} = 2R - 2 \sum_j R \beta_j = 2R \left(1 - \sum_j \beta_j \right) = 0 \Rightarrow \sum_j \beta_j = 1 \tag{3.24}$$

$$\frac{\partial L}{\partial a} = - \sum_j \beta_j \Phi(x_j) + a = 0 \Rightarrow a = \sum_j \beta_j \Phi(x_j) \tag{3.25}$$

$$\frac{\partial L}{\partial \xi_j} = -\beta_j - \mu_j + C = 0 \Rightarrow \beta_j = C - \mu_j \tag{3.26}$$

Then the KKT restrictions are:

$$\xi_j \cdot \mu_j = 0 \tag{3.27}$$

$$\left(R^2 + \xi_j - \left\|\Phi(x_j) - a\right\|^2\right) \cdot \beta_j = 0 \tag{3.28}$$

Analyzing the problem and the restriction above we notice the following cases:

$\xi_i > 0$ and $\beta_j > 0 \Rightarrow \left\|\Phi(x_i) - a\right\|^2 > R$, which means that $\Phi(x_i)$ lies outside the sphere

$\mu_i = 0 \Rightarrow \beta_j = C \Rightarrow \left\|\Phi(x_i) - a\right\|^2 > R$, which means that x_i is a bounded support vector - BSV

$\xi_i = 0 \Rightarrow \left\|\Phi(x_i) - a\right\|^2 \le R$, which means these elements belong to the interior of the sphere and they will be classified when we remap them into the original space.

$\xi_i = 0$ and $0 < \beta_i < C \Rightarrow \left\|\Phi(x_i) - a\right\|^2 = R$, which means that the points x_i lie on the sphere. These points are actually the support vectors that will determine the clusters and that will then be used to classify new elements.

Thus the support vectors (SV) will lie exactly on the sphere and the bounded support vectors (BSV) will lie outside the sphere. We can easily see that when $C \ge 1$ there are no BSV, so all the points in the input space will be classified (the hard margin case).

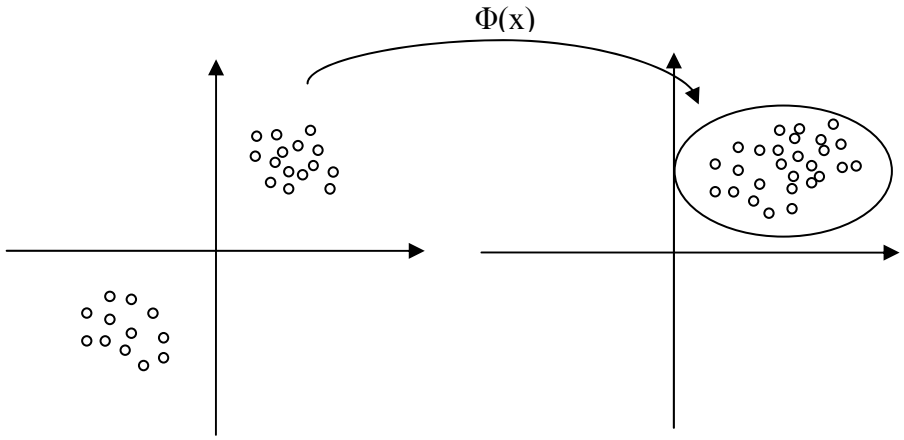

Figure 3.2– Mapping the unlabeled data from the input space into a higher dimensional feature space and determining the sphere that includes all points (hard margin)

In Figure 3.2 we presented the geometrical interpretation of the first part of the optimization problem: mapping the unlabeled data from the input space into a higher dimensional feature space via the Gaussian kernel function Φ. Thus we can determine the sphere that contains all the points to be classified. We considered the hard margin case. We will now present the mapping of the sphere in the input space to see the way we create the boundaries.

In Figure 3.3 we present the way data is grouped after mapping the boundary of the sphere from Figure 3.2 back in the input space.

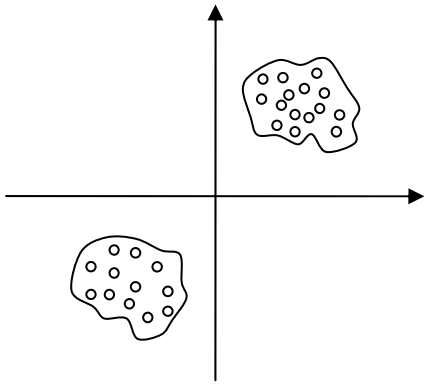

Figure 3.3 – Grouping data into classes

This is very difficult to implement as an algorithm. For an easier implementation we will eliminate R, a and β_j so that the Lagrangian becomes a Wolfe in the dual form. The obtained Wolfian so is an optimization problem in β_j. We will now replace in (3.23) formulas (3.25) and (3.26) and we will have:

$$W = R^2 - \sum_j \left(R^2 + \xi_j - \left\| \Phi(x_j) - \sum_i \beta_i \Phi(x_i) \right\|^2 \right) \cdot \beta_j - \sum_j \xi_j (C - \beta_j) + C \sum_j \xi_j \Rightarrow$$

$$W = R^2 - \sum_j R^2 \cdot \beta_j - \sum_j \xi_j \cdot \beta_j + \sum_j \left\| \Phi(x_j) - \sum_i \beta_i \Phi(x_i) \right\|^2 \cdot \beta_j - \sum_j \xi_j C + \sum_j \xi_j \beta_j + C \sum_j \xi_j \Rightarrow$$

$$W = R^2 \left(1 - \sum_j \beta_j \right) + \sum_j \left\| \Phi(x_j) - \sum_i \beta_i \Phi(x_i) \right\|^2 \cdot \beta_j$$

and based on (3.24) =>

$$W = \sum_j \left\| \Phi(x_j) - \sum_i \beta_i \Phi(x_i) \right\|^2 \cdot \beta_j \Rightarrow$$

$$W = \sum_j < \Phi(x_j) - \sum_i \beta_i \Phi(x_i), \Phi(x_j) - \sum_i \beta_i \Phi(x_i) > \cdot \beta_j \Rightarrow$$

$$W = \sum_j \left[< \Phi(x_j), \Phi(x_j) > - < \Phi(x_j), \sum_i \beta_i \Phi(x_i) > - < \sum_i \beta_i \Phi(x_i), \Phi(x_j) > + < \sum_i \beta_i \Phi(x_i), \sum_i \beta_i \Phi(x_i) > \right] \cdot$$

$\cdot \beta_j \Rightarrow$

$$W = \sum_j \left[< \Phi(x_j), \Phi(x_j) > + < \sum_i \beta_i \Phi(x_i), \sum_i \beta_i \Phi(x_i) > \right] \cdot \beta_j \Rightarrow$$

$$W = \sum_j < \Phi(x_j), \Phi(x_j) > \cdot \beta_j - \sum_{i,j} \beta_i \beta_j \Phi(x_i) \Phi(x_j) \qquad (3.29)$$

The restrictions for μ_j will be replaced by:

$$0 \le \beta_j \le C, \quad j = 1...N \qquad (3.30)$$

Using the support vector technique for representing the dot product $\langle \Phi(x_i), \Phi(x_j) \rangle$ by a Gaussian kernel:

$$K(x_i, x_j) = e^{-q\cdot\|x_i - x_j\|^2} \tag{3.31}$$

with the width parameter $q = \frac{1}{2\sigma^2}$ in our case.

In these conditions the Lagrangian becomes:

$$W = \sum_j K(x_j, x_j)\cdot \beta_j - \sum_{i,j} \beta_i \beta_j K(x_i, x_j) \tag{3.32}$$

We will denote the distance for each point *x* the distance from each image in the feature space to the center of the sphere as:

$$R^2 = \|\Phi(x) - a\|^2 \tag{3.33}$$

and by using (3.25) this becomes:

$$R^2 = \left\|\Phi(x) - \sum_j \beta_j \Phi(x_j)\right\|^2 \Rightarrow$$

$$R^2 = <\Phi(x) - \sum_j \beta_j \Phi(x_j), \Phi(x) - \sum_j \beta_j \Phi(x_j)> \Rightarrow$$

$$R^2 = <\Phi(x), \Phi(x)> - <\sum_j \beta_j \Phi(x_j), \Phi(x)> - <\Phi(x), \sum_j \beta_j \Phi(x_j)>$$
$$+ <\sum_j \beta_j \Phi(x_j), \sum_j \beta_j \Phi(x_j)> \Rightarrow$$

$$R^2 = <\Phi(x), \Phi(x)> - \sum_j \beta_j <\Phi(x_j), \Phi(x)> - \sum_j \beta_j <\Phi(x), \Phi(x_j)> + \sum_{i,j} \beta_i \beta_j <\Phi(x_j), \Phi(x_j)> \Rightarrow$$

$$R^2(x) = K(x,x) - 2\sum_j \beta_j K(x_j, x) + \sum_{i,j} \beta_i \beta_j K(x_i, x_j) \tag{3.34}$$

The radius will now be:

$$R = \{R(x_i) | x_i \in SV\} \tag{3.35}$$

which is:

$$R = \{R(x_i) | x_i \text{ with } \xi_i = 0 \text{ and } 0 < \beta_i < C\}$$

The boundaries that include points belonging to the same cluster are defined by the following set:

$$\{x | R(x) = R\} \tag{3.36}$$

Another issue to be discussed is assigning the cluster, which is how do we decided what points are in what class. Geometrically speaking this means determining the entering of the boundaries. When two points are in different clusters the line that unites them is transpose outside of the sphere in the higher space. This kind of line will contain one point *y* for which $R(y) > R$. In order to determine the existence of points *y* it is necessary to consider a number of test points to check if it goes outside the sphere or not. Let M be the number of test points and so the number of points being considered for testing if the line belongs or not to the sphere. The *n*-th test point will be:

$$y_n = \frac{n(x_i + x_j)}{M}, \quad n = \overline{1, M-1} \tag{3.37}$$

The results of these tests will be stored in an adjacency matrix A for each pair of points. The form of adjacency matrix will be:

$$A_{ij} = \begin{cases} 1, & \forall y \in x_i x_j, \ with\ R(y) \le R \\ 0, & otherwise \end{cases} \tag{3.38}$$

The cluster will be defined by the connected components of graphs induced by matrix A.

When we use the soft margin and thus have BSV, these will not be classified in any of the classes by this procedure. They can be assigned to the closest class or they can remain unclassified.

3.4 SMO - Sequential Minimal Optimization

There are a lot of methods for solving the optimization problems specific to Support Vector Machines presents in a large amount of code and number of publications. There are four classes of these algorithms after Scholkopf in [Sch02]; interior point codes, subset selection, sequential minimization and iterative methods. Interior point methods are some of the most reliable methods for moderate problem size, yet their implementation is not trivial. Subset selection methods act as meta-algorithms on top of basic optimization algorithm by carving out sets of variables on which the actual optimization takes place. Sequential minimal optimization is a special case thereof. Due to the choice of only two variables at a time the restricted optimization problem can be solved analytically which is obviating the need for an underlying base optimizer. Finally, iterative methods such as online learning, gradient descendent and Lagrangian Support Vector Machines gives a rough overview describing under which conditions which optimization algorithm is recommended. I present here only Sequential Minimal Optimization algorithm because it is the method I chose to implement in my application and because it can work with huge data sets.

This represents one of methods that implement a problem of quadratic programming introduced by Platt [Pla99_1]. The strategy of SMO is to break up the constraints into the smallest optimization groups possible. Note that it is not possible to modify variables α_i individually without modifying the sum of constraints (the KKT conditions). Therefore are generated a generic convex constrained optimization problem for pairs of variables. Thus, we consider the optimization over two variables α_i and α_j with all other variables considered fixed, optimizing the target function with respect to them. Here i and j are sample i and sample j from the training set. The exposition proceeds as follows: first we solve the generic optimization problem in two variables and subsequently we determine the value of the placeholders of the generic problem. We adjust b properly and determine how pattern can be selected to ensure speedy convergence.

Thus the problem presented above implies solving a linearly analytical optimization problem in two variables. The only problem now is the order that the training samples are chosen in for speedy convergence. By using two elements we have the advantage of dealing with the linear decision function which is easier to solve than a quadratic programming optimization problem. Another advantage of SMO algorithm is that not all training data must be simultaneously in the memory, but only the samples for which we optimize the decision function. This fact allows the algorithm to work with very high dimension samples.

We begin with a generic convex constrained optimization problem in two variables. Using the shorthand $K_{ij} := k(x_i, x_j)$ the quadratic problem becomes:

$$\begin{aligned} &\underset{\alpha_i,\alpha_j}{\text{minimize}} \frac{1}{2}\left[\alpha_i^2 K_{ii} + \alpha_j^2 K_{jj} + 2\alpha_i\alpha_j K_{ij}\right] + c_i\alpha_i + c_j\alpha_j \\ &\text{subject to} \quad s\alpha_i + \alpha_j = \gamma \\ &\qquad\qquad 0 \le \alpha_i \le C_i \quad \text{and} \quad 0 \le \alpha_j \le C_j \end{aligned} \tag{3.39}$$

where α_i and α_j are the Lagrange multipliers for those two samples, K is the kernel and s is +1 or -1. Here, c_i, c_j, $\gamma \in \mathrm{R}$, and $K \in \Re^{2\times 2}$ are chosen suitably to take the effect of the *m-2* variables that are kept fixed. The constants C_i represent pattern dependent regularization parameters. To obtain the minimum we use the restriction $\alpha_i + \alpha_j = \gamma$. This formula allows us to reduce actual optimization problem to a optimization problem in α_i by eliminating α_j. For shorthand we use the following notation $\chi := K_{ii} + K_{jj} - 2K_{ij}$.

In the implementation of SMO I take the value for the constants C_i equal to 1 because I have chosen hard margin in primal form of SVM. Constants c_i, c_j, and γ are defined thus:

$$c_i := \sum_{l\neq i,j}^{m} \alpha_l K_{il},\ c_j := \sum_{l\neq i,j}^{m} \alpha_l K_{jl} \text{ and } \gamma = 1 - \sum_{l\neq i,j}^{m} \alpha_l \tag{3.40}$$

As the initial condition I have considered all weight vectors equal to zero and the Lagrange multipliers α for all samples equal to zero. The dimension of the weight vector is equal to the number of attributes and the dimension of **α** vector is equal to the number of samples. To solve the minimization problem in the objective function I have created five sets that split the input data, according to KKT condition, as follows:

$$\begin{aligned} I_0 &= \{i \mid \alpha_i \in (0, C_i)\} \quad I_{+,0} = \{i \mid \alpha_i = 0, y_i = +1\} \quad I_{+,C} = \{i \mid \alpha_i = C_i, y_i = +1\} \\ &\qquad\qquad I_{-,0} = \{i \mid \alpha_i = 0, y_i = -1\} \quad I_{-,C} = \{i \mid \alpha_i = C_i, y_i = -1\} \end{aligned} \tag{3.41}$$

From now on I will use the following notations:

- m_I0 - all the training samples for which the Lagrange multipliers **α** are between 0 and C;
- m_I0_p - all the samples for which **α** = 0 and positive topic;
- m_I0_n - all the samples for which **α** = 0 and negative topic;
- m_IC_p - all the samples for which **α** = C and positive topic;
- m_IC_n - all the samples for which **α** = C and negative topic;

I have also defined:

$$\begin{aligned} m\_bUp &:= \min_{i\in m\_I0\cup m\_I0\_p\cup m\_IC\_n} f(x_i) - y_i \\ m\_bLow &:= \min_{i\in m\_I0\cup m\_I0\_n\cup m\_IC\_p} f(x_i) - y_i \end{aligned} \tag{3.42}$$

where m_bUp and m_bLow are the superior and the inferior limits for the margin of the hyperplane. As an improved estimation of *b* I have used the averaged formula $m\_b = \frac{(m\_bUp + m\_bLow)}{2}$. The stopping criteria is no longer that the KKT conditions are smaller than some tolerance *Tol*, but that m_bLow ≤ m_bUp, holds with some tolerance $m\_bLow \le m\_bUp - Tol$.

In the training part, the samples with **α** ≠ 0 are considered support vectors and are relevant in the objective function. They are the only ones that need to be kept from the training set. For this, I

have created a set called "m_supportVectors" that stores all the samples that have $\boldsymbol{\alpha} \neq 0$. In the developed application I have used a member "m_data" that keeps all the training samples from the file. Each sample has a specified position in "m_data". This position is the same in all the sets used to make the search easier. If the sets don't contain the sample on the specified position, the entry for that position contains a *null* value.

Considering the above facts I have put all the samples in the two sets, m_I0_p and m_I0_n, because for the beginning I have considered that $\boldsymbol{\alpha}$ and the weight vector equal to zero. Then I examined all the samples, renewing all the sets. If during this process there is no change I can consider that the training is over, otherwise I will repeat the process again as is presented below.

Regarding to the minimization problem, in order to find the minimum, we use $\alpha_i + \alpha_j = \gamma$. Modifying α_i depends on the difference between the values $f(x_i)$ and $f(x_j)$. There are some algorithms that can be used for choosing the index *i* and *j*, thus the larger the difference among them, the bigger is the distance to the hyperplane. I have chosen the two loops approach to maximize the objective function. The outer loop iterates over all patterns violating the KKT conditions, or possibly over those where the threshold condition of the m_b is violated. Usually we first loop over those with Lagrange multipliers neither on the upper nor the lower boundary. Once all of these are satisfied we loop over all patterns violating the KKT conditions, to ensure self consistency on the entire dataset. This solves the problem of choosing the index *i*.

It is sometimes useful, especially when dealing with noisily data, to iterate over the complete KKT violating dataset before complete self consistence on the subset has been achieved. Otherwise considerable computational resources are spent making subsets. The trick is to perform a sweep through the data once only less than, say 10% of the non bound variables change.

Now for selecting *j*: to make a large step towards the minimum, one looks for large steps in α_i. Since it is computationally expensive to compute $K_{ii} + K_{jj} - 2sK_{ij}$ for all possible pairs (*i,j*) one chooses an heuristic to maximize the change in the Lagrange multipliers α_i and thus maximize the absolute value of numerator in expressions for $\overline{\alpha}_i$ (new alpha). This means that we are looking for patterns with large difference in their relative errors $f(x_i) - y_i$ and $f(x_j) - y_j$. The index *j* corresponding to the maximum absolute value is chosen for this purpose.

If this heuristic happens to fail (if little progress is made by this choice) all other indices *j* are looked at in the following way (a second choice):

- All indices *j* corresponding to non-bound examples are looked at, searching for an example to make progress on.
- In the case that the first heuristic was unsuccessful, all other sample are analyzed until an example is found where progress can be made.
- If both previous steps fail, SMO precedes to the next index *i*.

In the problem of computing the offset b from the decision function another stopping criterion occurs. This new criterion occurs because, unfortunately, during the training, not all Lagrange multipliers will be optimal, since, if they were, we would already have obtained the solution. Hence, obtaining *b* by exploiting the KKT conditions is not accurate (because the margin must be exactly 1 for Lagrange multipliers for which the box constraints are inactive). Using the limits of the margin presented in (3.42) and since the KKT condition has to hold for a solution we can check that this corresponds to $m\_bUp \geq 0 \geq m\_bLow$. For m_*I0* I have already used this fact in the decision function. After that using KKT conditions I have renewed the thresholds "m_bUp" and "m_bLow" (superior and inferior limit for margin of the hyperplane). I have also chosen an interval from which to choose the second sample which I renew by renewing "m_iUp" and

"m_iLow". According to the class of the first sample I choose the second training sample as follows. If the first sample is in the set with positive topic I choose the second sample from the set with negative topic. After choosing the samples for training I check if there is a real progress in the algorithm. The real benefit comes from the fact that I may use m_iLow and m_iUp to choose patterns to focus on. The large contribution to the discrepancy between m_iUp and m_iLow stems from those pairs of patterns (i, j) for which the discrepancy:

$$discrepancy(i,j) := (f(x_i) - y_i) - (f(x_j) - y_j) \; where \begin{cases} i \in I_0 \cup I_{-,0} \cup I_{+,C} \\ j \in I_0 \cup I_{+,0} \cup I_{-,C} \end{cases} \tag{3.43}$$

is the largest. In order to have a real benefit the discrepancy needs to have a high value.

For the first sample I have computed the superior and inferior limit of α using:

$$\alpha_i = \begin{cases} L & \zeta > 0 \\ H & otherwise \end{cases} \tag{3.44}$$

Where:

$$\begin{aligned} L &= \begin{cases} \max(0, s^{-1}(\gamma - C_j)) & if\ s > 0 \\ \max(0, s^{-1}\gamma) & otherwise \end{cases} \\ H &= \begin{cases} \min(C_i, s^{-1}\gamma) & if\ s > 0 \\ \max(C_i, s^{-1}(\gamma - C_j)) & otherwise \end{cases} \\ \zeta &:= sc_j - c_i + \gamma s K_{jj} - \gamma K_{ij} \\ \chi &:= K_{ii} + K_{jj} - 2sK_{ij} \end{aligned} \tag{3.45}$$

After that I can compute the new α using the formula:

$$\overline{\alpha} = \begin{cases} \alpha_i^{old} + \chi^{-1}\delta & if\ \chi > 0 \\ -\infty & if\ \chi = 0\ and\ \delta > 0 \\ \infty & if\ \chi = 0\ and\ \delta < 0 \end{cases} \tag{3.46}$$

where $\delta := y_i((f(x_j) - y_j) - (f(x_i) - y_i))$, and $\overline{\alpha}$ is the value of the new alpha (the solution without restriction).

After that I have computed the new α for first sample and I checked and forced it to hold the KKT conditions (belonging between 0 and C). After that I have computed the new α for the second sample using the formula $\alpha_j = s(\alpha_i^{old} - \alpha_i) - \alpha_j^{old}$. I have modified all the sets using the new α. Thus for the first sample if the new α is greater than zero it is included in the "m_supportVector" set, otherwise it is eliminated if it is in set. If α is between 0 and C and the topic is positive this sample is included in "m_I0_p" otherwise it is eliminated from the set, etc. I repeated that with the second sample. After that I have updated the weight vector for all the attributes using the formula:

$$w = \sum_{i=1}^{m} \alpha_i y_i \mathbf{x}_i \tag{3.47}$$

Using the new objective function we can compute the classifier error for all samples, the inferior and the superior limit of the margin of the hyperplane *b,* and the inferior and superior margin of the domain for choosing sample "m_iUp" and "m_iLow".

To speed up the computing process, I have used in my program a large matrix called "m_store" to store the outputs for the kernels computed between two samples. The next time when the value is needed, I will take it from the matrix. I have put the value in this matrix after I have calculated the value of the kernel. I use the trick that the kernel's value for two variables is constant reporting to the evolution of variables from the algorithm.

There is a different approach of the algorithm for linear kernel and other kernels. In the case of the linear kernel the objective of the algorithm is to compute the weights from the initial decision function taking into consideration the primal optimization problem "$f(x) = sign(\langle \mathbf{w}, \Phi(x) \rangle + b)$".

The linear kernel is a particular case of the polynomial kernel, when the degree is 1. We take into consideration only the weight because with this type of kernel, the data is kept in the original space. In almost all cases this type of kernel, presented by some researchers separately, obtains poor classification results. Also it is used especially when the algorithm based on support vectors is used for feature selection because it produces a separate weight for each attribute. Other type of kernels used, like the polynomial kernel with the degree greater then 1 and the Gaussian kernel, are using the decision function $f(x) = \text{sgn}\left(\sum_{i=1}^{m} y_i \alpha_i k(x_i, x) + b\right)$ that comes from the dual optimization problem (where I compute only the Lagrange multipliers **α.**

3.5 Probabilistic Outputs for SVM

The standard SVM doesn't have an output to measure the classification's confidence. Platt in [Pla99_2] presents a method for transforming the standard SVM output into a posterior probability output. Posterior probabilities are also required when a classifier is making a small part of an overall decision, and the classification outputs must be combined for the overall decision. Thus the author presents a method for extracting probabilities P(class / input) from the output of SVM. These can be used for post processing classification. This method leaves the SVM error function unchanged. For classification in multiple classes, the class is chosen based on maximal posterior probability over all classes. The author follows the idea of [Wah99] for producing probabilistic outputs from a kernel machine. Support Vector Machine produce an un-calibrated value that is not a probability, and if signum function is applied it obtain only two values (-1 and 1).

$$f(x) = \text{sgn}\left(\sum_{i=1}^{m} y_i \alpha_i k(x_i, x) + b\right) \tag{3.48}$$

that lies in a Reproducing Kernel Hilbert Space. Training a SVM minimizes an error function that penalizes an approximation of the training misclassification rate plus a penalty term. Minimizing the error function is also minimizing a bound on the test misclassification rate, which is also a desirable goal. An additional advantage of this error function is that minimizing will produce a sparse machine where only a subset of possible kernels is used in the final machine.

$$P(class|input) = P(y = 1|x) = p(x) = \frac{1}{1 + \exp(-f(x))} \tag{3.49}$$

where *f*(x) is the output function for SVM. Instead of estimating the class-conditional densities $P(f|y)$, the authors suggest using a parametric model to fit the posterior probability $P(y = 1|f)$ directly. The parameters of the model are adapted to give the best probability outputs. The authors analyze the empirical data and show that the density is very far away from Gaussian. There are

discontinuities in the derivates of both the densities and the positive margin f=1 and the negative margin f=-1. These discontinuities occur because the cost function also has discontinuities at the margins. The class-conditional densities between the margins are apparently exponential and the authors suggest the usage of a parametric form of the sigmoid:

$$P(y=1|f)=\frac{1}{1+\exp(Af(x)+B)} \tag{3.50}$$

This sigmoid model is equivalent to assuming that the output of the SVM is proportional to the logarithmical probabilities of positive samples. This sigmoid model has two parameters which are trained independently in an extra post processing step. They are trained using the regularized binomial likelihood. As long as A < 0, the monotony of (3.48) is assured. The parameters A and B from (3.50) are found using maximum likelihood estimation from the training set (f_i, y_i). The complete algorithm for computing the parameters A and B is presented in [Pla99_2]. Also this method was used by Kai in [Kai03] using only a single parameter.

3.6 Types of Kernels

The purpose of the kernel is to transpose the training data from the input space in a higher dimensional feature space and to separate the data in this new space. The idea of the kernel is to compute the norm of the difference between two vectors (the cosine of the angle between the two vectors) in a higher dimensional space without representing those vectors in the new space (kernel trick). In order to use the kernel trick we need to express the kernel in terms of dot products. By adding a scalar constant to the kernel we can get better classifying and clustering results. In this PhD thesis I tested a new idea to correlate this new scalar with the dimension of the space where the data will be represented because I consider that those two parameters (the degree and the scalar for polynomial kernel or number of elements and parameter C for Gaussian kernel) need to be correlated.

In order to demonstrate the improvement introduced by this correlation I will present in chapter 5 the results for different kernels and for different parameters of each kernel. For the polynomial kernel I change the degree of the polynomial and for the Gaussian kernel I change the parameter C according to the following formulas:

- Polynomial: $k(x,x')=\left(2\cdot d+\langle x\cdot x'\rangle\right)^d$ - d being the only parameter to be modified.
- Gaussian (radial basis function RBF): $k(x,x')=\exp\left(-\frac{\|x-x'\|^2}{n\cdot C}\right)$ - C being the only parameter to be modified and n being the number of elements greater that 0 from the input vectors.

In both formulas x and x' being the input vectors.

Another interesting kernel used in the literature is linear kernel. For the linear kernel I used the polynomial kernel with degree 1. In my tests I used the linear kernel also in the classification step as a polynomial kernel with degree = 1 and especially in the feature selection step where was used as a method for compute the weight vector **w**.

3.7 Correlation of the SVM Kernel's Parameters

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. Here I want to present a new idea to correlate this scalar with the dimension of the space where the data will be represented. Thus I consider that those two parameters (the degree and the scalar) need to be correlated.

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

Polynomial kernel:

$$k(\mathbf{x},\mathbf{x}') = \left(\mathbf{2}\cdot d + \langle \mathbf{x},\mathbf{x}'\rangle\right)^d \tag{3.51}$$

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

Gaussian kernel:

$$k(\mathbf{x},\mathbf{x}') = \exp\left(-\frac{\|\mathbf{x}-\mathbf{x}'\|^2}{n\cdot C}\right) \tag{3.52}$$

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.

3.7.1 Polynomial Kernel Parameters Correlation

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

$$k(\mathbf{x},\mathbf{x}') = \left(\langle \mathbf{x}\cdot\mathbf{x}'\rangle + b\right)^d \tag{3.53}$$

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 – usually is chosen as equal with the number of features. I notice that if this parameter was eliminated (i.e., chosen zero) the quality of results can be poor. It is logical that I 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 section 5, I suggest using "*b=2*d"* in my application.

3.7.2 Gaussian Kernel Parameters Correlation

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

$$k(x,x') = \exp\left(-\frac{\|x-x'\|}{C}\right) \tag{3.54}$$

where the parameter C is a number representing the vectors' dimension from the training set. This is usually a very big number in the text classification problems – being equal with the number of the features. These makes vectors' dimension being quite large and sparse. Using the same large dimension (the same C) for all vectors can lead to poor results. Based on running laborious classification simulations I suggest a correlation between the parameter C and the real dimension of the current input vectors. This is why I introduced 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 a new parameter noted also C. I kept the notation C for a parameter that becomes a small number (usually I obtain best results between 1 and 2).

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

In order to become someone you have to start by being nothing."

Honoré de Balzac

4 Feature Selection Methods Developed

4.1 Data Reduction

Text categorization is the problem of automatically assigning predefined categories to free text documents. The major problem is the high dimensionality of the feature space. Automatic feature selection methods include the removal of non-informative terms according to corps statistics and the construction of new features which combine lower level feature into higher level orthogonal dimensions. In feature selection the focus consists in an aggressive dimensionality reduction of the feature vector.

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 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 consist in 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, which maximize the classification performance of a 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!

In the obtained set of 7053 samples I have 19038 distinctive attributes (features) that represent in fact the root of the words from the documents. A training algorithm with a vector of this dimension is time consuming and in almost all cases the accuracy of the learning is low due to the noise in data (as we will further see in this work). In the learning phase, also data needs to be stored in the memory in order to analyze it and so the memory needed is considerably huge. I have applied in this work some techniques to reduce the dimension of this large features vector. For doing so I have used the *Information Gain* [Yan97] [Cov91], Support Vector Machine [Nel00] [Dou04] and also Genetic Algorithm in combination with Support Vector Machine technique. I tested also the Random selection to see in that I can obtain good results with sample methods and justify the necessity of using sophisticated feature selection methods.

In the literature there are another type of methods used for features induction that automatically create a nonlinear combination of existing features and additional input features to improve classification accuracy like the method proposed by [Jim04]. But all methods use the elimination

of tokens which occurs less than the preordain threshold. As follows I'll preset four methods of feature selection that I'll further use in my work. All feature selection methods use as a starting point the same vectors obtained after the extraction step.

4.2 Random Selection

This method was tested due to its simplicity. It is a maladjusted method that was tested to justify the necessity of using more powerful selection methods. In the Random feature selection method random weights between 0 and 1 are assigned to each feature. Then training and testing sets of various sizes are chosen by selecting the features according to their descending weights. These sets (with various sizes) are generated so that the larger sets are containing the smaller sets. This is because in order to generate the larger sets are used smaller thresholds and in order to generate the smaller sets, greater thresholds are used. With this obtained sets I make a classification step in order to have the accuracy of classification. The accuracy is computed as a percentage of documents correct classified from all documents from the testing set. Because the features were selected randomly I repeat this process (of assigning weights, selecting features and classifying datasets) three times. The classification accuracy considered as been obtained with this feature selection method is computed as the average over all three obtained individual accuracies for each dimension of the dataset.

4.3 Entropy and Information Gain

The measure named *Entropy Based Discretization* is a measure commonly used in information theory, which characterizes the (im) purity of an arbitrary collection of samples, being a measure of homogeneity of samples. The entropy and information gain are functions of the probability distribution that underlie the process of communications. The entropy being a measure of uncertainty of a random variable can be used to recursively partition the values of a numeric attribute.

Given a collection S of *n* samples grouped in *c* target concepts (classes), the entropy of S relative to the classification is:

$$Entropy(S) = -\sum_{i=1}^{c} p_i \log_2(p_i) \tag{4.1}$$

where p_i is the fraction of S belonging to class *i*. In all calculations involving entropy I define $0*\log_2 0$ to be 0. Note that the entropy is 0 if all members of S belong to the same class, and the entropy take the maximum ($\log_2 c$) when the samples are equally distributed to classes. If the samples, in the collection, are unequally distributed in classes the entropy is between 0 and $\log_2 c$.

This definition of entropy is related to definition of entropy in thermodynamics. It is possible to derive the definition of entropy axiomatically by defining certain properties that the entropy of a random variable must satisfy. Also the concept of entropy in information theory [Cov91] is closely connected with the concept of entropy in statistical mechanics. If we draw a sequence of *n* independent and identically distribution random variables, it can be shown that the probability of a "typical" sequence is about $2^{-nE(X)}$ and that there are about $2^{nE(X)}$ such "typical" sequences. This property (known as the asymptotic equation property) is the basis of many of the proofs in information theory.

One interpretation of entropy from information theory is that it specifies the minimum number of bits of information needed to encode the classification of an arbitrary member of S. If p_i is 1, the receiver knows the drawn sample will be positive, so no message needs to be sent, and the entropy is zero. On the other hand, if p_i is *1/c*, $\lceil \log_2 c \rceil + 1$ bits are required to indicate whether the drawn sample is in what class ("[x]" represents the integer part of the real number x). For example in the two classes case if p_i is 0.8, then a collection of messages can be encoded using an average less than 1 bit per message by assigning shorter codes to collections of positive samples and longer codes to less likely negative samples. The logarithm is still base 2 because entropy is a measure of the expected encoding length measured in bits. If the target attribute can take *c* possible values, the entropy can be as large as $log_2 c$.

Entropy represents the expected minimum number of bits needed to encode the class of a randomly drown sample from S. Therefore, entropy is a measure of the impurity in a collection of training samples. Using entropy an attribute effectiveness measure is defined in classifying the training data. The measure is called *information gain*, and is simply the expected reduction in entropy caused by partitioning the samples according to this attribute. More precisely, the information gain of an attribute relatively to a collection of samples *S*, is defined as:

$$Gain(S, A) \equiv Entropy(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} Entropy(S_v) \qquad (4.2)$$

where *Values(A)* is the set of all possible values for attribute A, and S_v is the subset of *S* for which attribute A has the value v. The first parameter is just the entropy of the original collection S, and the second term is the expected value of the entropy after S is partitioned using attribute A. In other words, the information gain is therefore the expected reduction in entropy caused by knowing the value of attribute A. The information gain is the number of bits saved when encoding the target value of an arbitrary member of S, by knowing the value of attribute A.

Using equation 4.2 for each feature is computed the information gain obtained if the set is split using this feature. The obtained values are between 0 and 1 being closer to 1 if the feature splits the original set in two subsets with almost the same dimensions. For selecting relevant features I use different thresholds. If the information gain obtained for a feature exceeds the threshold I will select it as being relevant, other way I will not select it.

In [For04] the author justified that Information Gain failed to produce good results on an industrial text classification problem. The author says that for a large class of features scoring methods suffers a pitfall: they can be blinded by a surplus of strongly predictive features for some classes, while largely ignoring features needed to discriminate difficult classes.

4.4 Feature Subset Selection Using SVM

In [Mla04], Mladenic et al. a method for selecting features based on linear support vector machine (SVM) is presented. For tests the author used a particular implementation for SVM called SVMLight [Joa99]. The authors compare more traditional feature selection methods, such as Odds Ration and Information Gain with a new method based on SVM, in achieving the desired tradeoff between the vector sparseness and the classification performance. The results indicate that at the same level of sparseness, feature selection based on normal SVM yields better classification performances. First the authors train the linear SVM on a subset of training data and retain only those features that correspond to highly weighted components (in the absolute value sense) of the resulting hyperplane that separates positive and negative samples. The reduced feature space is

then used to train a classifier over a large training set because more documents now fit into the same amount of memory. A close idea was also presented in [Dou04].

In [Jeb00] and [Jeb04] are explained the advantages of using the same methods in the features selection step and in the learning step. Also some usually used methods into literature failed to produce good results on an industrial text classification problems, as Reuter's database. For example Forman in [For04] reported that Information Gain produces bad results when working with huge collections of text documents. The author attributed this to the property of many feature scoring methods to ignore or to remove features needed to discriminate difficult classes.

Based on this idea I have used my implemented SVM algorithm with linear kernel in the step of features selection. I wasn't interested in reduce the level of sparseness for the input vectors as Mladenic. They were interested in the advantage obtained by freeing memory as a result of increasing data sparsity in order to work with larger training sets keeping the memory consumption constant. I use feature selection method in order to find an optimal reduced set, decreasing the training time and increasing the accuracy of classification / clustering. In this method I was interested in finding the weight vector that characterizes the hyperplane that separate positive and negative samples from the training set. For this I use the following formula for the linear kernel:

$$k(w,x) = \left(2 + \langle w \cdot x \rangle\right) \tag{4.3}$$

where **w** represents the weight vector perpendicular to the separation hyperplane and *x* represent the input vector (sample). I add at the kernel the constant 2 because I want that this kernel to use the correlation of the parameters presented in 3.7 and as I showed this correlation produces good results. Because I use more than two topics, for each topic a decision function (equation 3.13) using the linear kernel presented above needs to be learn. The SVM algorithm assure that after the training step the features that have a great influence in designing the hyperplane will have a great weight absolute value mode and the features that have no influence in designing the hyperplane will have a weight closer to 0. Thus the feature selection step becomes a learning step that trains over all the features (attributes, in my case 19038) and tries to find the hyperplane that splits the best the positive and the negative samples. This hyperplane is computed for each topic (I my case 24 topics) separately. The resulting weight vector in the decision function has the same dimension as the feature space because I worked only in the input space. The final weight vector is obtained as a sum over all weight vectors obtained for each topic separately normalized between 0 and 1. Using this weight vector I select only the features with an absolute value of the weight that exceeds a specified threshold. The resulting set has a smaller dimension, from features point of view. The resulting sets are then used in the learning and testing steps of the algorithm for classifying data and computing the classification accuracy.

I use different values for the threshold, ranging from 0 to 1. In order to have a good comparison between this method and other used methods I can use 2 different thresholds. If the threshold represents a value between 0 and 1 it means that I want to select all values that have the weight greater than the specified threshold. If the threshold has a value greater than 1 it means that I want to obtain a specified number of features according with the descending weights. This was introduces in order to simplify the specification of the number of features that we want to obtain.

The results in comparison with other classical feature selections are presented at the end of this chapter. Some experimental results were also presented in [Mor06_2] and [Mor06_6]. In [Jur03] the authors go further on to grammatical semantic parsing using support vector machine adding different weights to words appearing in different parts of the phrase.

4.5 Features Selection Using Genetic Algorithms

In this section I am introducing a feature selection method, which can minimize most of the problems that can be found in the conventional approaches, by applying genetic algorithms. 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. I combined a powerful and rigorous mathematical method based on kernels with a randomly starting point permitted by genetic algorithm. This method was also presented in [Mor06_3] and [Mor06_5].

4.5.1 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 chosen 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 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 stochastically selected and the new chromosomes are formed by three essential operations (genetic operators): 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.

4.5.1.1 Chromosomes Encoding and Optimization Problems

Usually there are only two main components of most genetic algorithms that are problem dependent: the encoding problem and the function's evaluation. The chromosome should contain

in some way 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 a function that allows giving a confidence to each chromosome from the population. In my representation I chose value encoding.

I 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 I take into consideration all entries from the input vector and I try to eliminated those entries that are considered irrelevant for this classification. For the fitness function in the genetic algorithm I used 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_2] and in the section 3. 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 better solutions. The general scheme of the implemented genetic algorithm is presented in Figure 4.1.

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:

$$c = (w_1, w_2, ..., w_n, b) \tag{4.4}$$

where w_n, $n = 1,2,..,19038$ 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.

I 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) I make, in my 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 4.4). In each chromosome I put only half number of features, chosen randomly, different from 0. I make this because in the feature selection step I am 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

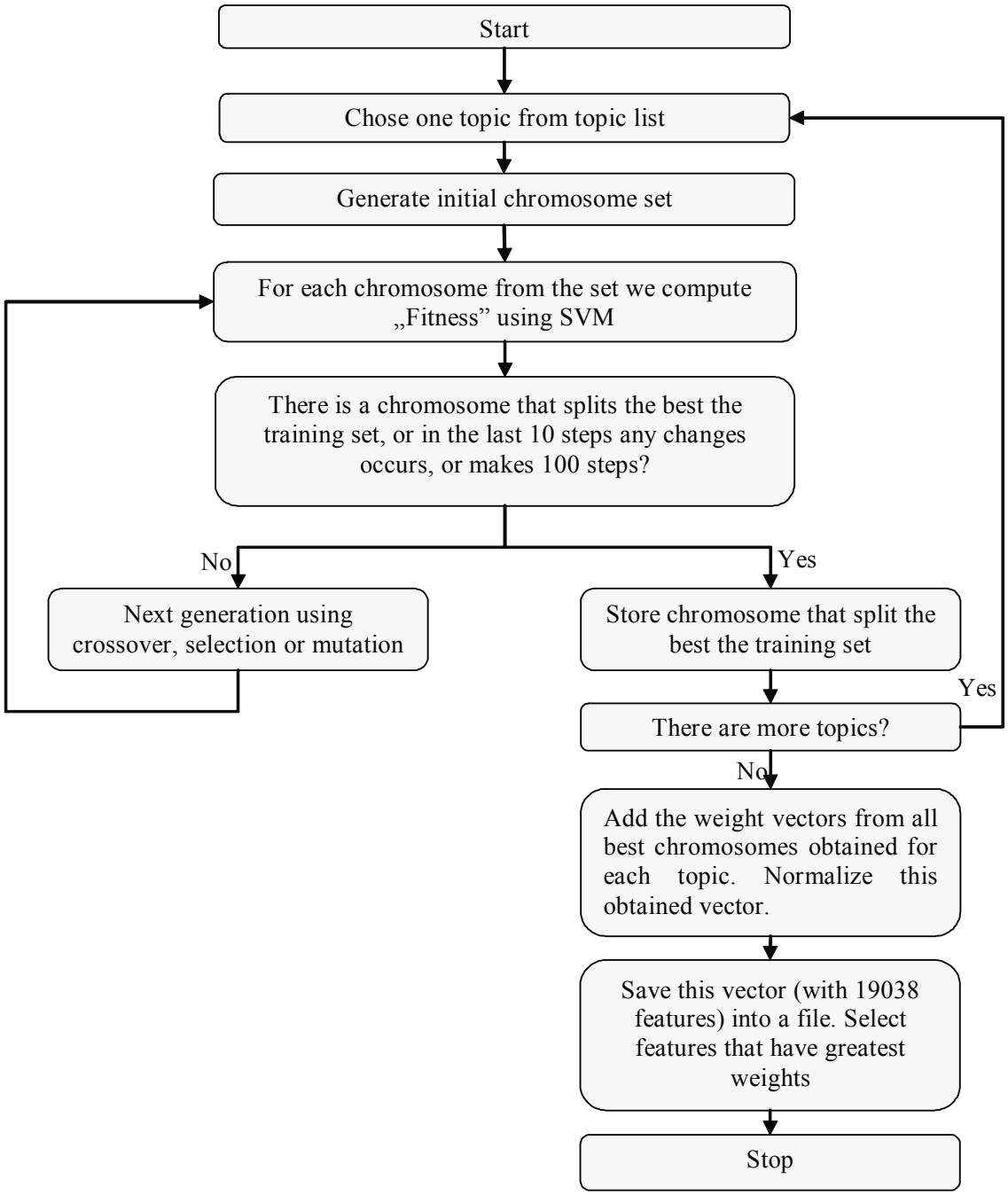

Figure 4.1 – The adapted Genetic Algorithm

current topic, I 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:

$$f(c) = f((w_1, w_2, ..., w_n, b)) = \langle \mathbf{w}, \mathbf{x} \rangle + b \tag{4.5}$$

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 4.2. Into the step of generating the next population those

operators are chosen randomly for each parent. I 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.

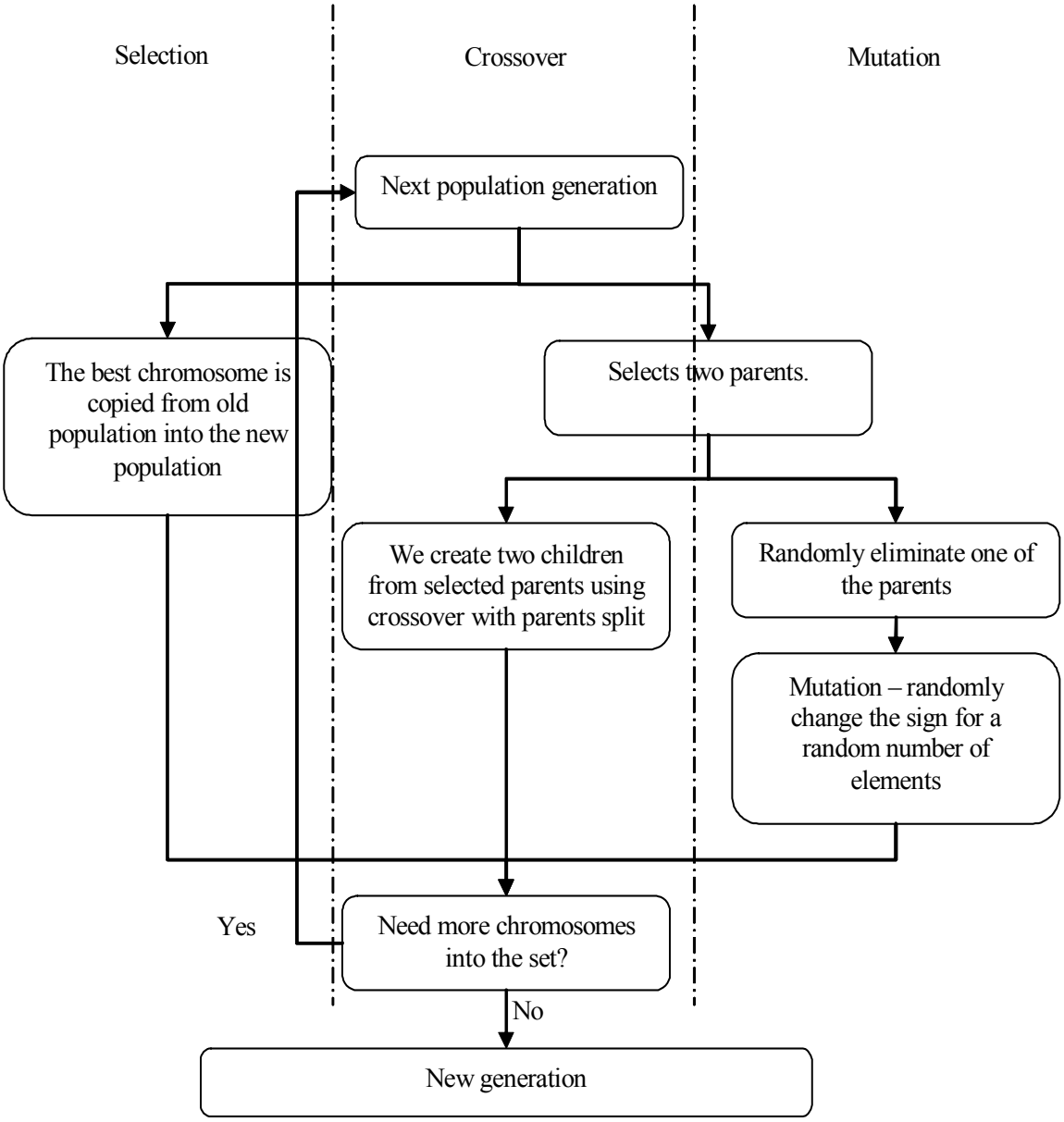

Figure 4.2 – Obtaining the next generation

From the initial population I select two parents using two methods: Gaussian selection or Roulette Wheel [Whi94] selection. These methods will be explained later. Because I want to randomly select one of the operators to generate the next candidates, I randomly delete one of parents with a small probability to do this. Thus with a small probability I can have only selection or mutation and with a greater probability to have crossover. If in the new generation I find a chromosome that best splits (without error) the training set I stop and consider this chromosome as the decision function for the current topic. If not, I generate a new population and stop when in the last 10 generations no evolution occurs.

At the end of the algorithm, I obtain for each topic from the training set a distinct chromosome that represents the decision function; I normalize each of the weight vectors meaning 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 I take the weight vector and select only the features with an absolute value of the weight that exceeds a specified threshold.

4.5.1.2 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. If we generate the new population only by new offsprings, this might lose 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 can survive at the end of the algorithm.

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 more chances to be selected for the most efficient chromosomes. 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 other chromosomes.

In the Gaussian selection method we randomly choose a chromosome from the current generation. Using equation (4.6) we compute the probability that the chosen chromosome is the best chromosome (that obtains the fitness function with maxim value *M*).

$$P(c_i) = \exp\left(-\frac{1}{2}\left(\frac{fitness(c_i) - M}{\sigma} \right)^2 \right) \tag{4.6}$$

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 σ 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.

4.5.1.3 Using Genetic Operators

4.5.1.3.1 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 to not 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 Gaussian methods in almost all cases and rarely using Roulette Wheel method.

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

4.5.1.3.2 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 chromosome's encoding. 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 the 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 or 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**

4.6 Feature Subset Selection. A Comparative Approach

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 I 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_1]. This method was also presented in details in the second PhD report [Mor05_2].

In text classification problems, usually we have a great number of features that are extracted from the dataset in order to create the input vectors. Usually many of those features are rather irrelevant in the classification process. These features generally don't improve classification accuracy sometime actually decrease but increase the training and testing time and the memory requirement. These features are generally considered noise and some methods for reducing the number of these features are need to be used.

In this section a comparative study of features selection methods used priory to documents classification is presented: Random Selection (depict RAN), Information Gain (depict IG), SVM as feature selection method (depict SVM_FS) and Genetic algorithm with SVM fitness function (depict GA_FS). I am also studying the influence of input data representation presented in section 2.4.5, on classification accuracy. I have used three types of representation: Binary (note as Bin), Nominal (note as Nom) and Cornell Smart (noted as CS). The accuracy is computed as percent of correct classified samples from all samples from the testing set. As training and testing data set I use the sets presented in section 2.4.

For a fair comparison between the presented feature selections methods, I need to use the same number of features. For the Information Gain method the threshold for selecting the relevant features represents a value between 0 and 1. For the other three presented methods the threshold represents the number of features that we want to be obtained. This number must be equal with the number of features obtained through Information Gain method. If after the feature selection step we obtain input vectors (samples that characterize a document) that have for all selected features the value equal to zero (those samples haven't representation into the new feature set) we eliminated those samples from the sets.

4.6.1 Data Set Dimension's Influence

In what follows I am presenting the influence of the number of features regarding to the classification accuracy for each input data representation and for each feature selection method, considering 24 distinct classes (topics). I present here the results for some of the values of degree for polynomial kernel and for some of the values of parameter *C* for Gaussian kernel because I am interested to show the influence of the feature selection methods only. In next chapter will be presenting more results for different kernel parameters. Here I chose to present results only for parameters that offer the best results.

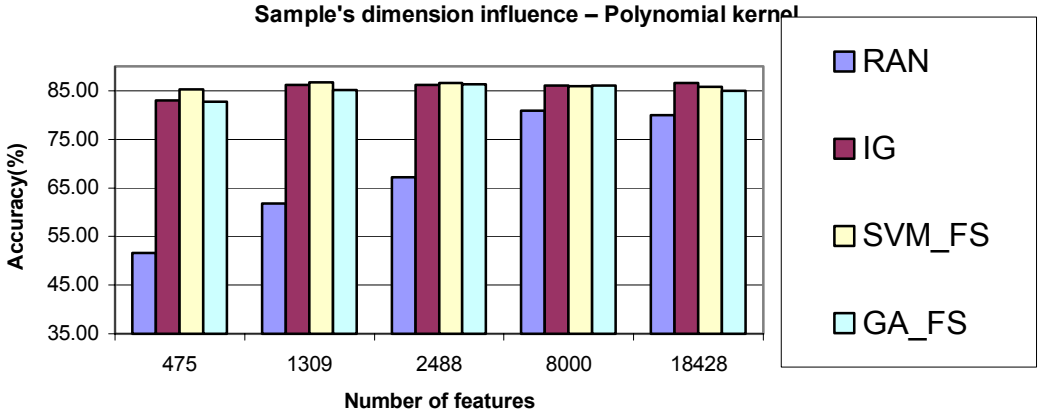

Figure 4.3 - Influence of the number of features on the classification accuracy using Polynomial kernel with degree equal to 2 and Binary data representation

In Figure 4.3 are presented results obtained for polynomial kernel with degree equal with 2 and binary data representation. For a small number of features the best results were obtained by SVM_FS and when the number of features increased the IG method obtained better results.

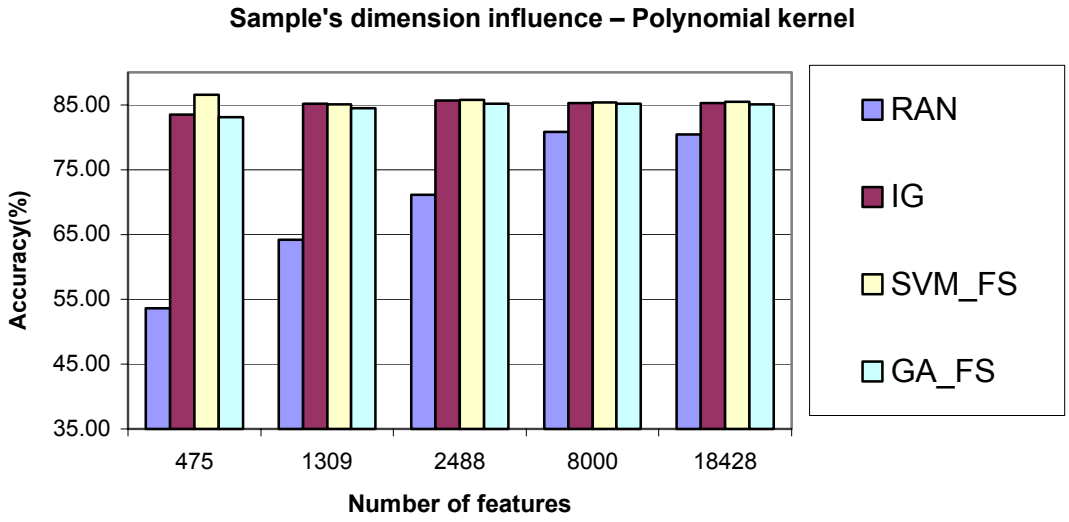

Figure 4.4 - Influence of the number of features on the classification accuracy using Polynomial kernel with degree equal to 2 and Nominal data representation

In Figure 4.4 results obtained for polynomial kernel with degree 2 and Nominal data representation are presented. For this type of data representation SVM_FS method obtain all the time the best results. An interesting observation is that the best results (86.52%) are obtained for a small number of features (475). In Figure 4.5 are presented classification accuracy results using polynomial kernel with degree 2 and Cornell Smart data representation.

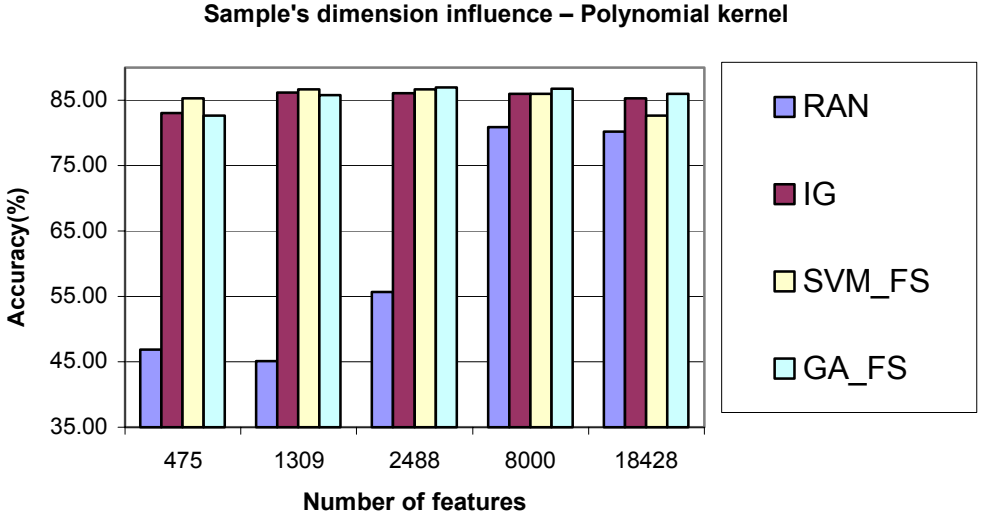

Figure 4.5 - Influence of the number of features on the classification accuracy using Polynomial kernel with degree equal to 2 and Cornel Smart data representation

The classification performance, as it can be observed from Figure 4.3, Figure 4.4 and Figure 4.5 is not improved when the number of features increases (especially for SVM_FS). I notice that there

is a slight increase in the accuracy when I raise the percentage of features from the initial set from 2.5% (475 features) to 7% (1309 features) for polynomial kernel. An interesting observation is that the accuracy doesn't increase when the number of selected features increases for all tested methods. More than this, if more than 42% of the features were selected, the accuracy slightly decreases for majority of feature selection methods. This can occur because the additional features can be noisy for the current classification process. As I expected, for Random features selection the value of the accuracy is very poor comparing with the other methods. The other methods, Information Gain, SVM_FS and GA_FS obtained comparable results. SVM_FS has slightly better results in comparison with IG for polynomial kernels and obtains best results using a small number of features. Working with the entire feature set I obtain no increases in accuracy; sometimes, obtain some decreases.

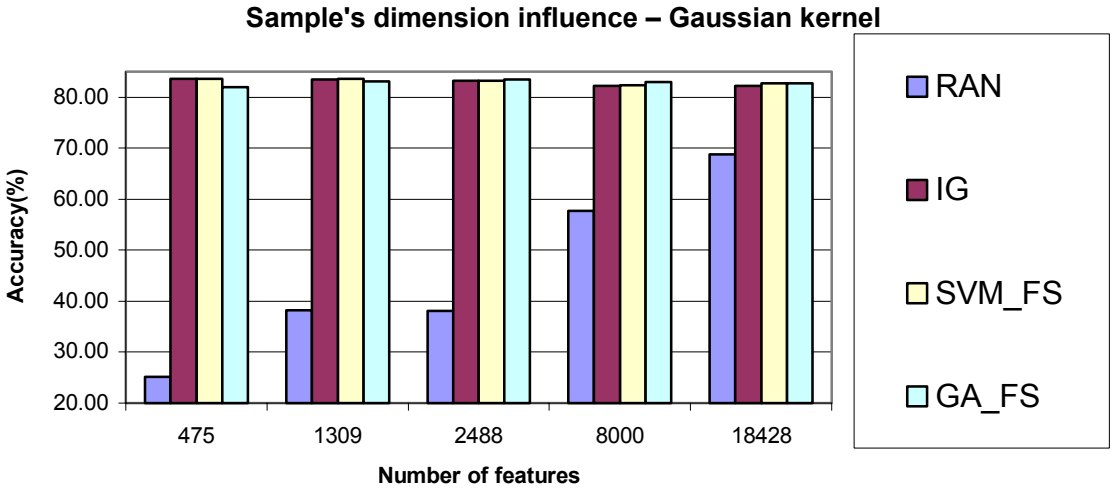

Figure 4.6 - Influence of the number of features on the classification accuracy using Gaussian kernel with parameter *C* equal to 1.3 and binary data representation

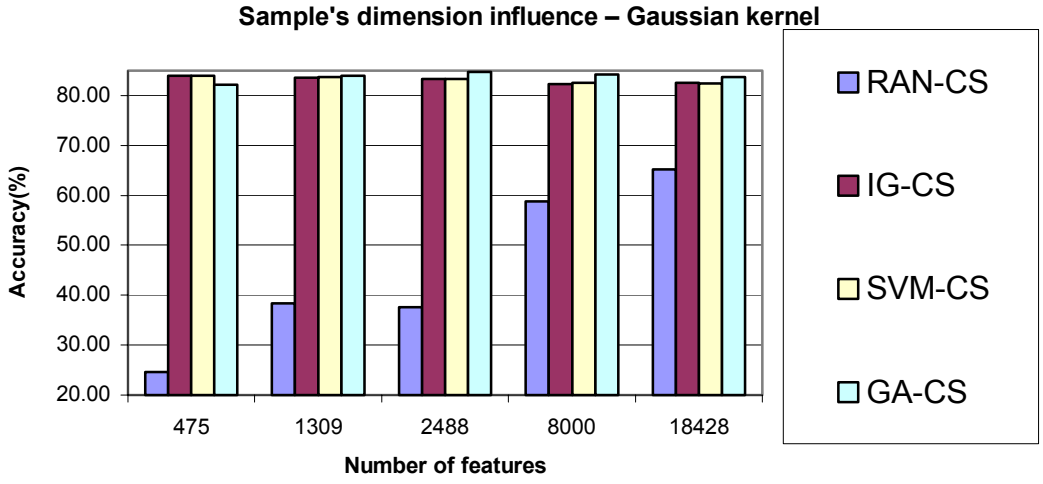

Figure 4.7 - Influence of the number of features on the classification accuracy using Gaussian kernel with parameter *C* equal to 1.3 and Cornell Smart data representation

In that follow I am presenting results obtained using a Gaussian kernel in the classification step with parameter *C* equal with 1.3. I will show in next chapter that for this parameter the best results

are obtained. There are presented results obtained with binary data representation (Figure 4.6) and Cornell Smart data representation (Figure 4.7). Here are not presented results obtained with Nominal representation because usually very poor classification results are obtained.

The SVM algorithm depends on the order of selecting input vectors, finding different optimal hyperplanes when the input data are selected in different order. Genetic algorithm with SVM fitness function stipulates this in the feature selection step. The SVM_FS and GA_FS obtained comparable results but they are better comparatively with Information Gain. As can be observed GA_FS obtain better results with Gaussian kernel comparatively with the other three methods. So, GA_FS is better at average with 1% comparatively with SVM_FS (84.27% for GA_FS comparatively with 83.19% for SVM_FS) and with 1.7% comparatively with IG (84.27% for GA_FS and 82.58% for IG). Polynomial kernels SVM_FS obtain at average better results with 0.9% comparatively with IG (from 86.24% for SVM_FS to 85.31% for IG) and with 0.8% comparatively with GA_SVM (from 86.24% to 85.40%). In almost all cases the best results are obtained for a small numbers of features (in average for 1309).

In the Figure 4.8 the training classification time as a function of selected numbers of features for each feature selection method and polynomial kernel with degree 2 is presented. As it can be observed, the timing increases with about 3 minutes when the features increases from 475 to 1309 and with about 32 minutes when the features increases from 1309 to 2488 for SVM_FS method. Also the time needed for training with features selected using IG is usually greater than the time needed for training with features selected with SVM_FS. When number of selected features increases the best classification time was obtained of GA_FS method.

For Gaussian kernel the time is at average (for all tests) with 20 minutes greater then time needed for training polynomial kernel for all features selected with IG, SVM_FS or GA_FS. The numbers are given for a Pentium IV at 3.4 GHz, with 1GB DRAM and 512KB cache, and WinXP.

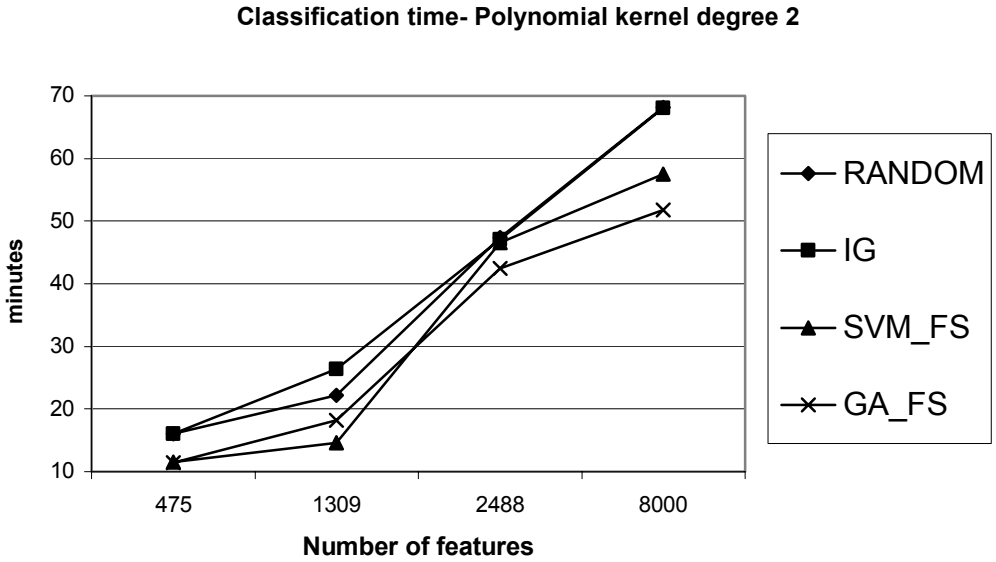

Figure 4.8 - Learning classification time function of selected numbers of features

4.6.2 Polynomial Kernel Degree's Influence

In what follows I am presenting some results obtained for polynomial kernel and nominal data representation for different kernel degree (from 1 to 5). There are also presented some results

obtained for Gaussian kernel for different parameter *C* (1.0, 1.3, 1.8, 2.1, and 2.8) and Binary data representation. For polynomial kernel I choose to present results only for nominal data representation here because as it will be shown in section 5.6 it obtains at average the best results. Also Gaussian kernel with binary data representation obtains at average the best results. In Figure 4.9 are presented results obtained for a set with 475 features selected with all presented methods and polynomial kernel.

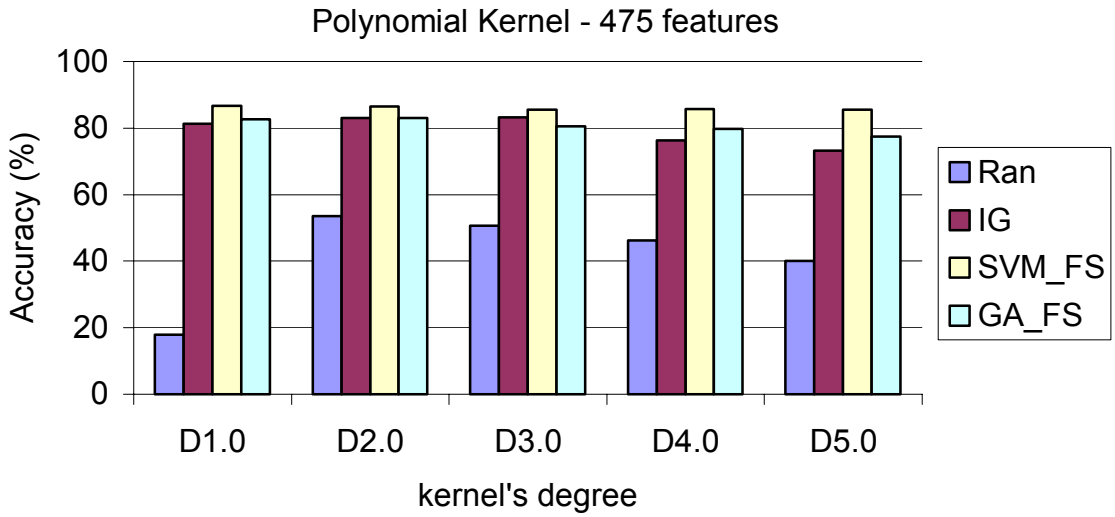

Figure 4.9 – Influence of feature selection method and kernel degree on classification accuracy

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 most documents are linearly separable. For example the best value obtained for this dimension of the set was 86.64% for SVM_FS and degree equal to 1. For this number of features SVM_FS method obtain all the time the best results. Also GA_FS obtain comparable results. IG obtains good results for small value of degree but obtained accuracy decrease when the kernel degree increases.

In Figure 4.10 are presented results obtained when the dimension of the vector increases to 1309 features. I showed that for this dimension the best results were obtained and on average for all tested degrees the best results also was obtained. 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. However, for this dimension 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. The Information Gain method achieved an average accuracy (computed for all three types of data representation) of 84.62% in comparison with the SVM_FS method that achieved an average accuracy of 84.90% and GA_FS that obtain only 83.91% (for Random selection method an average accuracy of 51.02% was achieved). Figure 4.10 shows also that text files are generally linearly separable in the input space (if the input space has the right dimensionality) and the best results were obtained for a linear kernel and for a small kernel degree using a nominal representation of the input data.

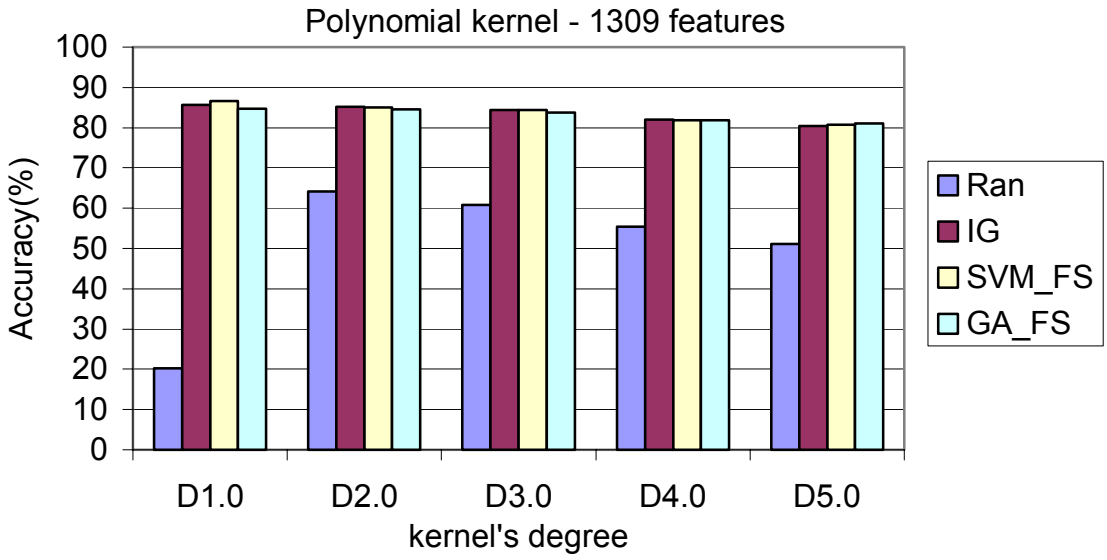

Figure 4.10 – Influence of features number and kernel degree on accuracy of classification

When the vector dimension increases even more (Figure 4.11 and Figure 4.12) 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, these results exceed with only 0.26% those obtained by SVM_FS for 1309 features. But taking in consideration the time needed for training this takes 32 more minutes (as it can be seen in Figure 4.8).

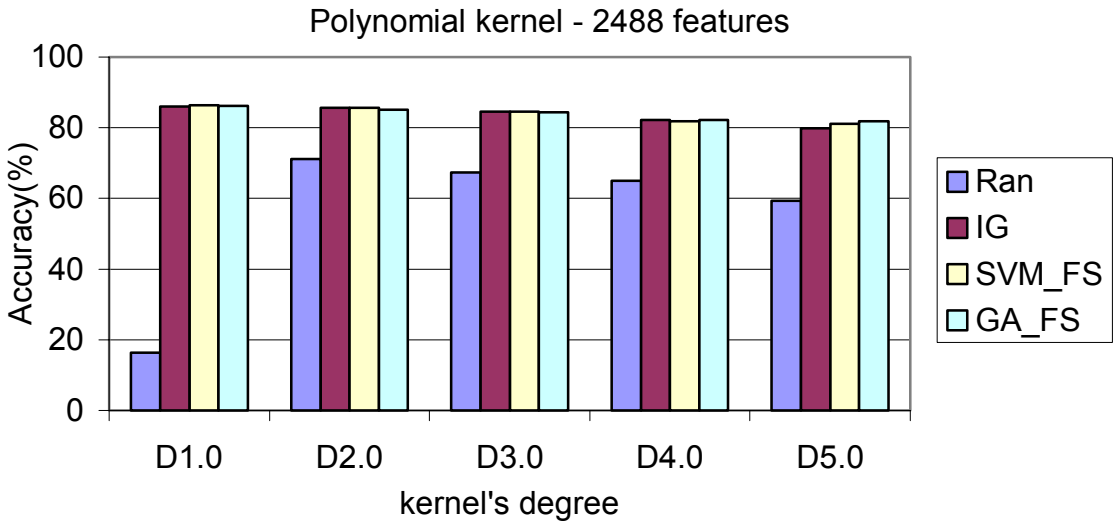

Figure 4.11 – Comparison between feature selection methods with 2488 features

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.

As a conclusion for the polynomial kernel the GA_FS obtains better results comparatively with SVM_FS only when we work with a relatively great 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.

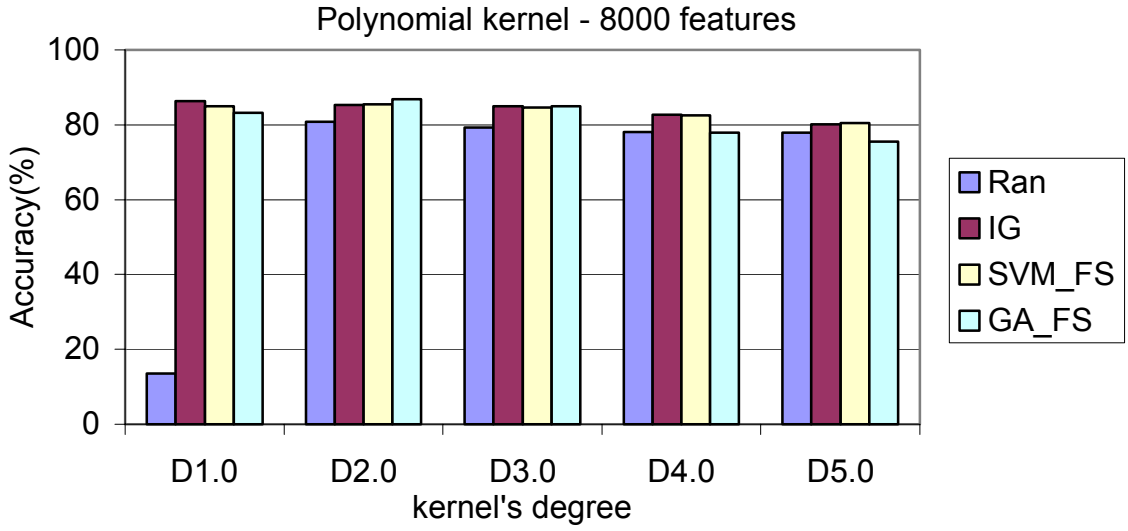

Figure 4.12 – Comparison between features selection methods with 8000 features

In Table 4.1 I presented all averages accuracies obtained and it can observe that the SVM_FS method obtains better results for each dimension of the data set. Also it can observe that the average accuracy doesn't increase when the dimension of the set increases (especially for SVM_FS). The GA_FS obtains the best results only for 8000 selected features. The SVM_FS method obtains best results with a small dimension of the features space (85.03% for 475 features) in comparison with IG that needs more features (8000 features for 84.72%) for obtaining its best results. GA_FS needs also 8000 features for obtaining its best results 85.01%.

Method / Nr. features	Random	IG	SVM_FS	GA_FS
475	44.56	82.91	**85.03**	81.94
1309	51.02	84.62	**84.90**	83.91
2488	60.57	84.54	**85.02**	84.90
8000	78.63	84.72	82.42	**85.01**

Table 4.1 – Average accuracy over all data sets tested from polynomial kernel and nominal data representation

Even if I haven't presented here results obtained using Cornell Smart data representation the greatest value that was obtained for classification accuracy (87.11%) was for this type of data representation having 1309 features and kernel degree equal with 2.

4.6.3 Parameter *C* for Gaussian Kernel's Influence

In the next charts I am presenting a comparison of results obtained for Gaussian kernel and different values of parameter *C* and Binary data representation.

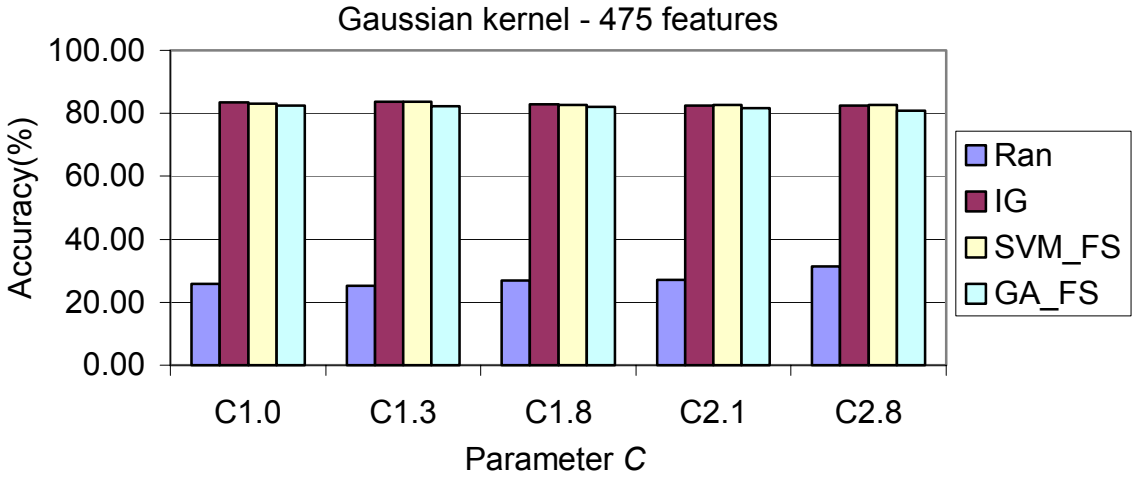

Figure 4.13 – Comparison between feature selection methods for 475 features

For a relatively small dimension of the feature space (Figure 4.13) the SVM_FS obtains better results in all tested cases than GA_FS. The IG also obtains better results in comparison with GA_FS but worst than SVM_FS. Thus the maximum value 83.63% was obtained for SVM_FS with *C*=1.3 in comparison with the maximum value obtained by GA_FS which was only 82.39%, obtained for *C*= 1.0. The maximum value obtained for IG is 83.62 for C=1.3 being very close to SVM_FS results. For this set dimension the best value was obtained with SVM_FS method and C=1.3 but Cornell Smart data representation.

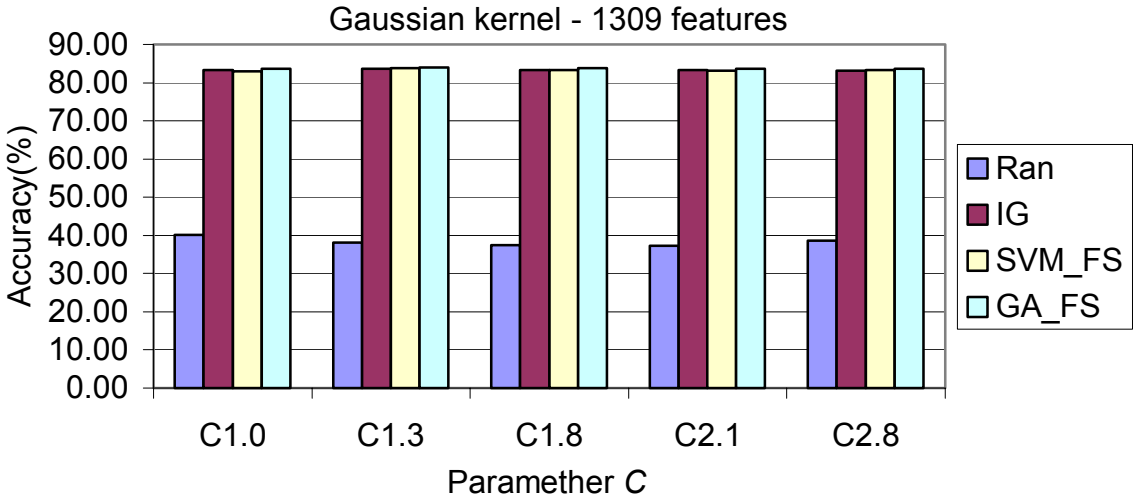

Figure 4.14 – Comparison between feature selection methods for 1309 features

When the number of features increases to 1309 (Figure 4.14) the GA_FS method obtains results with 0.22% better then SVM_FS. Thus GA_FS obtains an accuracy of 83.96% for *C*=1.3 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 all times the GA_FS obtains better results. For this set the IG method obtain only 83.62%.

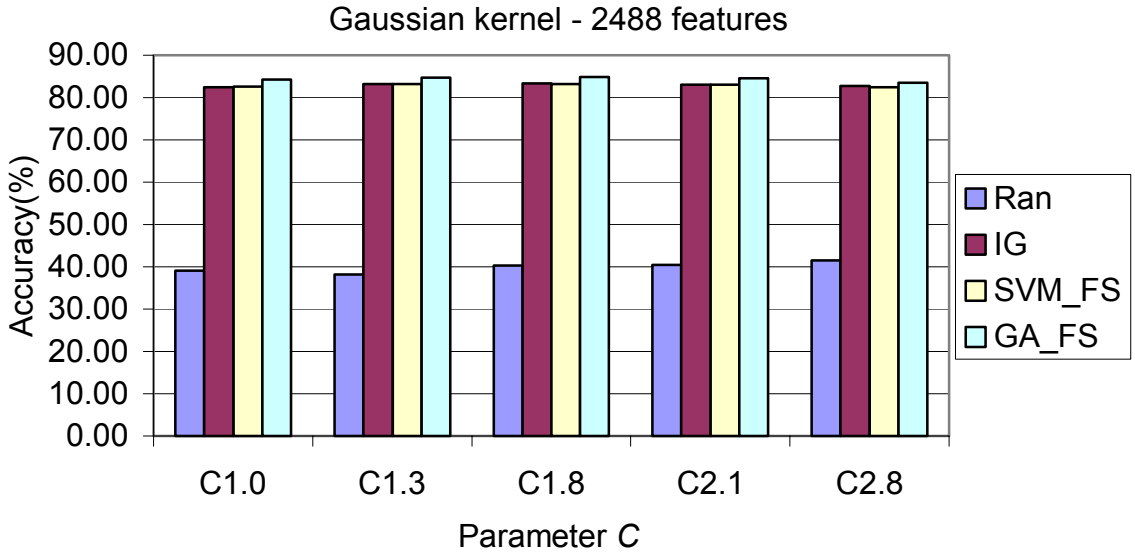

Figure 4.15 – Comparison between feature selection methods for 2488 features

In Figure 4.15, 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 increases to 84.85% for GA_FS and *C*=1.8 Binary data representation. With SVM_FS the maximum value obtained is only 83.24% considering *C*=1.3 and for IG is 83.36% considering *C*=1.8. The discrepancy has been more than 1% in accuracy between SVM_FS and GA_FS. For working with Gaussian kernel usually GA_FS return better results.

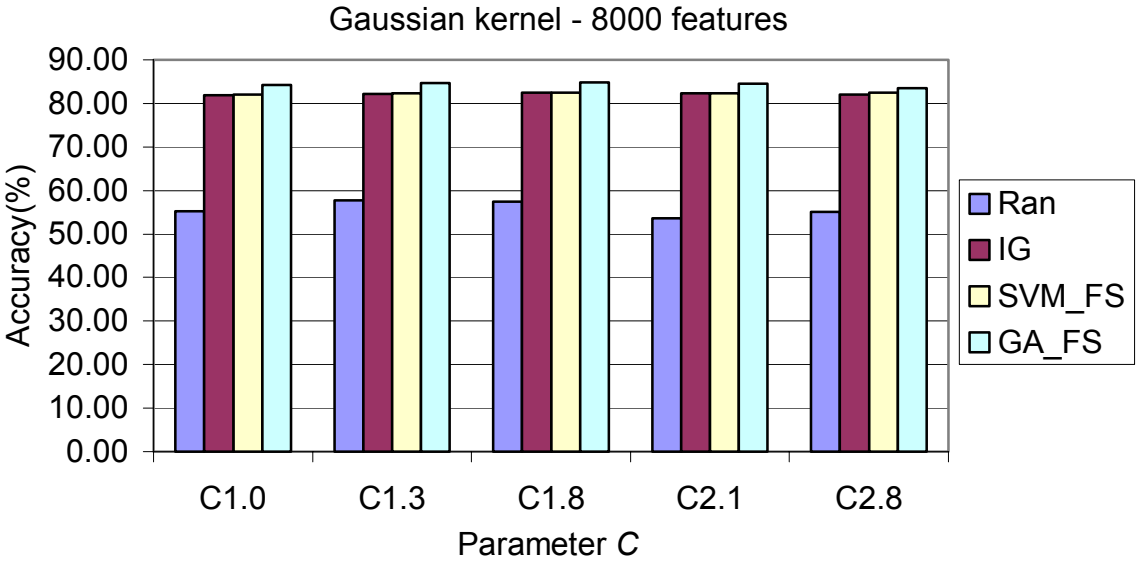

Figure 4.16 – Comparison between GA_FS and SVM_FS for 8000 features

When the number of features increases more the maximum accuracy obtained doesn't increase (see Figure 4.16). 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 classification accuracy decreases to 82.47%. Also for IG method the classification accuracy decrease to 82.51% for *C*=1.8. In general the GA_FS obtains better results for all the tested values.

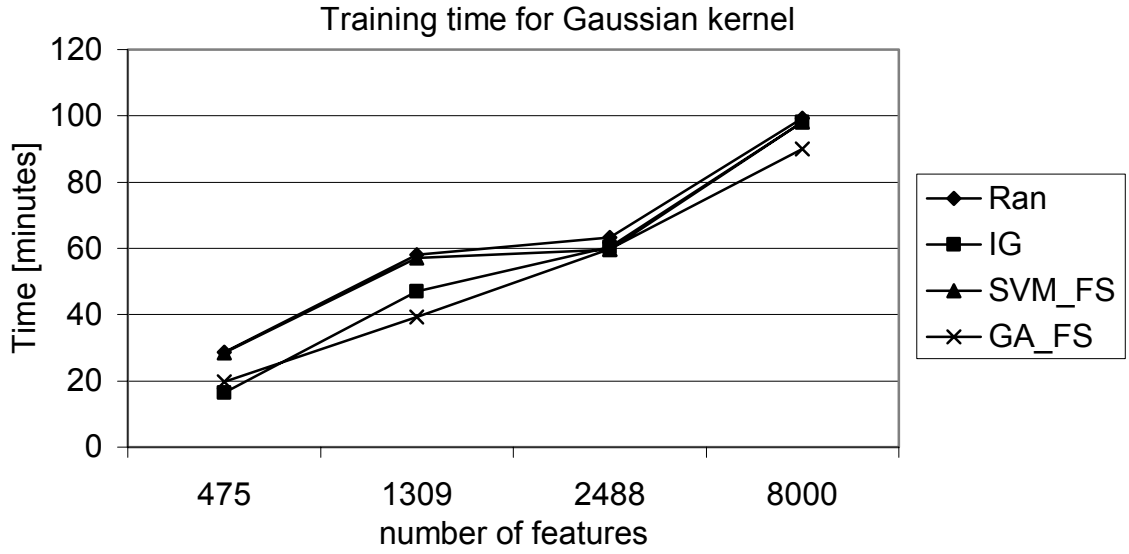

Figure 4.17 – Training time for Gaussian kernel with *C* = 1.3 and binary data representation

The training times for the Gaussian kernel are given for parameter *C*=1.3 and Binary data representation in Figure 4.17. 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 method obtains for a small dimension the best learning time, when dimension increase, increase also the training time more than the time needed with GA_FS method. The training time for Random features selection takes in all cases more time than other tested method.

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

Method / Nr. features	Random	IG	SVM_FS	GA_FS
475	25.26	83.27	83.31	81.93
1309	38.39	83.33	**83.39**	**83.41**
2488	39.49	83.07	83.02	**84.02**
8000	56.61	82.20	82.42	**83.32**

Table 4.2 – Average achieved over all data set tested for Gaussian kernel and Cornell Smart data representation

In Table 4.2 the average over all tested values for the Gaussian kernel are computed. For Gaussian kernel GA_FS method the results are better comparatively with SVM_FS, and in both cases the

results are greater than those obtained with IG or Random selection (see Table 4.1 and Table 4.2, IG and Random columns). In comparison with the results obtained using the polynomial kernel the results obtained using Gaussian kernel are smaller, whatever of feature selection method used.

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 I obtained only 84.84% for 2488 features and C=1.8 with GA_FS. In Table 4.3 I presented an average of classification accuracy over all results obtained for each feature set dimension and kernel type.

Kernel type	Nr. of features	Ran	IG	SVM_FS	GA_FS
Polynomial	475	39.30%	76.81%	83.38%	71.20%
	1309	44.72%	80.17%	82.92%	78.46%
	2488	52.07%	81.10%	81.88%	74.49%
	8000	66.65%	77.23%	77.33%	75.85%
Gaussian	475	26.51%	82.77%	83.00%	81.61%
	1309	38.49%	83.25%	83.27%	83.23%
	2488	40.31%	83.07%	82.85%	83.75%
	8000	51.52%	83.20%	82.45%	83.75%

Table 4.3 – 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%. IG obtains comparable results with SVM_FS for Gaussian kernel but worst results for polynomial kernel. Random selection obtains all the time the worst results.

4.6.4 The Influence of Features Selection on Web Documents

In this section I am presenting the influence of the number of relevant features using a dataset created with HTML documents taken from DMOZ as I already presented in section 2.4.4.

Because I am evaluating the influence of the features number on the accuracy of this dataset classification I need to use also the other steps from classification process presented in Figure 1.1. For this dataset I choose only the SVM_FS method for the feature selection step because as I showed before, this method offers the best results comparatively with other evaluated methods. I have more than 2 classes so that I use multi-class classification method that was presented in section 3.2. I will present the results for tree types of data representation: Binary, Nominal and Cornell Smart, presented in Section 2.4.5, and two types of kernels.

In the feature selection step I am using different input vector dimensions. So, using different thresholds I am selecting 8 different vector dimensions: 500, 1000, 1500, 2000, 2500, 5000, 10000 and 25646. I also showed that for this kind of file when number of features increases the accuracy of classification decreases as can be seen in the Figure 4.19.

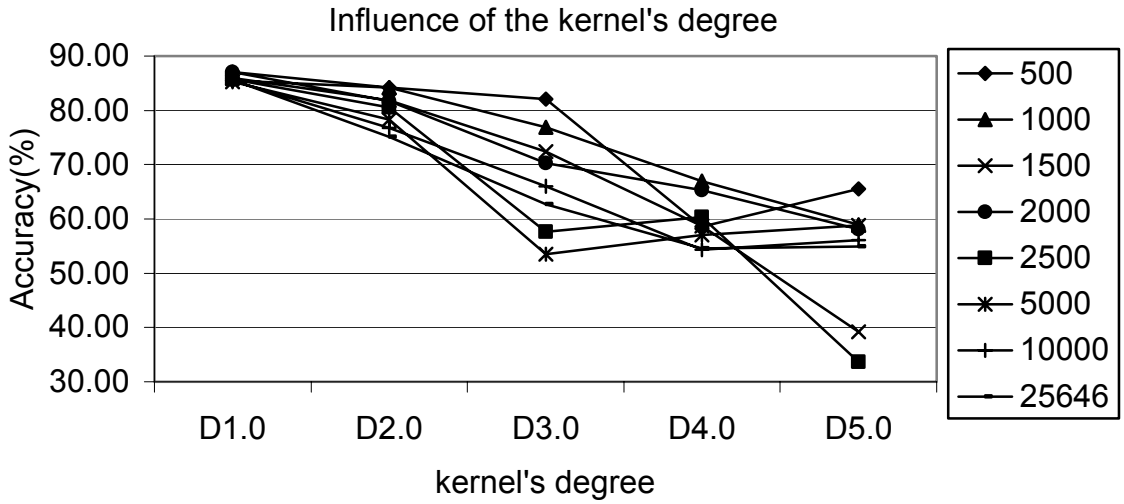

Figure 4.18 – Influence on Polynomial kernel's degree and Binary data representation

In Figure 4.18 the best results are obtained for a small dimension of the feature vector (between 500 and 1500) for all degrees of the polynomial kernel. Also an interesting observation occurs for degree 1 and binary data representation because regardless of the vector dimension I obtained the best results. This shows that the HTML files are relatively linear separable, conclusion that was also obtained for Reuters text file.

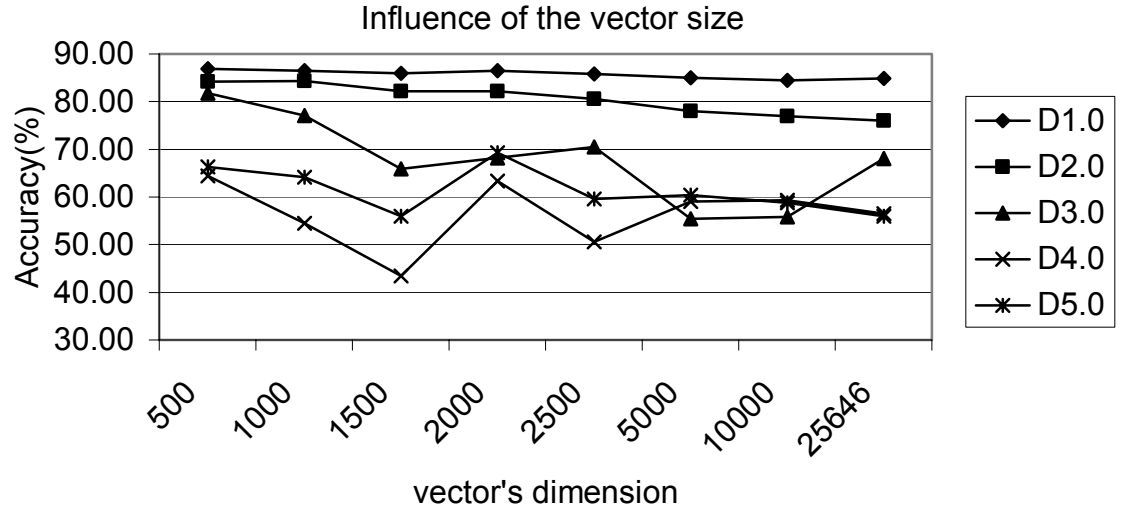

Figure 4.19 - Influence on vector size for Polynomial Kernel and Cornell Smart data representation

For Cornell Smart data representation and polynomial kernel (Figure 4.19) the same tendency was observed. Thus from another point of view can be observed that when the number of feature increase the classification accuracy have a small decrease. For degree 1 the classification accuracy has a small decrease from 86.89% for 500 features to 84.86% for 25646 features. Also the classification accuracy decreases when the degree on kernel decreases. The same tendency is obtained for Gaussian kernel. For example in Figure 4.20 it can be observed that for a Gaussian kernel and 500 features for all values of the parameter C are obtained the best results. When the number of features increases, the accuracy of classification sometime decreases or sometime increases, but not more than that obtained for 500 features.

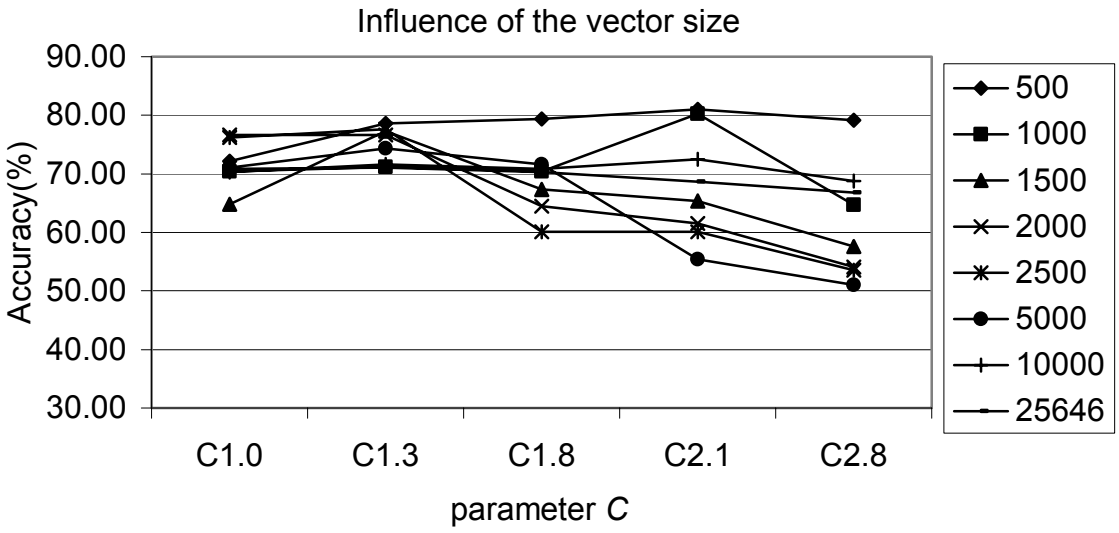

Figure 4.20 – The influence of the vector size for Gaussian kernel and Binary data representation

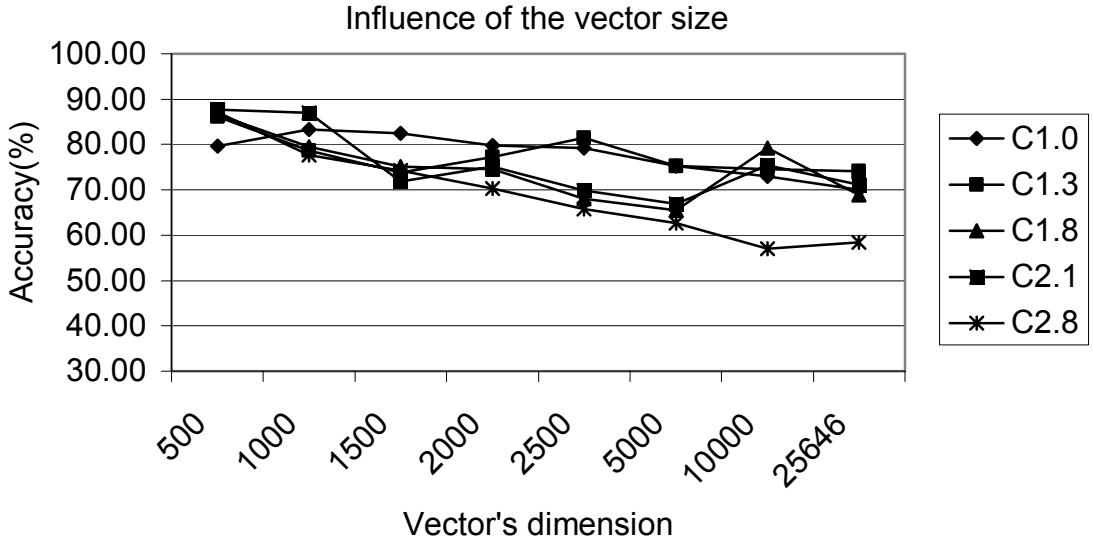

Figure 4.21 – The influence of the vector size for Gaussian kernel and Cornel Smart data representation

In Figure 4.21 are presented results obtained for Gaussian kernel and Cornell Smart data representation from a different point of view. For each value of parameter *C* are presented results obtained for each vector dimension. So, it is easier to observe the decreasing tendency of the classification accuracy when the number of features is increasing.

Analyzing all figures can be observed that all maximum values for classification accuracy are obtained for a small value of the input vector usually between 500 and 2000. For example considering a Polynomial kernel and Cornell Smart data representation, for kernel degrees equal with 1, 3, 4, and 5 the maximum accuracy is obtained for 500 features. Only for degree 2 the maximum accuracy (84.41%) is obtained for 1000 features. Comparing with Reuters text documents, for Web documents the best result (87.70%) was obtained for Gaussian Kernel with parameter *C*=2.1 and Cornell Smart data representation. Maximum value for polynomial kernel was only 87.01% obtained for degree of kernel equal with 2 and Cornell Smart data representation.

"In theory, there is no difference between theory and practice. But in practice, there is."

Unknown Author

5 Results Regarding Classification / Clustering Problems using SVM Technique

5.1 Classification Obtained Results for Polynomial Kernel

In order to improve the classification accuracy using polynomial kernel my idea was to correlate the kernel's bias with the kernel's degree and it was presented in section 3.7.1. As a consequence I 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 kernel's degree I vary the value of the bias from 0 to the total number of features (presenting here only the most representative results obtained for 16 distinct values).

Here I am presenting results obtained using a set with 1309 dimensions with SVM_FS method because, as I showed in the previous chapter, the best results were obtained with it. So that, in presented cases, I vary the bias from 0 to 1309. Usually in the literature, when the bias is not correlated with the degree of the kernel, it is selected between 0 and the total number of features.

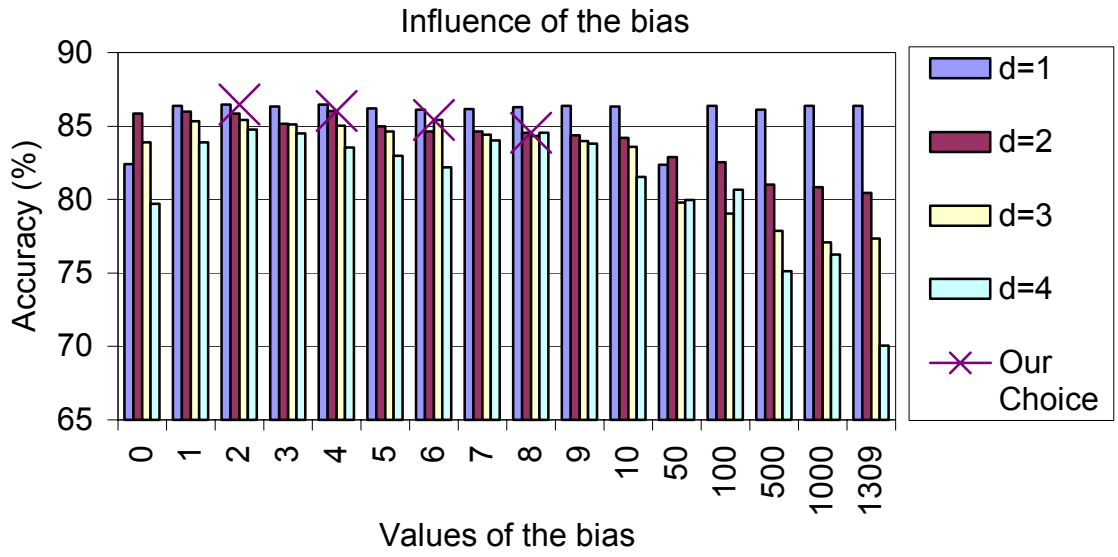

Figure 5.1 – Influence of bias for Nominal representation of the data

In Figure 5.1 results obtained with polynomial kernel and Nominal data representation by varying the degree of the kernel and the bias are presented. In "Our choice" entry I put only the values that were obtained using my formula that correlates the polynomial kernel's parameters. As it can be observed, using my correlation (equation 3.51) 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 my formula I obtained a value with 0.21% smaller than the best obtained results (84.56% in comparison with the best obtained 84.77%).

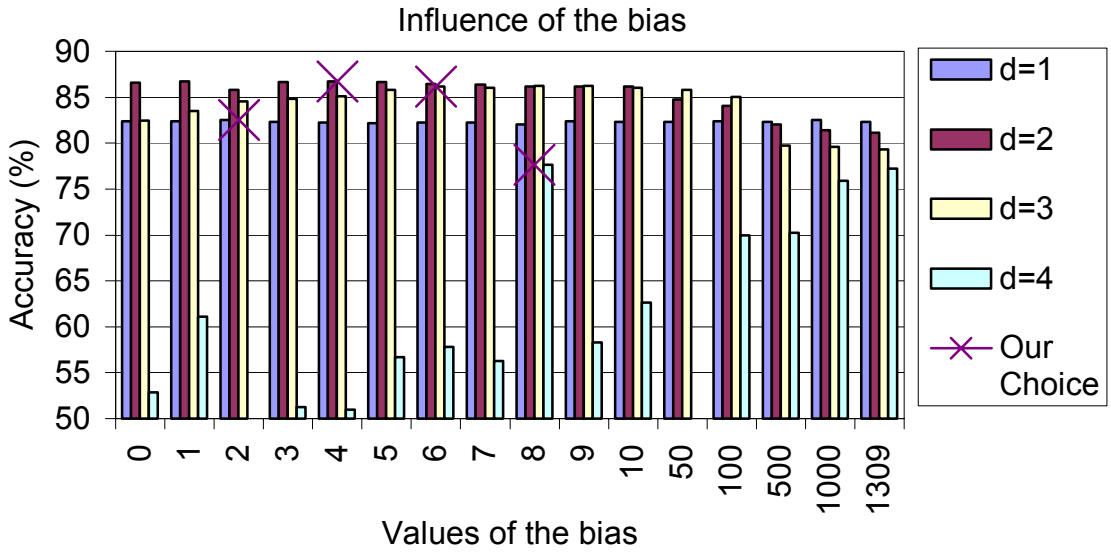

Figure 5.2 – Bias influence for Binary representation of the input data

Bias	D=1	D=2	D=3	Our Choice
0	80.84	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**	**87.11**	65.12	**87.11**
5	82.01	86.47	85.54	
6	81.71	86.60	**86.51**	**86.51**
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 5.1 Bias influence for CORNELL SMART data representation

Effective values of the accuracy obtained using Cornell Smart data representation, for each kernel degree and for each bias, are presented in Table 5.1, with bold are represented the best obtained values. For each degree there are multiple bias values involving best results and my proposed formula assures to hit these values in almost all cases. Also an interesting remark is that for kernel degree equal to 1, I 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 Table 5.1, with no bias I obtain the worst results for each degree of kernel. In this table are presented all obtained results for each bias and three different values of kernel degree. With Cornell Smart data representation and degree of Polynomial kernel equal with 2 I obtain the best value 87.11% and my formula assure to hit this. The same tests were developed also for Binary data representation and also have more values that obtain best results but I hit these values in almost all cases (Figure 5.2). Only for degree 3 the best value was obtained for bias equal with 8 with 0.085% greater than my choice.

5.2 Classification Obtained Results for Gaussian Kernel

For the Gaussian kernel I modified the usually parameter *C* that represents the number of features from the input vector, with a product between a small numbers (noted also *C* in my formula 3.52) and a number *n* that is computed automatically. I present here tests with four distinct values of *C*. For each of these values, I vary *n* from 1 to 1309 (total number of the features used). Because my proposed value for *n* is automatically computed, this number can not be specified in the command line, so that for each value of this parameter I specified a value called "auto" (in Figure 5.3) that means the value automatically computed using my formula.

I developed tests only for Binary and Cornell Smart representations of the input data. Into Gaussian kernel I fill in a parameter that represents the number of elements greater then zero (parameter "*n*" from equation 3.52). Nominal representation 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 I don't present here the results obtained using Nominal representation and Gaussian kernel.

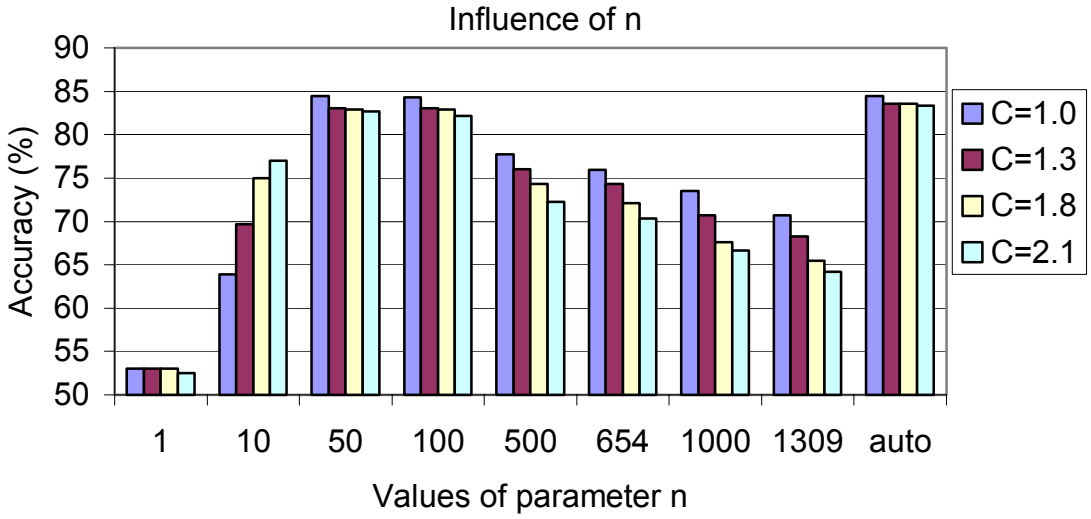

Figure 5.3 – Influence of the parameter "*n*" for Gaussian kernel and Binary data representation

In Figure 5.3 I present results obtained for Binary data representation. When I use my correlation, I 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 decreases, 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 decreases. The same tests were also developed using Cornell Smart data representation, obtaining the same tendency and average accuracy with 1% better than in Binary case (Figure 5.4).

I 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 I obtained best results only with my new proposed formula.

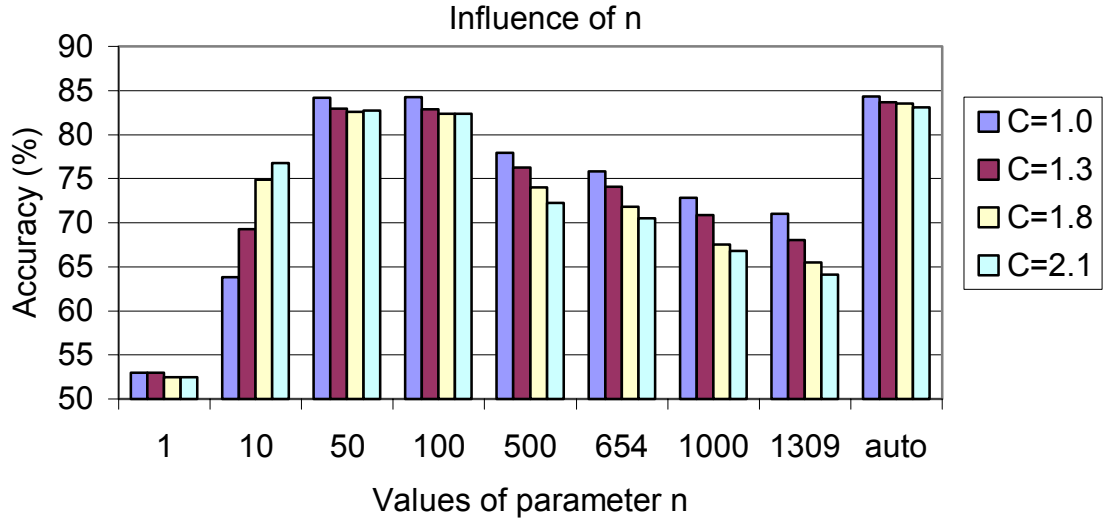

Figure 5.4 – Influence of the parameter *n* for Gaussian kernel and Cornell Smart data representation

5.3 A "de Facto" Standard: LibSVM

Almost all researchers present their results using an implementation of support vector machine technique called "LibSvm" developed by Chih-Chung Chang and Chih-Jen Lin two researchers from National Taiwan University, and available at [LibSvm]. LibSvm is simple, easy-to-use, and efficient software for SVM classification and regression. In what follows I try to present results of classification using my application versus the results obtained for the same set using LibSvm. LibSvm is a command line application and allows the user to select different algorithms for SVM, different types of kernels and different parameters for each kernel. LibSvm implemented five types of SVM (C-SVM, nu-SVM, one-class SVM, epsilon-SVR and nu-SVR) and has 4 types of kernels (linear, polynomial, Gaussian and sigmoid). The forms of the kernels that I used are:

- Polynomial $k(x,x') = \left(gamma \cdot \langle x \cdot x' \rangle + coef0\right)^d$, where *gamma*, *d* and *coef0* are variables.
- Radial basis function (RBF) $k(x,x') = \exp\left(-gamma * \|x - x'\|^2\right)$, where *gamma* is the single available variable.

The default value of *gamma* is 1/k, k being the total number of attributes, the default value of *d* is 3, and the default value of *coef0* is 0.

I used for comparisons only "C-SVM" and "one-class SVM" with different parameters. I compare with C-SVM for supervised classification and with "one-class SVM" for unsupervised case (clustering).

5.4 A Graphical Interpretation of the Decision Function

My developed application offers the possibility to see a graphical visualization of the results of the algorithm into a bidimensional space. The application offers functions like: saving and loading data training files containing points belonging to one or more classes and have the user interface closer to LibSvm. My developed application can work up to 7 classes (topics) but I presented results only for three classes because the LibSvm application can only work up to 3 classes. I am presenting results for different type of kernels and different degrees obtained using my application in comparison with results obtained with LibSvm using the same characteristics. I have called my application "UseSvm_toy". This representation was made in order to have a view of what it happens in the classification process and to have a visual interpretation of the obtained results.

5.4.1 SVM Classification

I have tested the graphical interface for two types of kernels: polynomial and Gaussian. I am presenting the results for the polynomial kernel. In order to have an equivalent comparison I have chosen *gamma*=1 and *coef0* = 2*d for LibSvm. In the left are results obtained with LibSvm and in the right are results obtained with my application. For comparison I take in consideration the number of errors obtained in testing part and the number of support vectors resulted after the training part. The numbers of support vectors are considered because it has a great influence on the testing time.

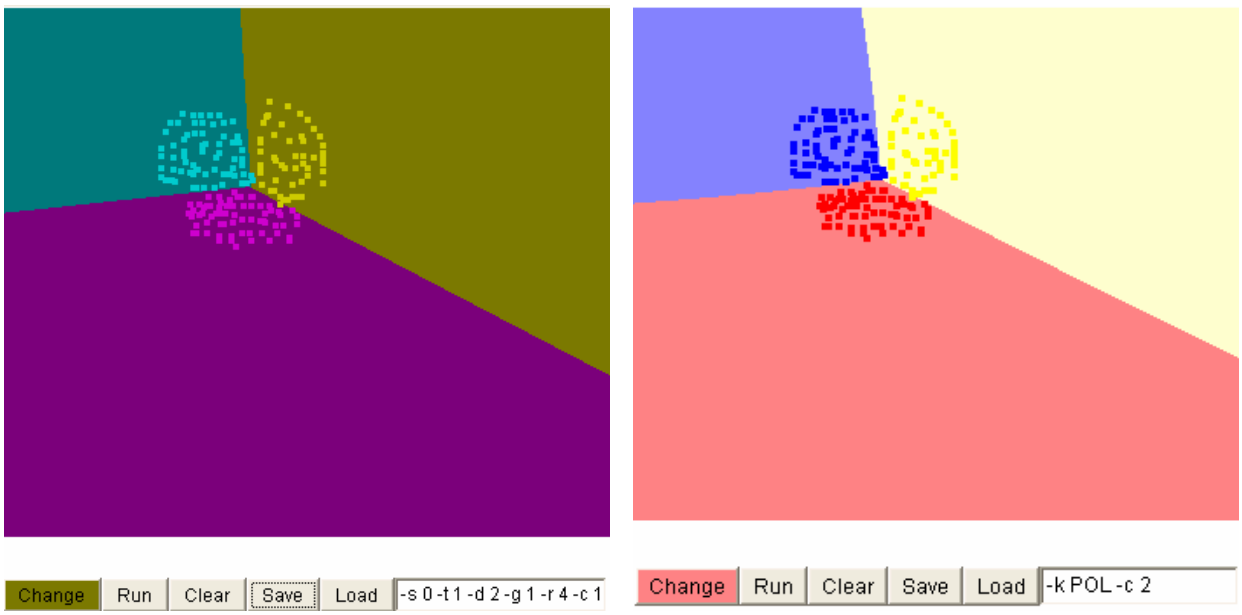

Figure 5.5- Polynomial kernel with degree 2. Left - LibSvm; Right - UseSVM_toy

For the degree 2 LibSvm has 3 errors in the testing part and UseSVM_toy has also 3 errors. The number of support vectors (sv) for LibSVM (27) is smaller the number of sv for UseSVM_toy (40). Thus, for a small number of the degree my application obtains same results using a larger number of support vectors.

As follows I present the results for the degree equal to 4. For this type of kernel both LibSVM and UseSVM_toy are working better. Both produce only 1 error, LibSVM using a number of 18 support vectors and UseSvm_Toy using a number of 15 support vectors. Better results are also obtained when the degree of polynomial kernel is increased by more than 4. In both applications obtained results are also influenced by the proposed kernel's correlation.

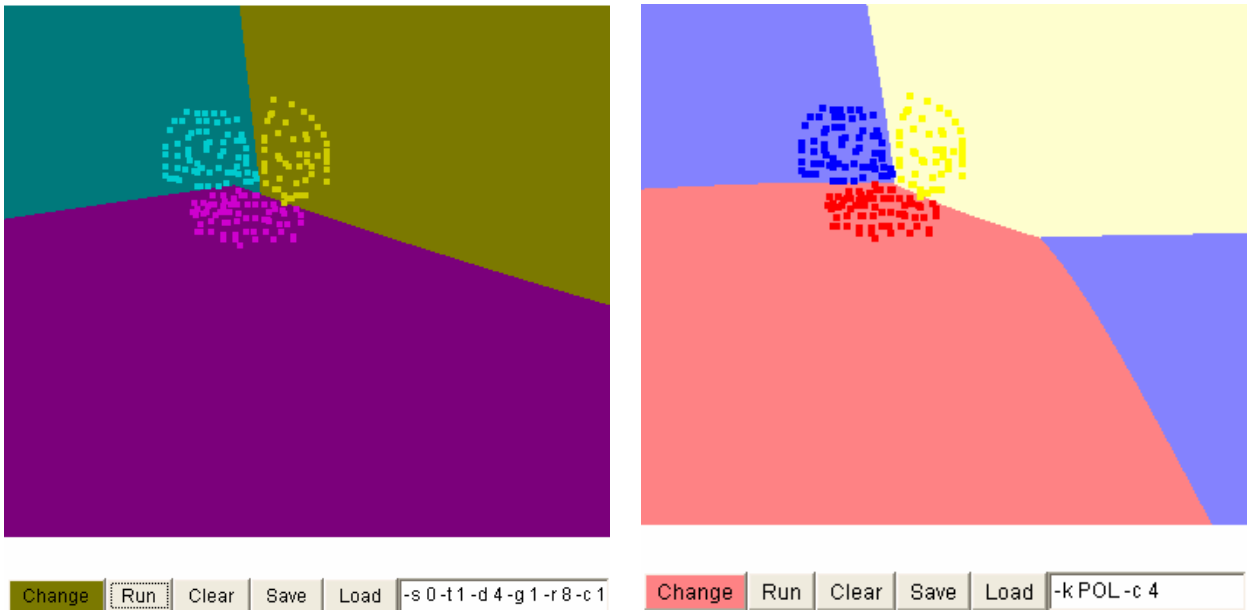

Figure 5.6 - Polynomial kernel with degree 4.

Up to this moment I presented a results obtained when the data are not so overlapped. There are three classes but these classes can be separated using a linear kernel to transpose those linearly data in a higher dimensional feature space. As it can be seen polynomial kernel usually finds right hyperplanes of separation or hyperplanes with small inflections. In the case of polynomial kernel if I have mixed data (the classes are close to each other and usually can not be separated by a right hyperplane) or I have a clump of data bounded by other data, the results are very poor for small degrees of the kernel. The results obtained for degree 2 are presented in Figure 5.7. I need to use a degree greater then five to obtain good results with the LibSvm. For UseSVM_toy the result are good for degree greater or equal with 4.

Using the default parameters of LibSvm leads to poor results. If I use the degree 2 it can still find 3 classes but for higher degrees it groups all the data into a single class. I have noticed that even though many researchers use no bias, the learning process is improved if I used a value of the bias equal to twice the value of the degree (my proposed correlation). It is logical that the bias and the kernel degree are interconnected as changing one leads to a shift in the points. In my opinion those parameters need to be connected because when changing the representing space the bias needs to be correlated in the new space.

The main advantage of the polynomial kernel is that a smaller number of support vectors are obtained. As a consequence we obtained a reduced time for testing. In order that mixed data are classified correctly I need a greater kernel degree which involves a longer time for training. In

almost all cases we do not know how does the data look like, thus we need to use a greater value of the degree to obtain better results an after that we need to try to find the optimal degree.

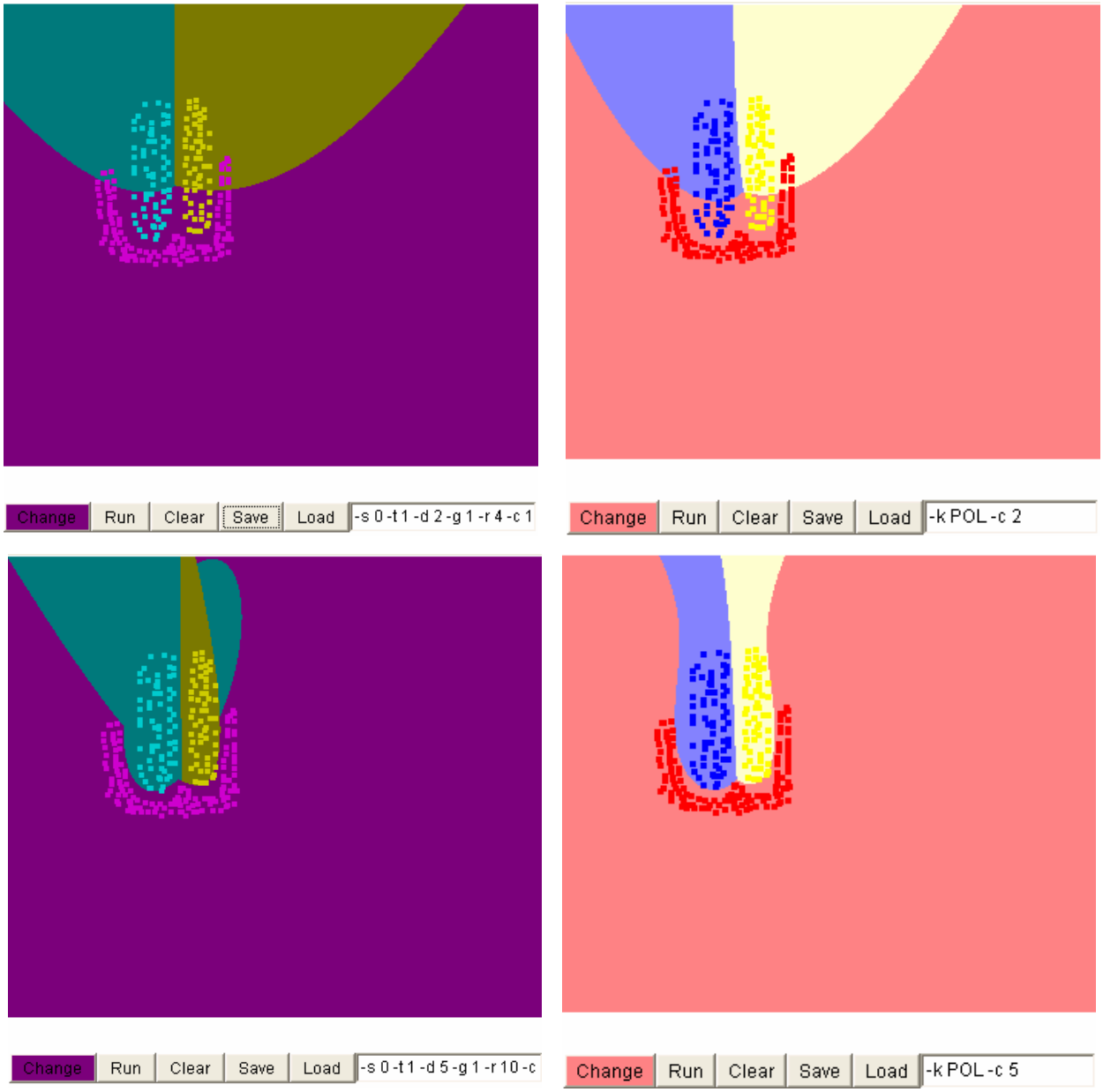

Figure 5.7 - Polynomial kernel used for mixed data

In Figure 5.8 I presented the visual results obtained for the Gaussian kernel when the variable *C* is changing. The Gaussian kernel is known to work better than the polynomial kernel when working with overlapped data and the results are poor when working with separated data. Even if good results are obtained for separated data, the resulted number of support vectors is significantly larger comparatively with the number obtained using polynomial kernel. This is why I presented only the results for overlapped data. Generally, one of the disadvantages of the Gaussian kernel is that the number of support vectors is greater than in the case of the polynomial kernel. In order to establish an equivalence between the two applications we need to use for LibSvm the parameter $gamma = \frac{1}{nC}$, where n needs to be equal to 2.

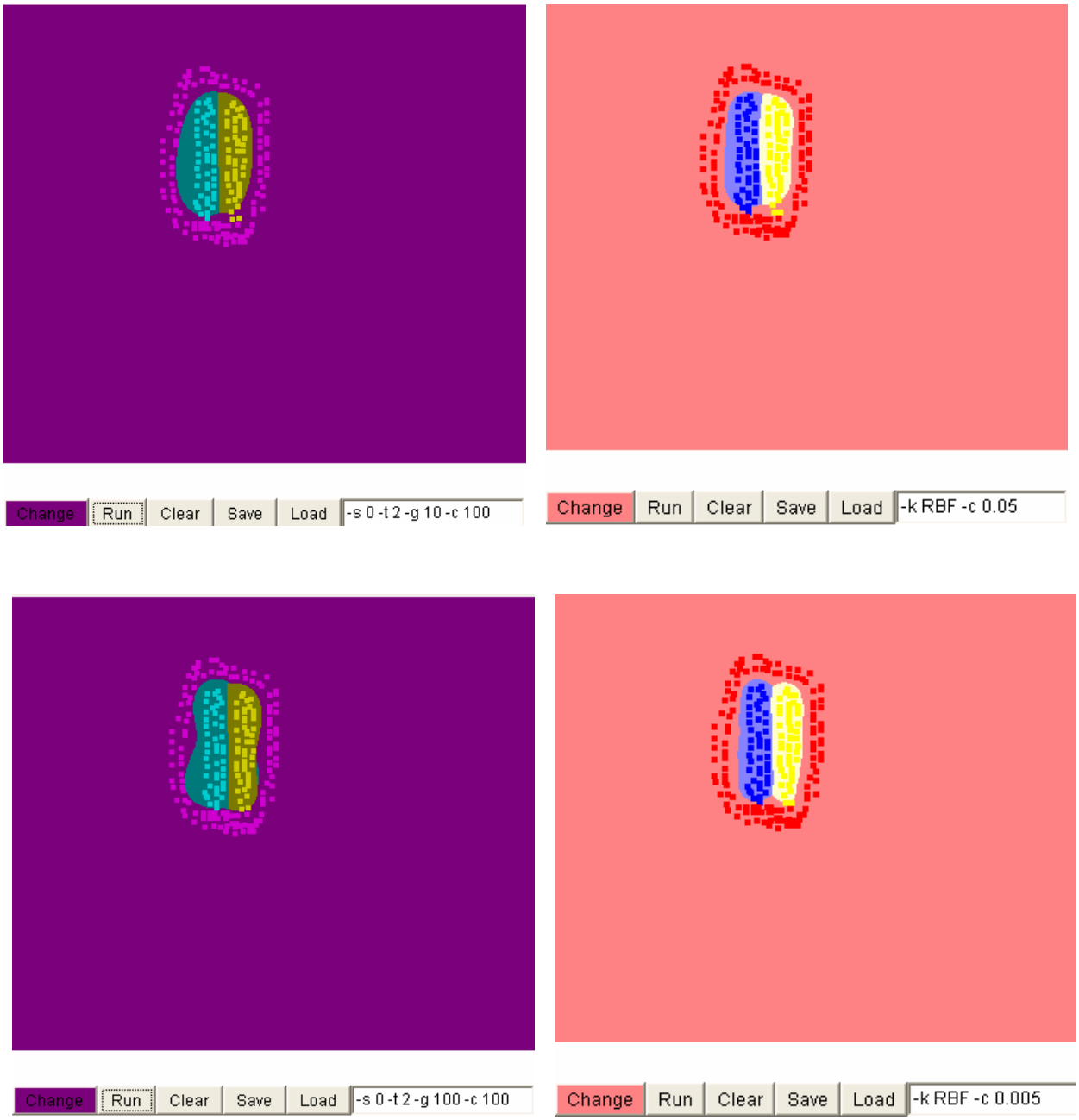

Figure 5.8 - RBF kernel with degree 0.05 and 0.005

5.4.2 One - Class SVM

In this section I am presenting some visual results obtained using clustering based on Support Vector Machine technique. The mathematical part of this method was presented in section 3.3. For clustering I present results only for the Gaussian kernel because it obtains the best results in comparison with other kernels and it is constantly used in the literature in the clustering problems. The parameters that need to be modified for this application are C – the parameter for Gaussian kernel and υ the percentage of data that are initially chosen to be support vectors (heaving Lagrange multiplier greater than 0). In the beginning I present results using a Java applet that offers the possibility of a graphical visualization of the results. The results obtained using Reuters testing files will be presented in section 5.7.

For a better view of the parameter influence on the number of classes I present the results obtained with some different values for the parameter C for different values of υ. The presented results are obtained using only my implemented application. There are not presented here a comparison between my application and LibSvm.

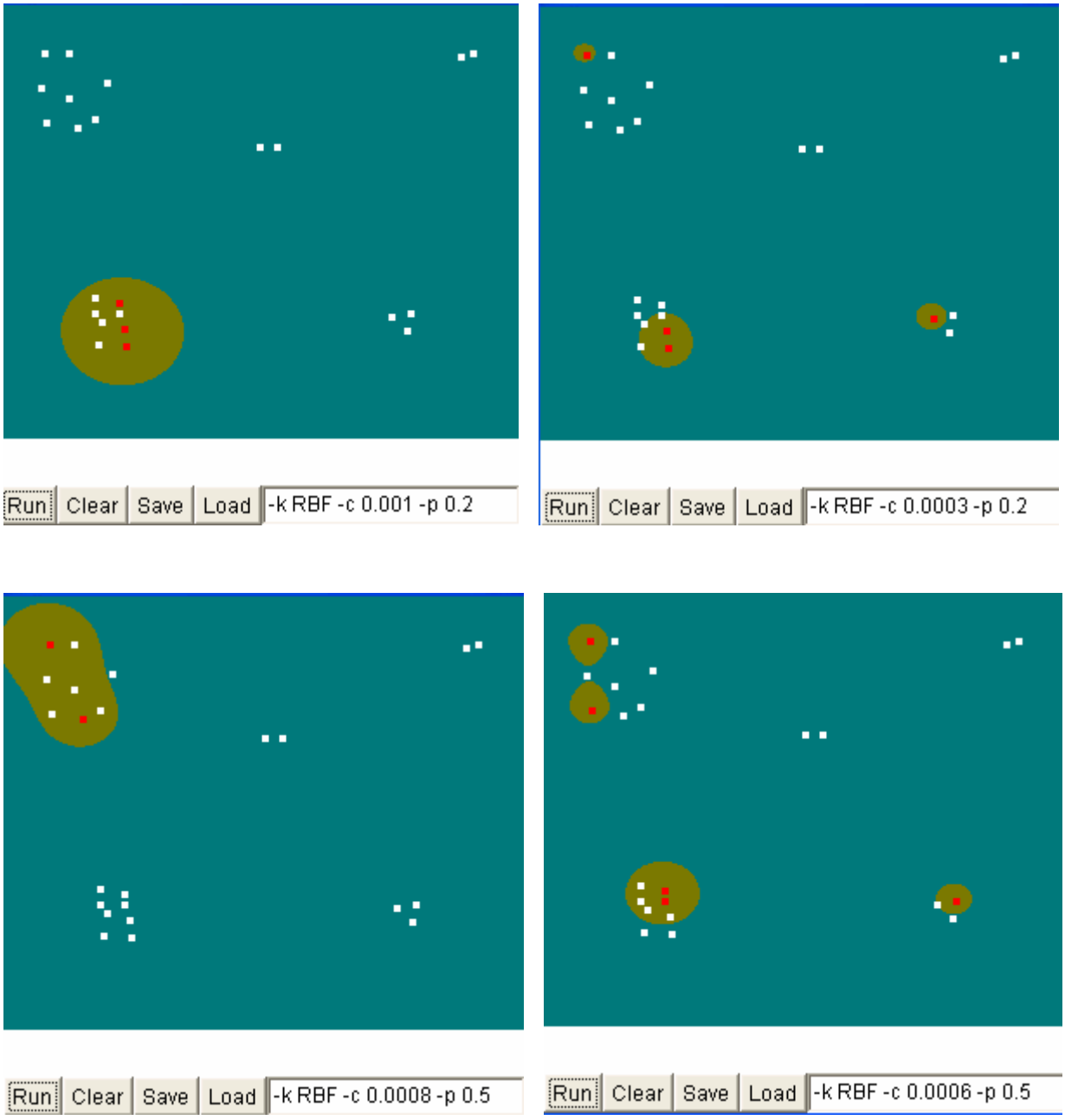

Figure 5.9 – Influence of constant *C* on the number of clusters

By modifying the parameter C we actually modify the accuracy and the number of classes, a small C leads to smaller clusters and more disconnected clusters. As C decreases, the dimension of the features space where the data is mapped increases. As C decreases the border of the clusters is more accurate (it fits the data better).

As follows I present the influence of the initially chosen data (I will consider percentages of data) on the results obtained in clustering.

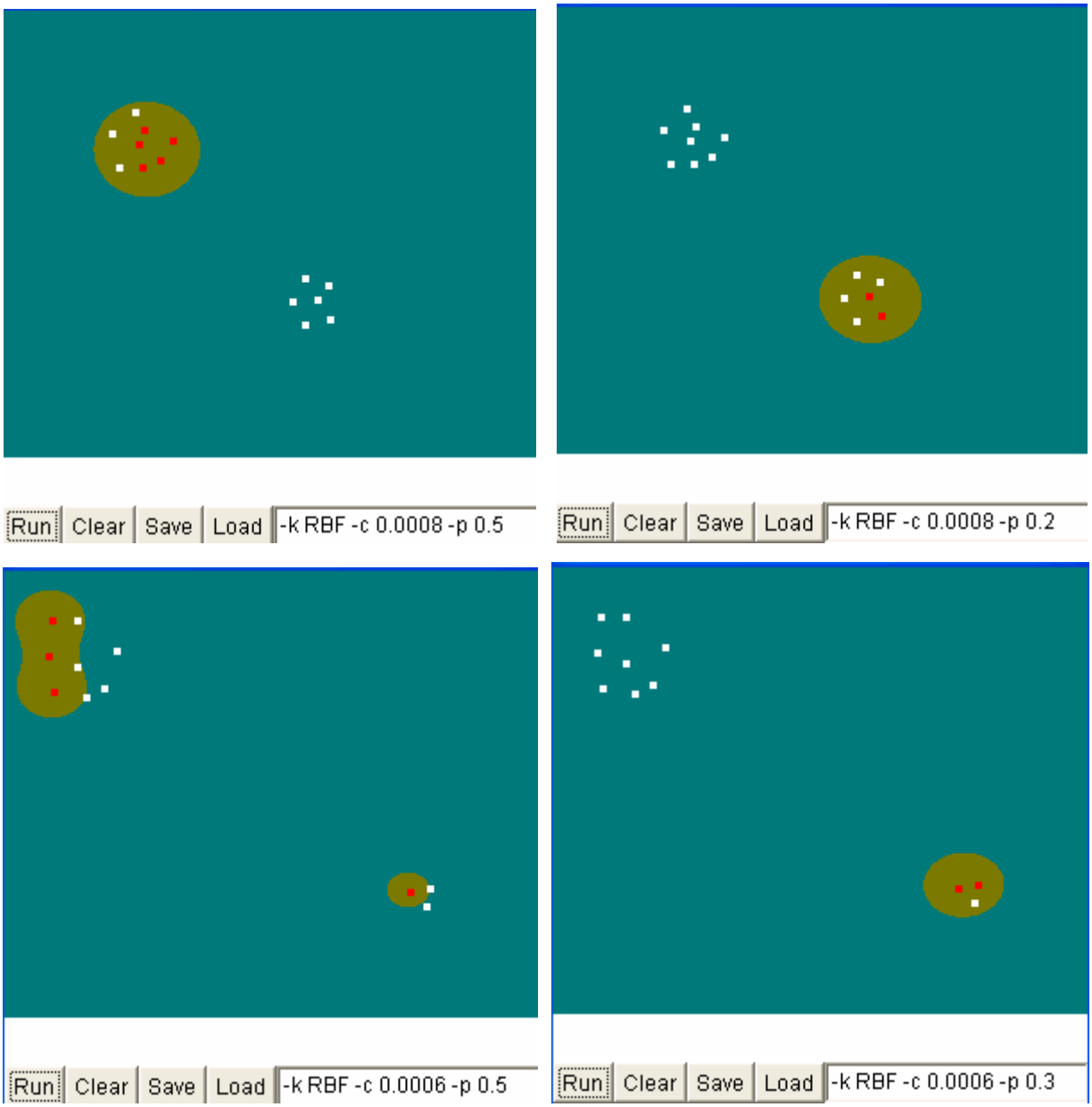

Figure 5.10 – Influence of initial percentage of data υ on the accuracy of learning

We can see that the percentage of initially chosen data has a substantial influence on the evolution of the algorithm for the same value of parameter *C*. When we have a large number of initial data (great value of υ) we have one great class or more and some smaller classes. When we have a small value of υ we have a smaller dimensional class and a small number of grouped data. I have also noticed that for value of υ greater than 0.5 the algorithm grouped all data in the same cluster.

5.5 LibSvm versus UseSvm

In this section I am presenting the influence of correlating kernel's parameters on classification accuracy. In order to do this I make a short comparison between the results that I obtained with the largely used implementation of SVM, called LibSvm [LibSvm], and my implemented application called UseSvm [Mor05_2]. LibSvm uses "one versus the one" method for multi-class classification. My developed UseSvm program uses "one versus the rest" method, as I already

mentioned. Reuter's database, used in my tests, contains strongly overlapped data and in this case the first method usually obtains poor results. In the Reuters database almost all documents contain more than one topic. There are larger topics that contain more documents, representing a general category, and specific topics that contain only some documents as a subcategory of the general category. In the "one versus the rest" multiclass classification all documents which are in that certain selected topic are chose comparatively with others documents from the set. Therefore, in each case we have documents also in positive and negative classes in the training set. In "one versus the one" multiclass classification, if we have a general topic versus a specific topic, the same documents are belonging to the general and the specific class too, making the learning process very difficult because these classes may be totally overlapped.

I have used only one set for these tests, set that obtains the best results in previous section (having 1309 features, obtained using SVM_FS method). In order to fairly compare LibSvm with my UseSvm, I eliminated, when possible, Reuter's overlapped data, for working only on non-overlapped classes, formally:

$$\forall i, j = \overline{1,13}, \mathbf{C}_i \cap \mathbf{C}_j = \varnothing \text{ for each } i \neq j \tag{5.1}$$

I choose for each sample only first class that was proposed by Reuters. I eliminated also classes that are poorly or excessively represented obtained only 13 classes. The data set was randomly split in two sets and used for training and testing for both LibSvm and UseSvm. Results obtained by LibSvm are poor in comparison with the results obtained with my application, because, despite my efforts, the data are however slightly overlapped. In the next figures I present results obtained for the polynomial kernel and the Gaussian kernel. I am using equivalent parameters for both applications. As LibSvm has more parameters than UseSvm, I have left on default value the parameters that appear only in LibSvm.

As I already specified, for polynomial kernel my suggestion was to make "b=2*d" (see Section 3.7.1). I present results using LibSvm with b=0 (default value depict as LibSvm) respectively with b=2*d (specified explicitly in the command line as LibSvm+"b") comparing with my UseSvm program.

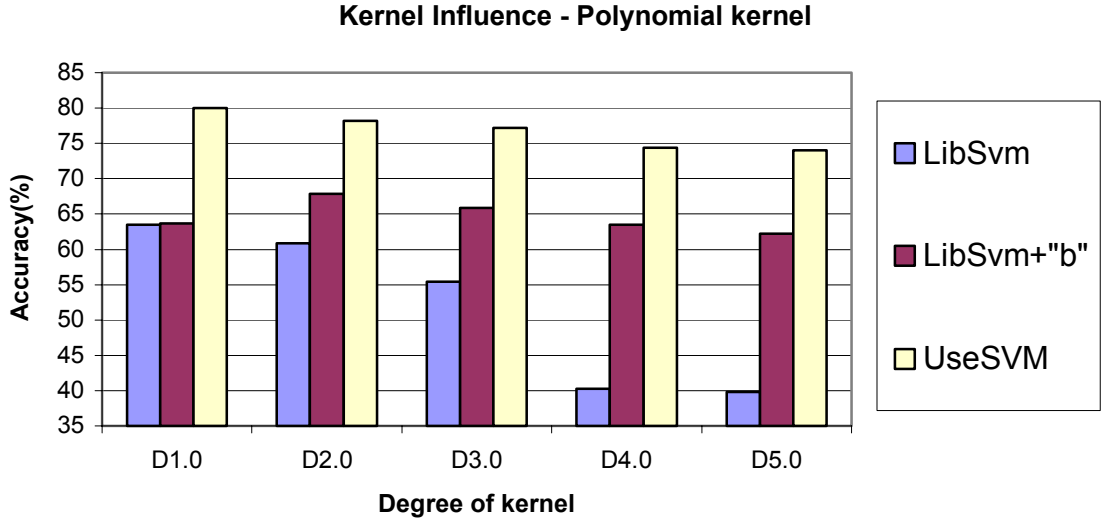

Figure 5.11 – Influence of correlation between parameters from polynomial kernel

As it can be observed from Figure 5.11, my UseSvm program obtains far better results than the well-known LibSvm (with an average gain of 18.82% better). By comparing LibSvm with the

default bias with LibSvm with modified bias (according to my formula, note in the chart by LibSvm+"b"), I noticed that the modified bias leads to better results (with an average gain of 24.26% better). The average gain is computed as average obtained by LibSvm with the default bias divided by the average obtained by LibSvm with modified bias. For degree 1 were obtained similar results because values of default bias and value computed using my formula are quite equal.

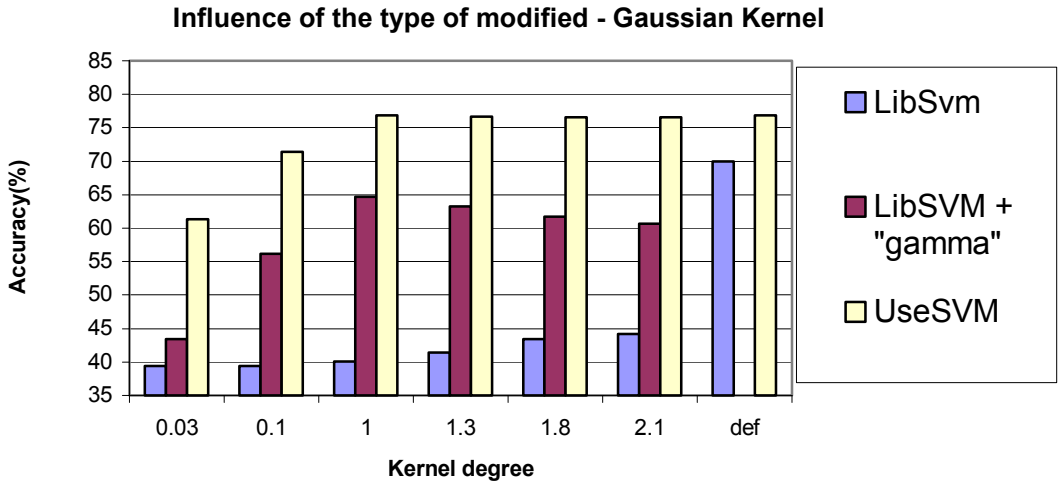

Figure 5.12 – Influence of modified about Gaussian kernel

For the Gaussian kernel simulations, presented in Figure 5.12, my suggestion was to multiply the parameter C with a parameter *n* (like I already explained in Section 3.7.2). It is difficult to give this parameter from the LibSvm's command line because *n* is computed automatically and LibSvm have only one parameter that can be modified, called *gamma*. More precisely, LibSvm uses $gamma = \frac{1}{n}$ only when gamma is default (*n* means the average of number of attributes in the input data). For LibSvm I have used gamma as $\frac{1}{C}$. For LibSvm+"gamma" I considered "gamma" to be equal to $\frac{2}{nC}$, where *n* is the number of features divide by 2. The single case of equivalence between these two programs (LibSvm and UseSvm) is obtained for the default value of gamma in LibSvm and respectively for C=1 in UseSvm. This case is presented separately as "def" in Figure 5.12.

As it can be observed, using my idea to modify the Gaussian kernel the results obtained using LibSvm are better in comparison with results obtained using LibSvm with standard kernel (with an average gain of 28.44%). My UseSvm program obtains far better results than the well-known LibSvm (with an average gain of 25.57% better). For the default parameter of LibSvm my application also has obtained better results (76.88% in comparison with 69.97% for LibSvm).

5.6 Multi-class Classification - Quantitative Aspects

For extending the SVM algorithm from two-class classification to multi-class classification typically one of two methods is used: "One versus the rest" that was presented above and "One versus the one", where a separate classifier is trained for each pair of topics. The Reuter's database contains strongly overlapping classes and assigns almost all samples in more than one class. Therefore I chose the first method for multi-class classification. Also I tested the method "one

versus the one", but the obtained results are not quite good. Also the training time doesn't decrease so much because there are more decision functions to be learned even for small datasets. Initial used set has 68 distinct topics. From these topics I have eliminated those topics that are poorly or excessively represented. Thus I eliminated those topics that contain less than 1% documents from all 7083 documents in the entire set. I also eliminated topics that contain more than 99% samples from the entire set, as being excessively represented. The elimination was necessary because with these topics we have the risk of classifying all documents into a class using only one decision function while ignoring the others. After doing so I obtained 24 different topics and I compute 24 different decision functions. The obtained set was split randomly in a training set (4702 samples) and an evaluation set (2531 samples).

For some number of features I tested multiclass classification using all 68 topics and in almost all cases I obtained an accuracy of 99.98% (we have cases in which only one sample wasn't classified correctly). This can occur if there are more elements into a class (for example "ccat" that occurs in 7074 from 7083). In this case the decision function for this topic will always return the greatest value and the other decision functions have no effect in multiclass classification. In this case it is easier to use a static classifier that predicts all the time the same class and that can have 99.87% accuracy.

In the training phase for each topic a decision function is learned. In the evaluating phase each sample is tested with each decision function and is classified in the class having the greatest absolute value. The obtained results are compared with the known Reuter's classification. Reuters usually classifies a document in more than one class. If the obtained results by us belong to the set of topics designated by Reuters we consider than we have a correct classification. I present also the results for polynomial kernel and Gaussian kernel and for three different types of data representation.

In order to find a good combination of kernel type, kernel degree and data representation I run ten tests, five tests for a polynomial kernel with a kernel degree between 1 and 5, and respectively five tests for Gaussian kernel with different values for the parameter C (1.0, 1.3, 1.8, 2.1 and 2.8). For this experiment I present results obtained using presented feature selection method for both types of kernels and for all the three types of data representation. For polynomial kernel I present the results only for Nominal data representation because it obtains de best accuracy. For Gaussian kernel I present results only for Binary data representation. In this section I choose to present results only for a dimension of feature set of 1309 features because, as I showed in section 4.6 obtains the best results. In [Mor05_2] I report additional results.

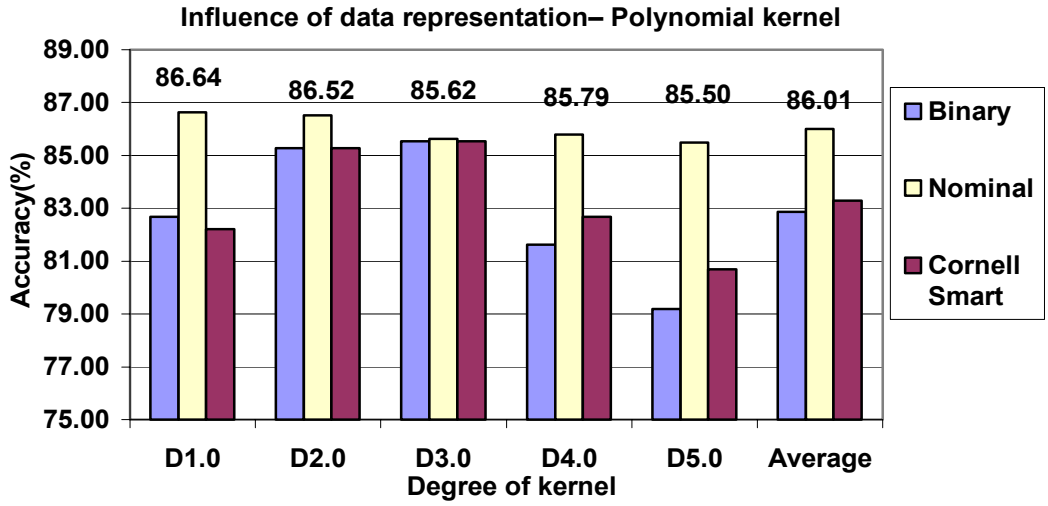

Figure 5.13 – Influence of data representation and degree of polynomial kernel

Figure 5.13 shows that text files are generally linearly separable in the input space (if the input space has the right dimensionality) and the best results were obtained for a small kernel degree (1 and 2). Taking into consideration data representation for all five tests, the best results were obtained with nominal representation, which obtained at average 86.01% accuracy in comparison with binary representation (82.86%) or Cornell Smart (83.28%).

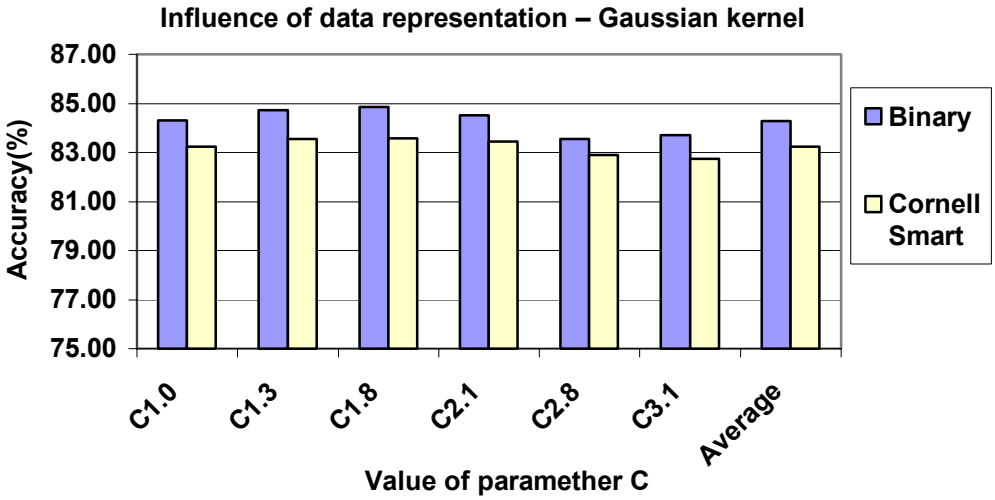

Figure 5.14 – Influence of data representation and parameter *C* for Gaussian kernel

Generally for polynomial kernel the training time is smaller than 1 hour regardless of feature selection used. For SVM_FS method, kernel degree equal with 2 and nominal data representation, the training time is 14.56 minutes for 1309 features. The results time are given for a Pentium IV processor at 3.2GH and 1Gb memory with WinXP operating system. More time results were presented in Figure 4.8.

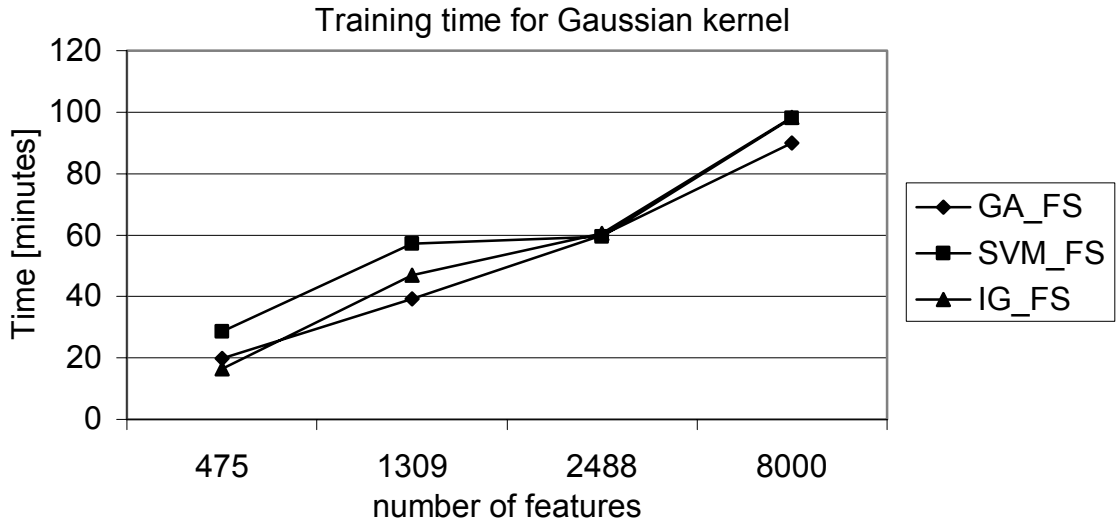

Figure 5.15 – Training time for Gaussian kernel with *C*=1.3 and binary data representation

In Figure 5.14 I present results obtained for Gaussian kernel for two types of data representation and for five distinct values of parameter C, using a data set with 1309 features obtained using GA_SVM method. I chose this method to present results here because it obtained the best results in first tests (see section 4.6.3). Into Gaussian kernel (Figure 5.14) I add a parameter that represents the number of elements greater then zero (parameter "n" from equation 3.52). Nominal

representation (formula 2.15) 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 to not belong to that document). So I don't present here the results obtained using the nominal representation.

The training times for the Gaussian kernel are given for parameter C=1.3 and Binary data representation in Figure 5.15. I presented here also the training time obtained with sets generated using IG and SVM_FS methods presented in section 4. For this type of 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 the dimension increases, it increases also the training time more than the time needed with GA_FS method.

5.7 Clustering using SVM. Quantitative Aspects

In this section I am presenting the results obtained using a clustering algorithm based on support vector machine. The implemented clustering algorithm was presented in section 3.3. As the results obtained using the Gaussian kernel are better then the results obtained using the polynomial kernel and because usually the Gaussian kernel is used in clustering I present only the results obtained with this kernel and the nominal representation of the initial data. The training and testing sets are obtained using different levels of thresholds for Information Gain. There are two parameters that have a great influence on the accuracy of clustering. These parameters are parameter C in the Gaussian kernel and parameter υ that represents the percentage of data that are initially chosen as support vectors (have α parameter greater than 0). For these parameters I initialize the Lagrange multipliers α by $\frac{1}{\upsilon m}$ where m is the number of samples. I also used two data sets, like for classification, one for training that is used to develop the hyperplane and one for testing that is used to compute the accuracy of clustering. All the results are presented in percentages of samples correctly classified (accuracy of clustering). From tests I notice that for a greater value of parameter υ (over 0.5 that means 50% of data are initially taken into consideration in the first step) the algorithm doesn't work so good (usually it classifies all documents belonging to the same class).

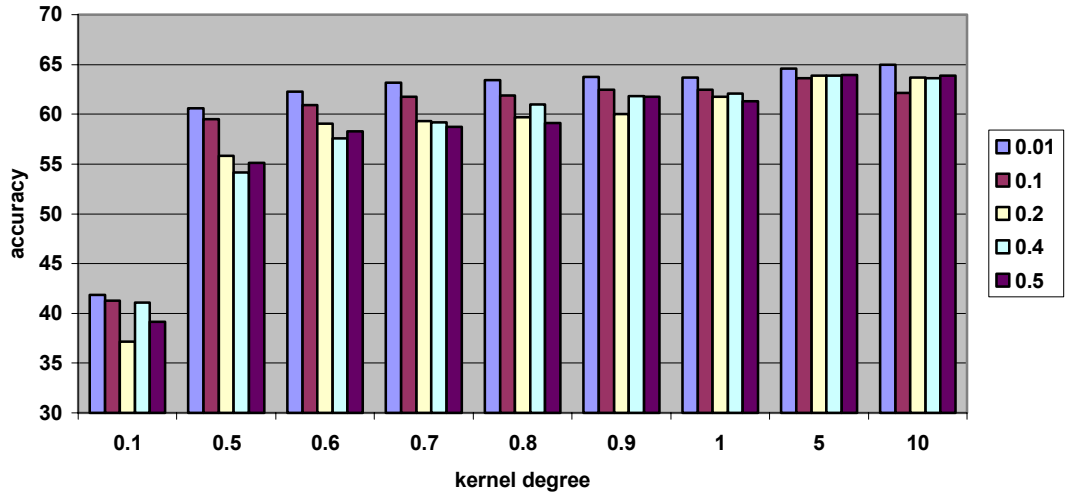

Figure 5.16 - Influence of initially chosen data (*υ*)

Thus I am presenting the results for values of υ smaller than 0.5. The results I am presenting now are not as good as the results presented before for classifying. Clustering is an unsupervised learning algorithm and after learning it is difficult to find the same classes (groups) as Reuters. However, I am comparing the groups I obtained with Reuter's classes only to have a metric for comparison, even if it is known that each clustering algorithm groups the training set from another point of view. This is why the presented results are not as important as obtained values, but are quite relevant in order to have an idea about clustering using SVM algorithm.

At first I present the influence of parameter υ and parameter *C* on the clustering accuracy for training and testing sets with a 2111 number of features using nominal representation.

As it can be observed, the classification accuracy increases when I used a greater value of parameter *C* and when I used a smaller number of initially chosen data. Best results were obtained for *C*=10 and υ =0.01 (64.99%).

I also notice that when I increase the initial number of samples taken into consideration the number of support vectors increases. This can be seen in the table 5.2.

In figure 5.17 I presented the influence of the number of selected features for the nominal representation of data. In this case I chose the sets resulting using Information Gain method as feature selection. I used sets with 41 features, 63 features, 1309 features, 2111 features and 7999 features and I analyzed the influence of the number of features on clustering accuracy. I presented the results for C=5.0.

C / υ	0,1	0,5	0,6	0,7	0,8	0,9	1,0	5,0	10,0
0,01	27	26	26	28	28	26	25	29	29
0,1	250	250	250	250	256	256	256	264	250
0,2	499	502	501	510	508	507	507	499	499
0,4	1007	1001	997	1110	998	997	999	1009	997
0,5	1247	1246	1248	1248	1246	1246	1248	1255	1247

Table 5.2 – Variation of number of support vectors for a nominal representation of data

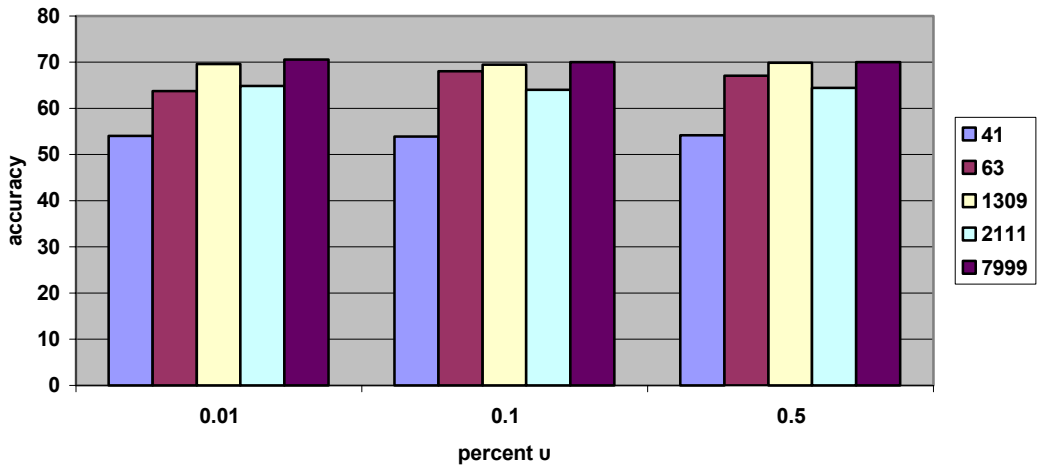

Figure 5.17 – Influence of vector size for C=5

As we can observe the best accuracies are obtained when we have a medium number of features (63 and 1309), the results decrease when the number of features are increased. We can also notice that the best results are obtained for a percentage of features similar to that used in classification (6% -10%). The accuracy decreases from 69.84% for 1309 features to 64.38% for 2111 features.

From Figure 5.16 it can be observed that when we have the smaller number of initially chosen data the value of *C* doesn't mater so much as the accuracy varies only from 60.62% to 64.99%. When the value of initially chosen data increases the coefficient *C* has a greater influence on the accuracy, which varies from 55.11% to 63.88%.

As a conclusion of these first tests we see that the best parameter is C=10 for a small parameter υ (0.01). Because I am also interested in the response time I presented here percentage of support vectors resulted after learning. I present the values in percentages as the number of documents varies.

#features / *υ*	41	63	1309	2111
0,01	0,6%	0,6%	0,7%	0,6%
0,1	0,5%	0,5%	0,5%	0,5%
0,5	25,2%	25,1%	25,1%	25,1%

Table 5.3 – Percentage of support vectors

As we can observe the percentage of support vectors is smaller in comparison with the number of support vectors used for classification. Considering the best results obtained and the smallest number of support vectors the best choice is 1309 features and 1% of initially chosen data. Even if we consider the best value was obtained for υ =0.5 (69.84%) this uses 25% of data as support vectors. For υ = 0.01 the accuracy of classification decreased only to 69.63% using only 0.7% of data as support vectors.

> "Science may set limits to knowledge, but should not set limits to imagination."
>
> Bertrand Russell

6 Designing a Meta-Classifier for Support Vector Machine

Meta-learning focuses on predicting the right (classifier) algorithm for a particular problem based on dataset characteristics [Bra94]. 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. Another advantage of meta-classification is the possibility to exploit the capability of parallel computation offered by multiprocessor computers.

The method of combining multiple results taken from classifiers is not trivial and will determine the effectiveness of the whole system (hybrid classification). 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_1], as more features are included and the features vectors grow excessively. The other one is an 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 being 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_1] a meta-classification strategies using SVM [Lin02_2] is presented and compared with probability based strategies. In [Lin02_1] the authors presented 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 the authors 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.

6.1 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_1]. My meta-classifier is build using SVM classifiers. I do this because in my previous work I showed that some documents are

correctly classified only by some certain type of SVM classifiers. Thus, I put together many SVM classifiers with different parameters in order to improve the classification accuracy. My strategies to develop the meta-classifier are based on the idea of selecting adequate classifiers for difficult documents. My selected classifiers are different through: type of the kernel, kernels' parameters and type of the input data representation. I chose the input representation because, as I 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 6.1 for polynomial kernel and in Table 6.2 for Gaussian kernel. With bold are presented the maximum value obtained for each category. In those tables I 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 6.1 – Possible classifiers for Polynomial kernel

In the Table 6.1 results obtained for different dimensions of the input feature vectors are presented. In chapter 4 I presented some techniques of features selection. Also I showed that the best results are obtained for an optimal dimension of the feature vector (1309 features). Thus, I 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 6.3 I present the selected classifiers, each of them with the specified selected parameters for polynomial and Gaussian kernels.

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 6.2 – Possible classifiers for Gaussian kernel

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 6.3 – The selected classifiers

6.2 The Limitations of the Developed Meta-Classifier System

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

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

6.3 Meta-classifier Models

The main idea that I had when I designed my meta-classifiers was that classifiers should have implemented a simple and faster algorithm in order to give the response. Also I was interested in selecting the adequate classifier for a given input vector. In order to design the meta-classifier I am used three models. First of them is a simple approach based on the majority voting principle, thus without any adaptation. The other two approaches are implementing adaptive methods.

6.3.1 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 I obtain two or more classes with the same value of the counter I classify the current document in all proposed classes. The counters are reset when it is necessary to classify another document. 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%).

6.3.2 Adaptive Developed Methods

6.3.2.1 Selection Based on Euclidean Distance (SBED)

Because the previous presented meta-classifier doesn't obtain such good results I develop a meta-classifier that changes the behavior depending on the input data, being therefore adaptive. To do this, we build a meta-classifier that selects a classifier based on the current input data. Thus, I design my meta-classifier to 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 my meta-classifier will learn only the input samples incorrectly classified. As a consequence the meta-classifier will contain for each classifier a self queue where are stored all incorrectly classified documents. Therefore, my meta-classifier contains 8 queues attached to the component classifiers.

For this approach of the meta-classifier I 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. 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 right classifier. The difference in accuracies is insignificant so I prefer the faster method.

6.3.2.1.1 First Classifier Selection Based on Euclidian Distance (FC-SBED)

Considering an input document (current sample) that needs to be classified, first I randomly chose one classifier. I compute the Euclidean distance (equation 6.1) between the current sample and all samples that are in that self queue of the selected classifier. If I obtain at least one distance smaller than a predefined threshold I renounce to use that selected classifier. In this case I randomly select another classifier. If there are cases when all component classifiers are rejected, however, I will choose that classifier with the greatest Euclidean distance.

$$Eucl(\mathbf{x}, \mathbf{x}') = \sqrt{\sum_{i=1}^{n} ([x]_i - [x']_i)^2} \tag{6.1}$$

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

The complete scheme of evolution for this meta-classifier FC-SBED is presented in Figure 6.1. 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 taken in my 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, I 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, I repeat these two steps many times.

6.3.2.1.2 Best Classifier Selection Based on Euclidian Distance (BC-SBED)

This method follows the method FC-SBED presented in Section 6.3.2.1 with one change. This is that the current tested classifier is not randomly selected. In contrast, I take into the consideration all classifiers. We'll compute the Euclidean distance between the current document and all misclassified documents that are into the queues. I 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 6.2.

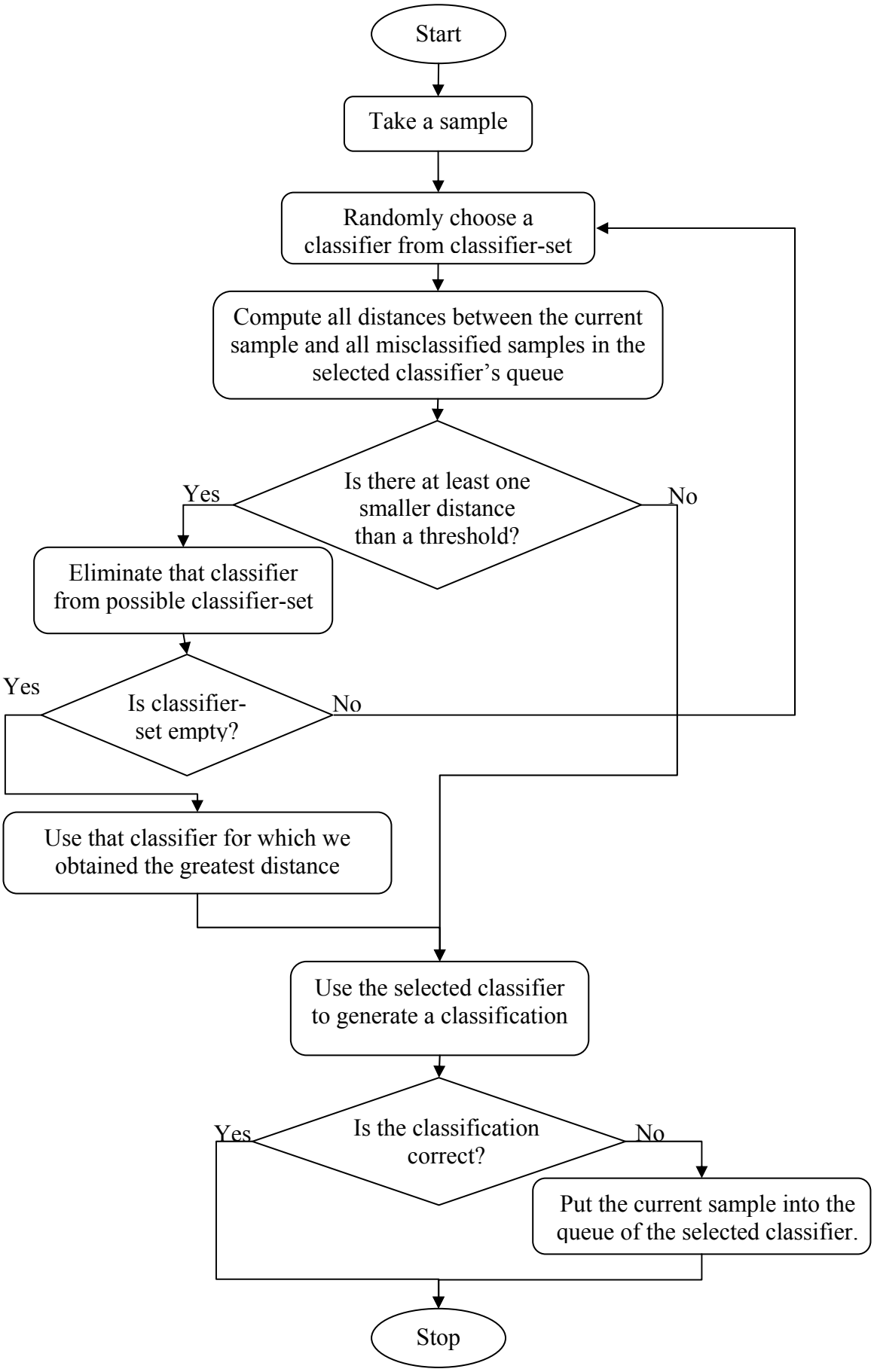

Figure 6.1 – FC-SBED – Meta-classifier diagram

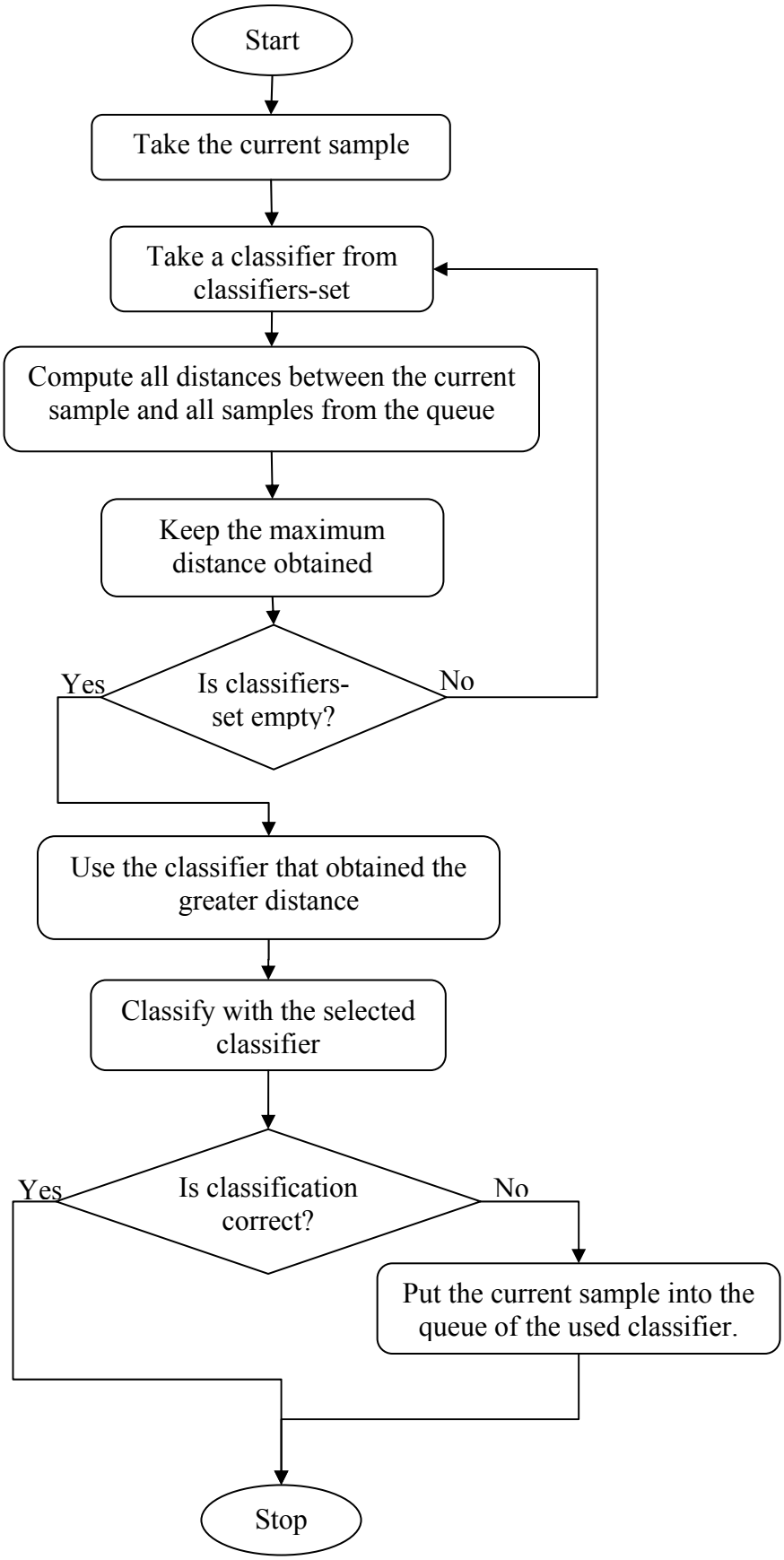

Figure 6.2 - BC-SBED – Meta-classifier diagram

6.3.2.2 Selection Based on Cosine (SBCOS)

The cosine is another possibility to compute the document similarity, usually used into the literature focused on documents' classification. This is based on computing the dot product between two vectors. The used formula to compute the cosine angle θ between two input vectors **x** and **x'** is:

$$\cos\theta = \frac{\langle \mathbf{x}, \mathbf{x}' \rangle}{\|\mathbf{x}\| \cdot \|\mathbf{x}'\|} = \frac{\sum_{i=1}^{n} [x]_i [x']_i}{\sqrt{\sum_{i=1}^{n} [x]_i^2} \cdot \sqrt{\sum_{i=1}^{n} [x']_i^2}} \qquad (6.2)$$

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 $\theta \in \left[2k\pi - \frac{\pi}{2}, 2k\pi + \frac{\pi}{2}\right]$, $(\forall)k \in Z$. In my 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 I implement two methods for selecting the current classifier as for SBED called first classifier selection based on cosine (FC-SBCOS) and best classifier selection based on cosine (BC-SBCOS). In there methods I 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 certain threshold. I will reject them if at least one cosine angle is greater then a threshold.

6.3.2.3 Selection Based on Dot Product with Average

In all presented methods I kept in the queue of each classifier the vector of documents that was incorrectly classified by that classifier. As an alternative to this, I 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 I kept only a single vector representing the partial sum of all the error vectors, and a value that represent the total number of vectors that should be kept (in order to be able to compute the average). This makes the algorithm faster but unfortunately the results are not so good.

6.4 Evaluating the Proposed Meta-classifiers

During chapter 4 I showed that the best results are obtained using a dimension of the feature space around 1309. So, that for the meta-classifier I present results obtained using only this feature dimension. As feature selection I 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 I present I 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 I am doing a short comparison between the influence of selecting the classifier, first good classifier or best classifier, so for similarity I computed both Euclidean distance and cosine angle.

6.4.1 Results for Selection Based on Euclidean Distance

As I 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. I do 14 learning steps with different threshold values. After each learning step I make a testing step. I stop after 14 learning steps because I noticed that after this value the accuracy doesn't increase but it sometime even decreases. In Figure 6.3 I present results for each step as a percentage of correctly classified documents.

In order to have a good view in all following charts, I 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.

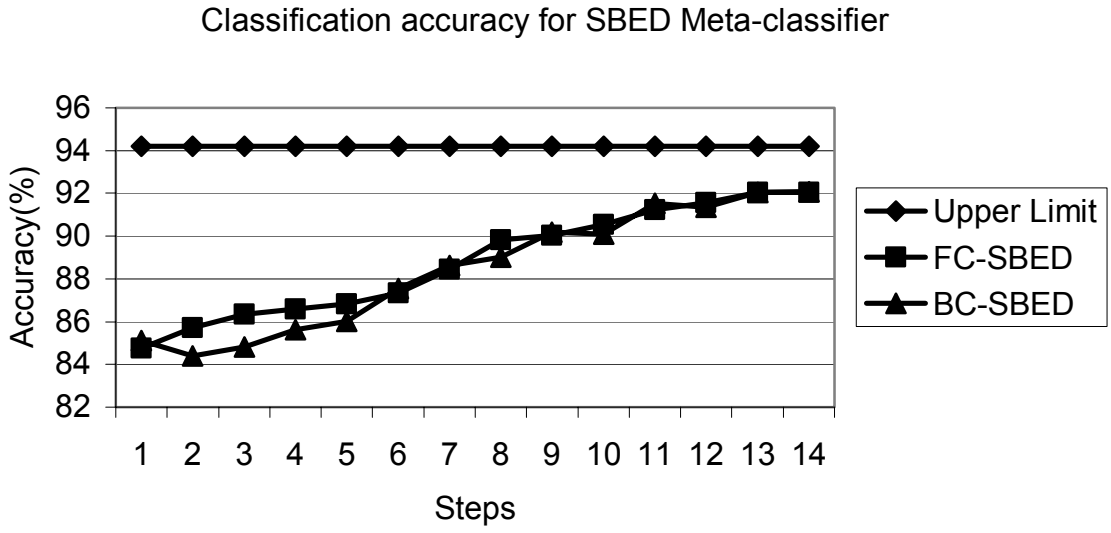

Figure 6.3 - Evolution of classification accuracy for SBED

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 I selected a greater threshold value in order to reject more possible classifiers. When in the queue there is already an error sample I will reject that classifier easier. I make this because in the first steps I am interested in populating all queues from my meta-classifier. In the last 7 steps I 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 13th step. Considering the ratio between my maximum obtained value

and the upper limit we see that my developed learning algorithm reaches 97.69% of its potential. Thus the results obtained after the 13th step can be considered as good results.

In Figure 6.4 I present the response time for methods based on Euclidian distance in order to compute the similarity. Also, for a good view I present the response time for Majority Vote, the method that takes the longest time.

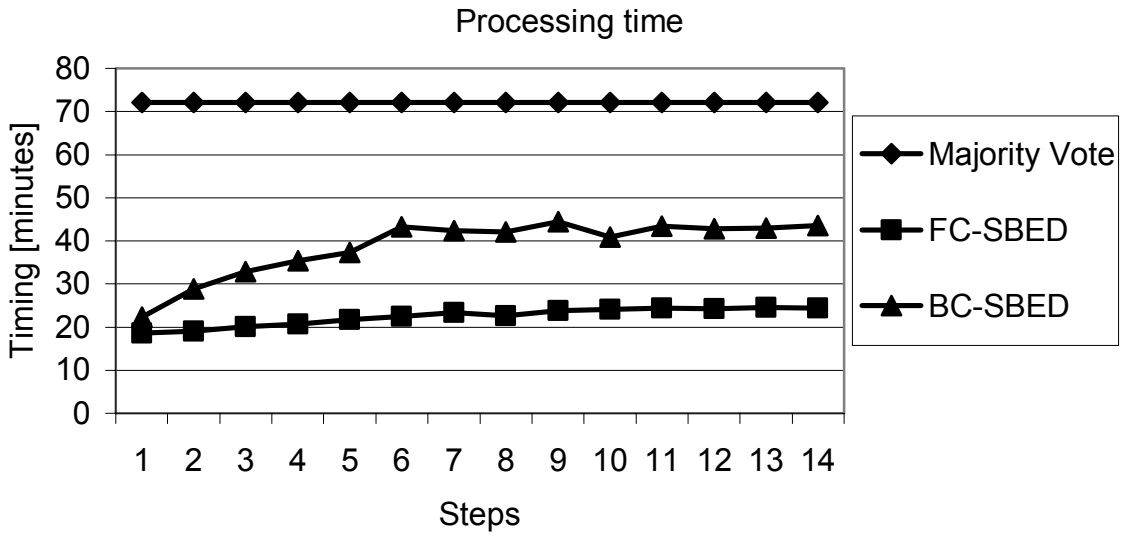

Figure 6.4 – Processing time for SBED

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.

6.4.2 Results for Selection Based on Cosine

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 there methods (SBED and SBCOS) I am also doing 14 learning steps with different values for the threshold. I have also noticed that I obtained good results for the accuracy after the 11 steps similar to the SBED. Each learning step is followed by a testing step. In Figure 6.5 I present results for each step as a percent of correctly classified documents.

In order to have a good view, I will also present the maximum limit for my meta-classifier. In this chart I also put the results obtained with average error vectors stored into the queue presented in section 6.3.2.3.

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. I am also interested to be able to easily reject a classifier in the first 7 steps. This is way I chose a much smaller value than 1, which means much less similar documents. In the last 7 steps I used a value closer to the value that means similar documents. This assures that in the first steps I populate the queues and in the last steps I actually calibrate my meta-classifier for documents which are more difficult to classify. I made also experiments with other values of the threshold between 0.25 and 1 and here I 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 my 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 my 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 important, 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.

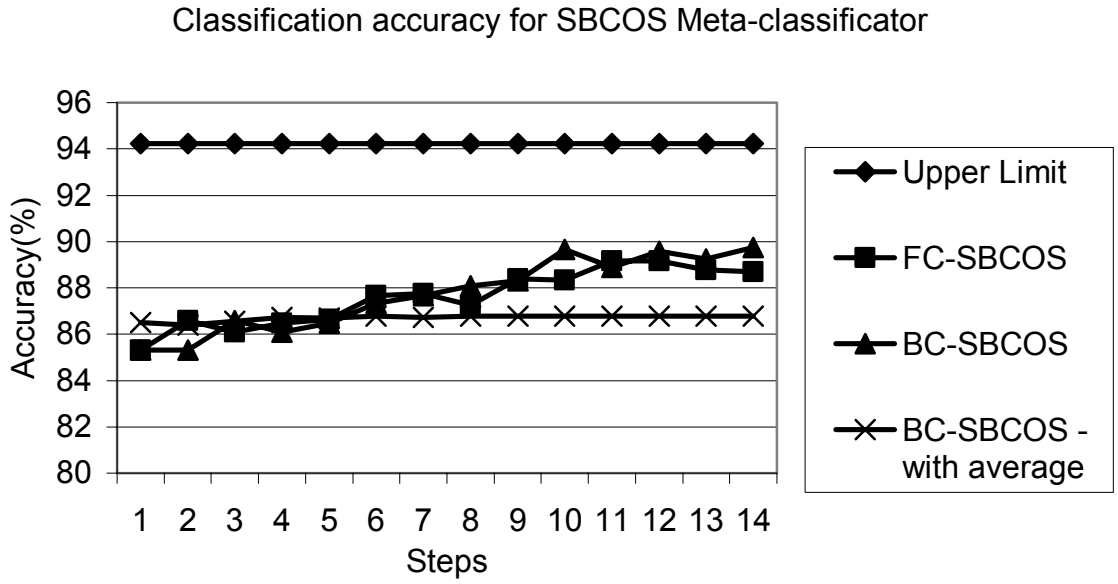

Figure 6.5 – Evolution of classification accuracy for SBCOS

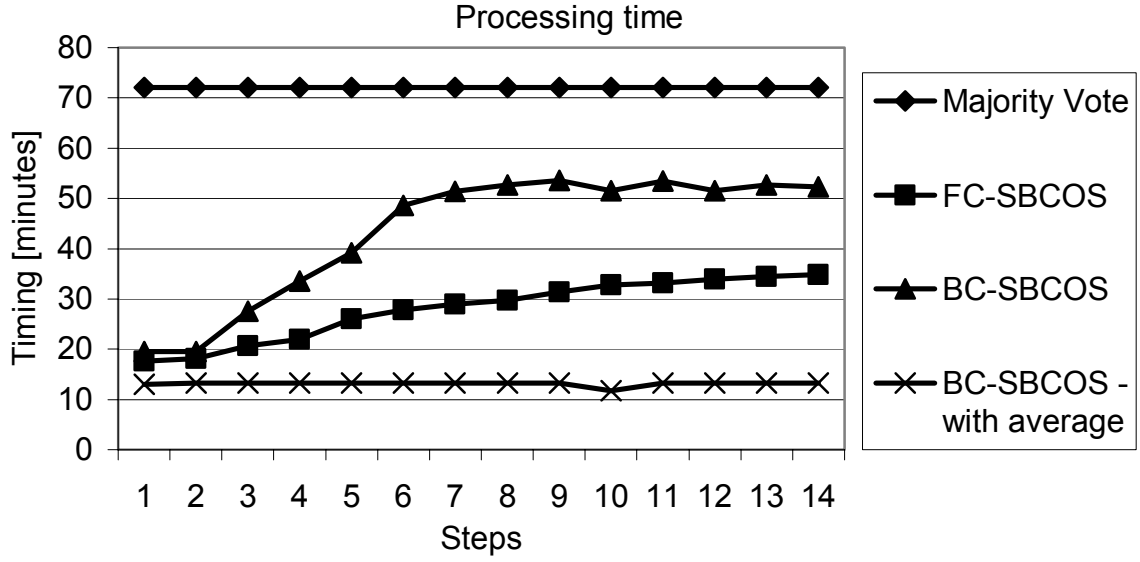

Figure 6.6 – Processing time for SBCOS

Considering the learning time for these two methods the times are considerably different. In Figure 6.6 I present all response times needed to compute these results. Thus, for each of them I put the time needed for training and testing parts together because I think that my 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.

I also present here the time needed for Majority Vote method because I want to have a good view. As I 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 6.5 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 6.7 comparative results between all presented methods used to build the meta-classifier are presented. From SBED I selected results obtained with FC-SBED and for SBCOS I selected results obtained with BC-SBCOS. Also those results were presented in [Mor06_4] and [Mor06_5].

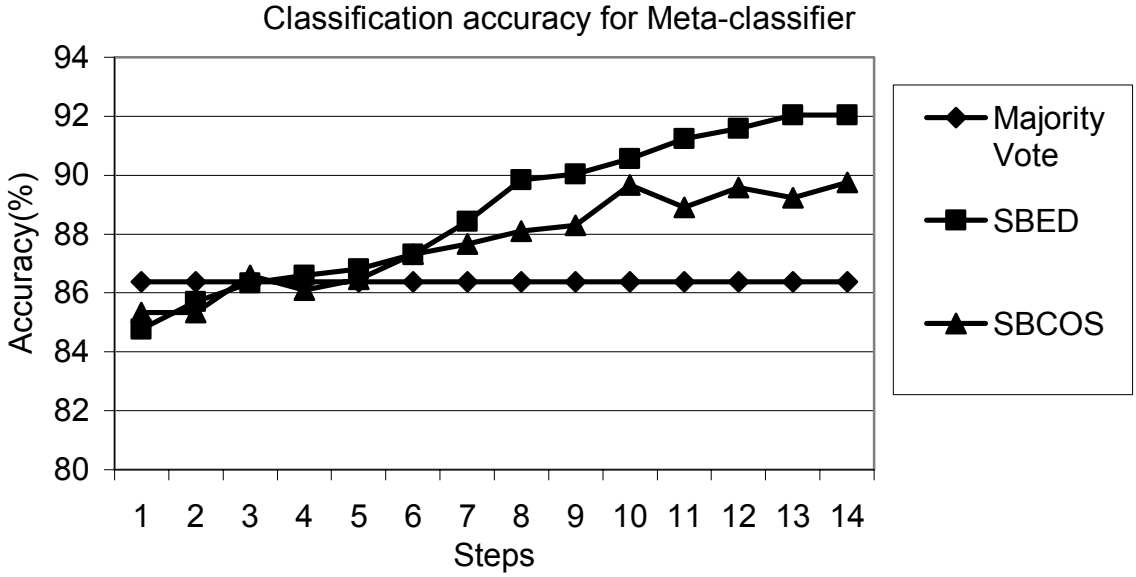

Figure 6.7 – Classification Accuracy for my meta-classifier

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 6.8 I present all response times obtained for the methods presented above. For Majority Vote I put more times the same response time value. For the other methods I 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.

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 I expected.

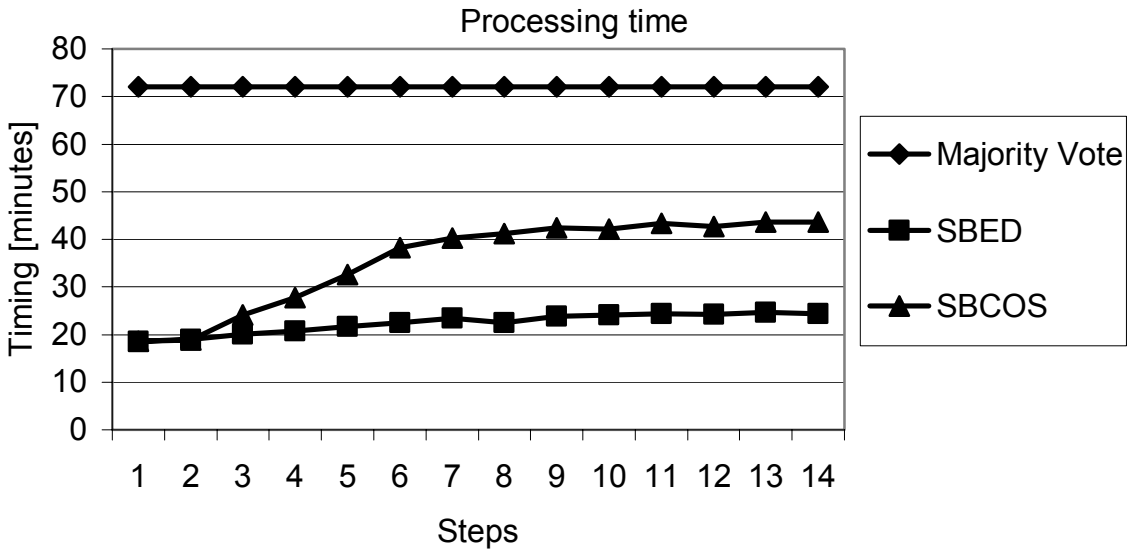

Figure 6.8 – Processing time – comparison between SBED and SBCOS

The difference between those two methods can occur because in SBED case the obtained value are not normalized and it is easier to find a good threshold for accepting and rejecting the classifier. In the SBCOS case the obtained values are between 0 and 1 in our case and in this narrows domain it is more difficult to find a good threshold. This is why I consider that the differences occur because the selected thresholds for the second method are not optimum.

6.5 Analyzing the Meta-classifier Response Time

Training and testing a meta-classifier is time consuming. In this section my intention is to establish if there is justified to improve the part of building and searching in the queue. I am not proposing a method for improving this time here but I am trying to see if justified to further work in this direction. In Table 6.4 are presented the running times for all 14 steps needed to train a meta-classifier. In this table I present results for a meta-classifier based on cosine (because is more time consuming in comparison with SBED method) choosing the BC-SBCOS model.

As can be observed in Table 6.4 when my meta-classifier evolves and the queues increase the training time also increases. Because, for each step, the time needed to evaluate the current training sample is the same (is used the same SVM algorithm for compute the resulting class), the operation producing the increase in time on each step is "selecting a classifier". This operation implies in each step a search in one or more queues. Those queues increase each time we incorrectly classify the current sample. When those queues increase the search in each queue needs more time and this implies an increase in the training times. In the worst case, this time becomes more than 57% from all training time. This justifies the need for a method to reduce this queue or to reduce the searching time. I tried to reduce the queues by keeping into the queue only the samples which are more un-similar than in case when I chose the classifier (when also the similarity is computed). When I constructed the queues I put into one queue only un-similar

samples because if the current sample is similar with one of the samples that are already in the queue the classifier is rejected. This un-similarity is computed based on Euclidean distance or cosine and is compared with a threshold (called first threshold). Using a new threshold (called second threshold) for establishing the similarity when I need to introduce a new sample into the queue the dimension of the queue can be reduced. Compared with the first threshold used for selecting the classifier the second threshold needs to be larger. This second threshold can't be too large because we can fall in the other extreme when we are treating samples from distinct classes as being in the same class. Thus the average between samples can define a new false class that will disturb the process of classifying. Using a second threshold not so large in comparison with the first threshold (for example with 1.5 greater for Euclidean distance or 0.1 for cosine) I obtained an insignificant reduces into the queue, only with 2 or 3 samples per each queue for the entire set. In average in each queue after 14 steps are 200 error samples. If this number is reduced with only 3 samples reduces are insignificant. A better method can have a good impact in reducing the time for training and testing the meta-classifier.

No. classifiers	Threshold	Total Learning time (seconds)	No. of samples	Accuracy	Searching time in queue (seconds)	percents [%]
8	0.8	1574.125	2351	85.11272	184.452	11.71
8	0.8	1979.312	2351	84.38962	554.268	28.00
8	0.8	2246.187	2351	84.81497	833.629	37.11
8	0.8	2411.937	2351	85.62314	1069.995	44.36
8	0.8	2509.562	2351	86.00595	1273.657	50.75
8	0.8	2949.953	2351	87.1544	1460.441	49.50
8	0.8	2879.641	2351	88.13271	1566.787	54.40
8	0.9	2857.22	2351	89.02595	1593.15	55.75
8	0.9	3013.563	2351	87.19694	1627.545	54.00
8	0.9	2832.047	2351	90.08932	1672.573	59.05
8	0.9	3019.172	2351	89.53637	1691.274	56.01
8	0.9	3051.156	2351	89.36623	1740.978	57.05
8	0.9	3108.953	2351	89.36623	1787.822	57.50
8	0.9	3217.922	2351	89.87665	1817.189	56.47

Table 6.4 – Running time for meta-classifier

"Just because something doesn't do what you planned it to do doesn't mean it's useless.

Thomas Edison

7 Research Regarding the Methods' Scalability

Up to this moment I presented a number of methods for text classification and laborious analysis 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]:

- *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.
- *Speed* and *scalability* for training and applying/testing in batch mode.
- *Simplicity*, speed, and scalability for document insertion, deletion, and modification, as well as moving large sets of documents from one class to another.
- *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 I focused on the issue of accuracy and speed, with some comments on performance where appropriate. Now, in this chapter I want to focus on scalability on training documents and applying results.

In the last years the available text data becomes larger and larger and a lot of algorithms were 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].

7.1 Methods for the SVM Algorithm' Scalability

In this chapter I would like to see if there are some huge influences when my algorithms work with large database from which are selected only some relevant input vectors. In order to do this I developed a strategy in three stages 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 I was inspired by a strategy presented in [Yu03] which uses a tree structure to group similar data from databases on levels. This strategy though was not recommended by the authors to be used on text documents, I modified it to work into a single level in order to grope similar documents.

For having comparable results on classification accuracies for this approach of scalability I use as training and testing set the same Reuters set of 7083 samples. This set has a total of 19038 features and 24 topics. The method for extracting this set was presented in section 2.4.

The original learning step that uses the Support Vector Machine technique to classify documents is split in three general stages. In the first stage all training data are grouped based on their similarity. For compute the similarity I use two different methods based on the idea of "winner-take-all" [Hun03]. First method computes the center of the class (called further the representative vector) using arithmetic average and the second method computes the center of the class using LVQ (Learning Vector Quantization) formula [Koh97] in one step. The number of groups that can be created in this stage is unlimited and depends on the similarity level, data dimension and a specified threshold. From each created group a representative vector is obtained. With those vectors I create a new training dataset that will be used in the next stage. The second stage will be the representative vector classification step. After the classification (made using Support Vector Machine techniques presented in chapter 3), besides the classification rules (decision functions) that are obtained, we also obtain the elements that have an effective contribution to the classification (named support vectors in the SVM algorithm). Taking only the support vectors (called further relevant vectors) we make a new reduced dataset, containing only a small but relevant part of the representative vectors obtained in the first stage. For each relevant vector I take all original vectors that were grouped in his category and make a new dataset. In the third stage, a new classification step, I 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 defining the decision function. All these stages might be seen in Figure 1.1, too.

Depending of the method used for compute the representative vector I use two type of representing this vectors.

- For the first method, when the representative vector is computed as a arithmetic average, it is represented as a vector which contain the sum over all elements that are included in that group, and a value that represents the total number of samples from that group. The value is used in order to be able to compute the average on each step.
- In the second method the representative vector is computed using equation 7.1 for each new sample that is included in the group. For this representative vector all the time the value that represents number of samples from that group is 1. The formula for computing the representative vector for the LVQ method is:

$$\vec{\mathbf{w}}_i := \vec{\mathbf{w}}_i + \alpha(\vec{\mathbf{x}} - \vec{\mathbf{w}}_i) \tag{7.1}$$

where $\vec{\mathbf{w}}_i$ represents the representative vector for the class i, $\vec{\mathbf{x}}$ represent the input vector and α represent the learning rate. Because I want that this learn to be done only in one step and for my set I have a small number of samples, I will choose a relatively great value for coefficient α.

In the first stage for each new sample I compute the similarity between this sample and all the representative vectors that was created up to this moment. If there is at least a similarity smaller that a predefined threshold I will put the current sample in that group (describe by representative vector) and recompute the representative vector using one of the two previous presented methods, others, I will create a new group.

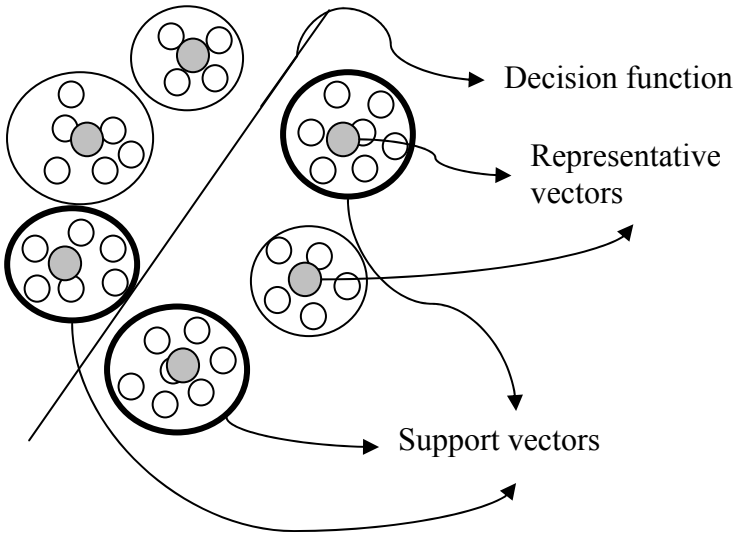

Figure 7.1 – Selecting of support vectors

The entire process (those three stages were presented before) contains the next 7 steps:

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

$$TF(d,t) = \frac{n(d,t)}{\sum_{\tau=0}^{19038} n(d,\tau)} \tag{7.2}$$

   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 I compute the Euclidian distance between each input vector and each representative vector (the gray small circles in Figure 7.1) that was created up to this moment. This makes my strategy slower when I have more groups. The formula for Euclidian distance is presented in equation 7.3.

$$E(t,c_i) = \sqrt{\sum_{k=0}^{19038} (x_k - w_{ik})^2} \tag{7.3}$$

   where x_k represent the terms from the input vectors and w_{ik} represent term from the representative vector of class *i*. Thus, I compute each distance and I keep the smallest obtained distance. If that this distance is smaller than a predefined threshold I will introduce the current sample in the winner group and recompute the representative vector for that group; if not, I will develop a new group and the input vector will be the representative vector for that group.

3. After this grouping (all the large circles in Figure 7.1), I create a new training dataset with all representative vectors. This set will be used in the classification step. In this step I am not interested in the accuracy of classification because the vectors are not the original vectors. Here, we are interested only in selecting relevant vectors from this new set. Because in this step I use a classification method that needs a topic for each vector from the set, I need to specify a topic. This topic is specified automatically as being the most frequent topic that occurs in all input vectors that were grouped together.

4. On this set reduced by the number of vectors (each of them having 19038 features) I make a feature selection step. For this step I prefer to use SVM_FS method presented as well in my second PhD report [Mor05_2]. After computing all weights I select only 1309 features because as I showed this number of features produced 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 kernel's degree equal to 1 because in almost all previous tests I obtained better results with this value.
6. After SVM learning step I chose only those vectors that are support vectors. In the SVM theory the support vectors are those vectors having the α parameter (Lagrange multipliers) greater that 0. They are represented in Figure 7.1 as being the large circles with a thick line. I chose all groups that are represented by those selected vectors and I made a new set only with vectors from these groups. This new set is a reduced original set containing only relevant input vectors that can have an influence on decision function.
7. This set will now be used in the feature selection and classification steps as the original input data but having a smaller dimension (as the number of vectors) and containing only vectors that can really contribute in making the decision.

7.1.1 Clustering Data Set using Arithmetic Mean

In my presented results I start with an initial set of 7083 vectors. After the grouping step I reduce this dimension at 4474 representative vectors that means 63% from the initial set. For this reduction I use a threshold equal with 0.2. On this reduced set I developed a feature selection step using SVM_FS method reducing the number of features at 1309. After that on this reduced set a classification step for selecting the relevant vectors was made. After the classification the algorithm returns a number of 874 support vectors. Taking those support vectors I create a dataset that contains only 4256 relevant samples that means approximately 60% from the initial set. For this new set (reduced as vectors number) I make again a feature selection step, using SVM_FS method, and I select only 1309 features. This new reduced set was split in a training set having 2555 samples and in a testing set having 1701 samples.

7.1.2 Clustering Data Set using the LVQ Algorithm

In order to have a good comparison between these two presented methods I tried to obtain a closer number of vectors in both methods. With the LVQ algorithm for a value of threshold equal with 0.15 and a learning rate equal with 0.9 I obtained after the first step 3487 groups (representative vectors). I used a great value for the learning rate because usually my groups have a small number of vectors and because I am interested in creating the groups in only one step. I expect this method to work better when huge data sets will be used. After creating the groups I continue by developing a feature selection step and reducing the number of features from 19038 to 1309. In the next step, using these smaller vectors, I train a classifier in order to select from these representative vectors only those vectors that are relevant. After this step, the algorithm returns a number of support vectors and I selected only those support vectors that have a Lagrange multipliers greater than a certain threshold. The multipliers Lagrange are normalized before being compared with this threshold. For the presented results the threshold was chosen equal with 0.25. I considered these support vectors as being the most relevant vectors from all representative vectors. I create a dataset

that contains only 4333 samples. This number represents approximately 61% from the initial data set. For this new reduced set I also make a feature selection step and I select only 1309 features. The obtained set is split randomly into a training set of 2347 samples and in a testing set of 1959 samples.

7.2 Methods' Scalability - Quantitative Aspects

I performed tests only for one reduced vector dimension, for a number of features equal with 1309. I used only this dimension because I was interested to see if the classification accuracy goes down excessively when a method to select a small (but relevant) number of vectors from the entire set is applied. For example to see how much the classification accuracy is going to decrease, if the original set is reduced with 40%. For this dimension of the features using the entire set I obtain the best results.

In the Figure 7.2 I presented comparative results obtained for Polynomial kernel and nominal data representation for all three sets (original set noted as SVM-7053, set obtained using average mean for compute the representative vector noted as AM-4256 and set obtained using the LVQ method for computed the representative vector noted as LVQ-4333).

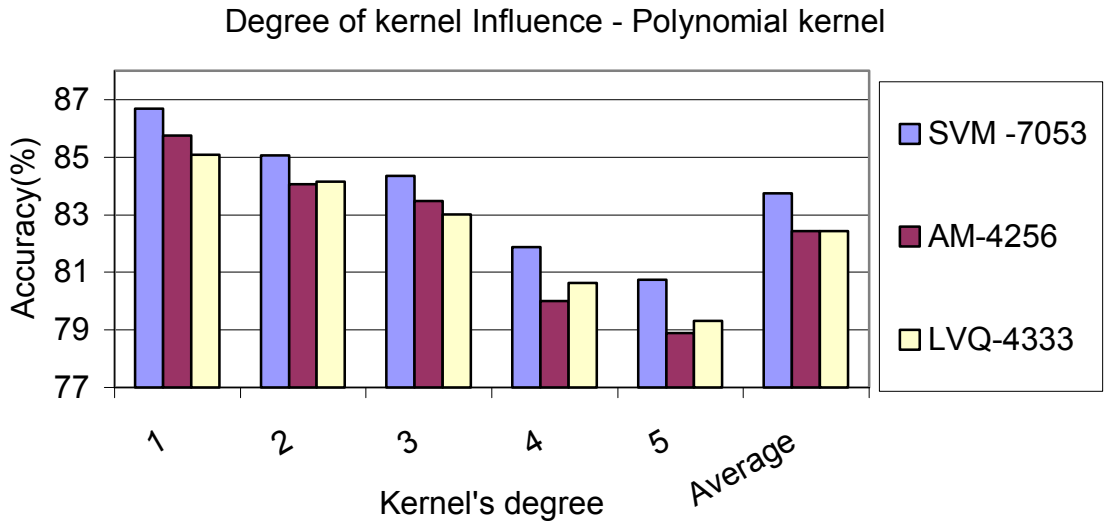

Figure 7.2 – Comparative results for different set dimensions – polynomial kernel

As it can be observed there is a small difference between results obtained for AM-4256 and LVQ-4333. The difference in the accuracy obtained between the original set and AM-4256 set is on average equal to 1.30% for all kernel degrees and nominal data representation. The same difference in average is obtained also between the original set and LVQ-4333. When we work with a small degree of the kernel the difference between original set and AM 4256 set is smaller than 1% but the difference between original set and LVQ-4333 is a little grated (1.60%). When the kernel degree increase results are better with LVQ-4333 comparatively with AM-4256 but usually the difference can be considered insignificant. For example at average over all kernels' degree and all data representation the difference between original set and AM-4256 is 1.65% and the difference between original set and LVQ-4333 is 1.64%. I observe that, for same values of kernel degree for that was obtained the best accuracy with original set, was obtained also the smallest difference between original set and reduced set.

In Figure 7.2 it is interesting to observe that for small degree of kernel we obtain better results using first method for creating the groups and for the grate kernel degree we obtain better results using second method for creating the groups. In the theory of Support Vector Machine researcher recommend to use great value for the degree in case where data are overlapped for obtaining better results. In the case when data are not strongly overlapped with small degree are obtained better results and with greater value of degree the accuracy of classification decrease. Obtained results suggest that with the first method usually we haven't obtain overlapped data but with the second methods based on LVQ we obtain overlapped data. This observation occurs also for Binary and Cornell Smart data representation and Polynomial kernel (see Table 7.1).

In Table 7.1 are presented all the accuracies obtained for all three type of data representation and for all kernels degree. As I expected when the training set has been reduced (in my cases with 40% using arithmetic average and with 39% using LVQ with one step) the quality of learning had decreased, but using a method for extracting only relevant vectors from the entire set the decrease was small. For polynomial kernel at average over all results the decrease was only with 1.65% for AM-4256 and with 1.64% for LVQ-4333. An interesting note can be observed when the kernel's degree increases (and the accuracy of classification decreases) the discrepancy between original set and both reduces sets decreases too.

Kernel type	Data representation	Kernel degree	SVM-7083	AM-4256	LVQ-4333
POLYNOMIAL KERNEL	BINARY	1.0	81.4547	79.65902	79.01991
		2.0	86.64398	83.83304	83.512
		3.0	86.09103	84.06878	83.52118
		4.0	82.1778	81.14109	81.58933
		5.0	76.94598	75.07525	75.54875
	NOMINAL	1.0	86.68652	85.74456	85.07351
		2.0	85.07018	84.0682	84.15467
		3.0	84.34709	83.48089	83.01072
		4.0	81.88005	80.01235	80.63247
		5.0	80.7316	78.89477	79.30526
	CONNEL SMART	1.0	80.98681	79.65902	79.93874
		2.0	86.64398	83.71546	83.02246
		3.0	86.1761	84.77484	84.13374
		4.0	79.84	77.56026	77.96631
		5.0	78.22	77.40623	77.88627

Table 7.1 – All accuracies obtained for all kernel degree for polynomial kernel

I make also some tests with different values for thresholds and learning rate (α) and this value only increase more or less the number of groups. Usually for both methods the threshold is the main parameter that adjusts the number of classes. The parameter (α) have only a small influence in determining the number of classes (depending of input set dimension and the on-coming of

samples), but have a great influence in specifying the center of the class. Parameter (α) in this tested case has a small influence because I worked with relatively reduced number of samples.

Even if the LVQ-4333 uses more samples for training in the second step comparatively with AM-4256 the results are usually poor because in the first step the representative vectors aren't able to represent exactly the center of the group; the selection of the representative vectors maybe can have a great influence for defining the hyperplane even if not all there vectors will be relevant in the final step.

Now, in Figure 7.3 are presented results obtained for Gaussian kernel and Cornell Smart data representation. For this kernel the average accuracy difference between those two sets is greater than in the polynomial kernel case, being at average of 1.89% for AM-4256 and 1.95% for LVQ-4333. Note about "when we obtain the best accuracy it is obtained also the smallest difference" is kept also for Gaussian kernel too. The smallest difference was obtained with a parameter *C* equal with 1.8, value for which in all previous tests I obtained the best results. This difference has been of 1.5% for AM-4256 and of 1.75% for LVQ-4333. For this type of kernel the method based on LVQ obtains poorly results all the time.

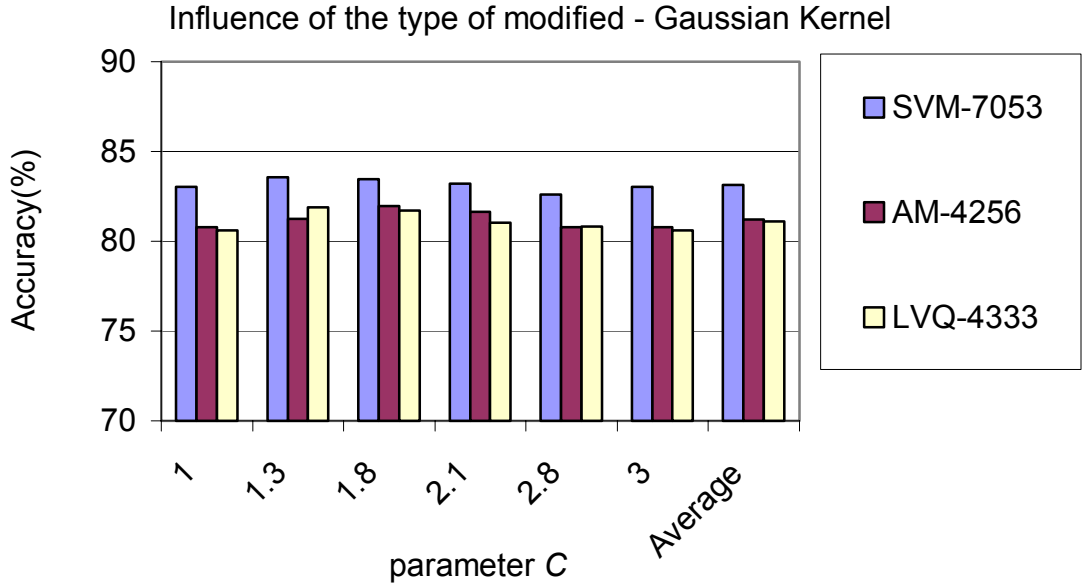

Figure 7.3 – Comparative results for different set dimensions – Gaussian kernel

In Table 7.2 is presented all results obtained for Gaussian kernel and two types of data representation. For Nominal representation I haven't presented any test because usually results obtained with Gaussian kernel and this type of data representation are poor.

I reduced the data in the first step at 63% from the initial set and in the second step at 60% from the initial set for first method and respectively to 50% in the first step and 61% in the second step for the LVQ method. With this reduction however the lose in accuracy was about 1% for polynomial kernel and about 1.8% for Gaussian kernel. It is interesting to note that the optimal parameter values (degree or *C*) are usually the same for the original data set and respectively the reduced one. Maybe the results are poorly for the second method because in the first step data reduction is more significantly. It is difficult that working only with the threshold and learning rate to obtain same value with both methods.

Kernel type	Data representation	Parameter C	SVM-7083	SVM-4256	SVM-4333
GAUSSIAN KERNEL	BINARY	C1.0	82.98596	79.01235	78.96886
		C1.3	83.74	79.7766	79.17305
		C1.8	83.24117	79.71781	79.91782
		C2.1	83.11357	79.30629	79.99898
		C2.8	83.02	78.7184	77.79479
		C3.0	82.98596	79.01235	78.96886
	CORNEL SMART	C1.0	83.0285	80.77601	80.60235
		C1.3	83.58145	81.24633	81.90863
		C1.8	83.45385	81.95179	81.70444
		C2.1	83.19864	81.65785	81.04084
		C2.8	82.60315	80.77601	80.83665
		C3.0	83.0285	80.77601	80.60235

Table 7.2 – All accuracies obtain for Gaussian kernel

Obviously, the time needed for training on smaller number of samples decreases. For example for polynomial kernel with degree 1 and Nominal data representation and first method used for grouping data need 1031 seconds to learn using all dataset and 209 seconds to learn using the smallest one. At this 209 second I need to add also the time needed to select support vectors that were equal with 548 seconds and the time needed for grouping data (84 seconds). The last two times occur only one time for all the tests with polynomial and Gaussian kernels. The total time for polynomial kernel and degree 1 is 892 seconds. To compute these times, in both cases (with original set or whit reduced set), I don't take into consideration the time needed for feature selection. All the time in the feature selection step I start with 19038 features but in the second case I have a reduced dimension of the set (as number of vectors). Some of these times are also smaller than the first times. These values were given using a Pentium IV at 3.2 GHz and 1GB DRAM memory. For the second method to obtain the representative vectors (using LVQ) the grouping part takes more time (97 seconds) depending of the number of groups that are created and the time for selecting support vector is of 571 seconds. The time needed for computing the accuracy of classification for polynomial kernel and degree 2 is for example 232 second, so that the total time that can be considered can be computed as:

$$\mathbf{t}_{total\_time} = \mathbf{t}_{group\_time} + \mathbf{t}_{select\_SV} + \mathbf{t}_{classify} \quad (7.4)$$

where $\mathbf{t}_{group\_time}$ is the time needed for grouping data, $\mathbf{t}_{select\_sv}$ is the time needed for classifying data and find support vectors and $\mathbf{t}_{classify}$ is time needed for classify reduced new set. This time can also include the time needed for selecting the features. But this time is not included in the classifying time using all data set so that will not be included also here.

In Table 7.3 are presented some training times needed for training each data set. I am interested only in training time because after training the testing time depends only of the testing set dimension. For only a sample the response is less than one second.

Data set	Characteristics (Kernel, degree, data representation)	t_{group_time}	t_{select_SV}	$t_{classify}$	$t_{training\_total}$
SVM-7083	POL, D2.0, BIN	-	-	-	1532.578
	POL, D1.0, NOM	-	-	-	1031.219
	POL, D1.0, CS	-	-	-	1107.641
	RBF, C2.8, BIN	-	-	-	4492.953
	RBF, C3.0, CS	-	-	-	4481.657
AM-4256	POL, D2.0, BIN	84	548.46	263.36	945.82
	POL, D1.0, NOM	84	548.46	209,625	892.085
	POL, D1.0, CS	84	548.46	215,563	898.023
	RBF, C2.8, BIN	84	548.46	511,875	1194.335
	RBF, C3.0, CS	84	548.46	513,813	1196.273
LVQ-4333	POL, D2.0, BIN	97	571.43	289.407	957.837
	POL, D1.0, NOM	97	571.43	232.532	900.962
	POL, D1.0, CS	97	571.43	250.672	919.102
	RBF, C2.8, BIN	97	571.43	599.141	1267.571
	RBF, C3.0, CS	97	571.43	618.047	1286.477

Table 7.3 – Training time for each data set and some training characteristics

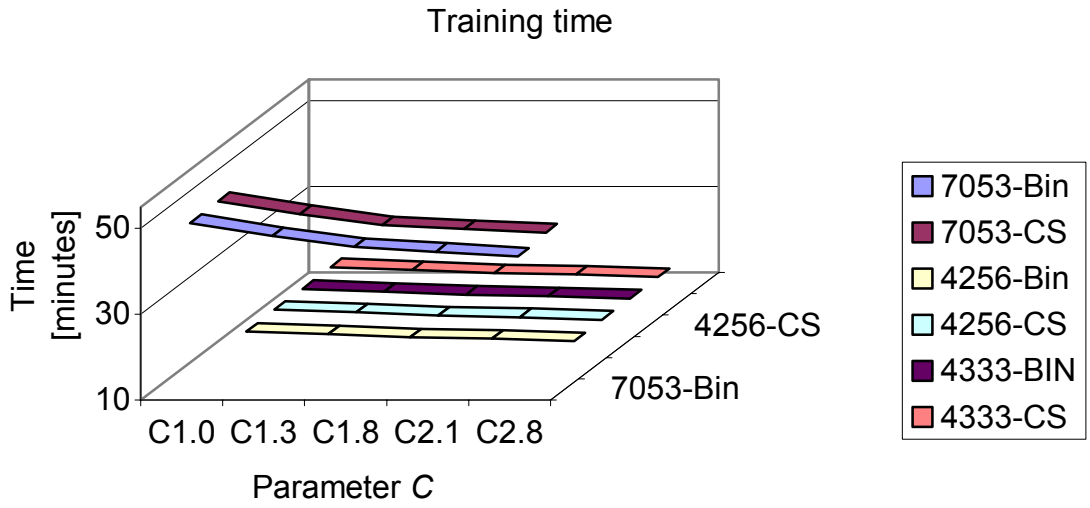

Figure 7.4 – Comparative training time for Gaussian Kernel

Figure 7.4 presents comparatively the times needed for training using original set with times needed for training using 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. We also remind that for the reduced time we need 2 feature selection steps, both of them with a smaller dimension (in

number of samples) in comparison with the original set. These two feature selection steps need a smaller time in comparison with a feature selection step that used all dataset.

In Table 7.4 the average difference between the accuracy obtained with the original set and the accuracy obtained with the reduced set is presented. In other words, I present here the average decrease obtained for each type of data representation and for each kernel when I work with a reduced set. For Gaussian kernel and Binary data representation I obtain the greatest decrease from all the tests (3.8%). For polynomial kernel the decrease is an average of 1.65%.

Kernel type	Data representation	Average accuracy [%] AM-4256	Average accuracy [%] LVQ-4333
POLYNOMIAL KERNEL	BINARY	1.907259	2.124463
	NOMINAL	1.302935	1.307764
	CONNEL SMART	1.750215	1.943873
GAUSSIAN KERNEL	BINARY	3.816914	3.755501
	CONNEL SMART	1.891519	1.954537

Table 7.4 – Decrease in average accuracy for all data representation

Maybe, when I 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. I don't make those tests with all databases because this is not my interest for this PhD and those tests usually take a lot of time. For example, for the entire Reuters database I only started the first step of feature extraction and after this step I had obtained 806791 vectors, each of them having 310033 dimensions (features) and a total of 103 topics.

7.3 Expected Results using a Larger Dataset

I chose a significantly larger training and testing datasets in order to test the presented idea. I can not make tests for this larger data set due to its dimensionality and the limits of my computational capability but I reduced it and perform tests only for the reduced dimension. Based on the obtained results from the previous section I will try to estimate the performance that can be obtained if the entire set will be used. The modality for obtaining this large data set and its characteristics was presented in section 2.4.3.

My intention is not to classify directly this large set. First I tried to create small groups in order to reduce the dimension following the steps presented at the beginning of this chapter (section 7.1) and depicted in Figure 7.1. To create the groups I use the method based on LVQ algorithm executed in just one step. Thus, using a threshold equal with 0.15 and a learning coefficient α=0.4 (smaller because a large number of vectors are used) after a first step I obtain 11258 groups that represent a reduction at 69% of the entire set. Using this reduced set I start training using SVM algorithm in order to select only the support vectors. After training I select only those support vectors that have values greater than a threshold equal with 0.2 (only relevant vectors). In the second step I obtained only 10952 samples that are split randomly in a training set of 5982 samples and a testing set of 4970 samples. In what follows I present the results obtained using only this reduced set that means 67% of the entire set. If the scalability is kept according to my figures

presented in the previous paragraph, where the results are usually with 1-2% smaller if the reduced set is used compared with used the entire set, I will expect that if the entire large set will be used my accuracy will be with 1-2% higher than the presented results.

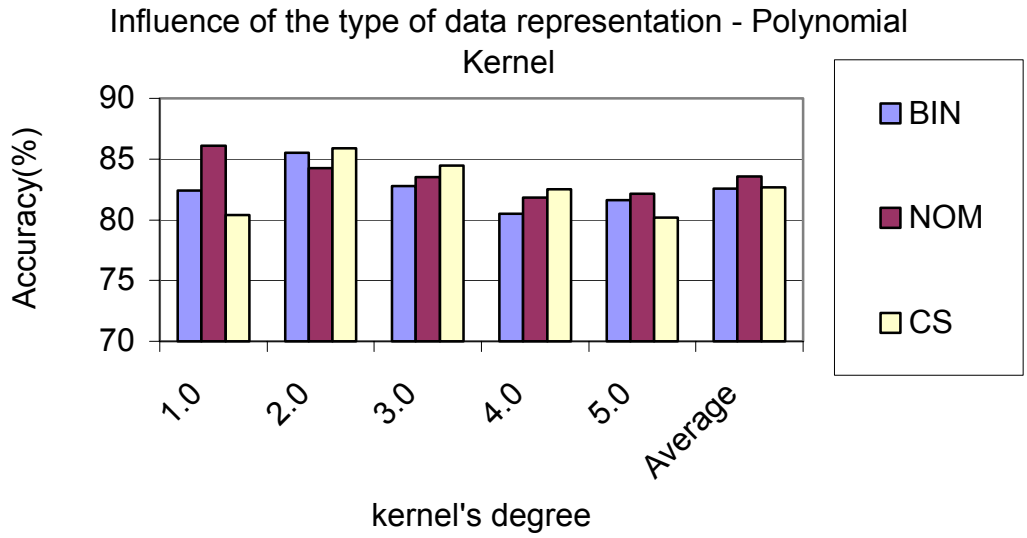

Figure 7.5 – Influence of the type of data representation for Polynomial kernel

I run a lot of tests for polynomial kernel and a lot of tests for Gaussian kernel for the three types of data representation. I present results comparatively for each type of data representation. Figure 7.5 presents results obtained for polynomial kernel and all types of data representation. The presented results are obtained using a set with 3000 relevant features. For features selection I used a SVM_FS method that was presented in section 4.4. In the Figure 7.5 legend BIN means Binary data representation, NOM means nominal data representation and CS means Cornell SMART data representation. Even if I work on the reduced large set the results are comparatively with results obtained working with an entire set but with a smaller number of samples (7053 samples). For example for polynomial kernel and nominal data representation using a set of 7053 samples I obtained an average accuracy of 83.73% and using this reduced large dataset I obtained 83.57% in average.

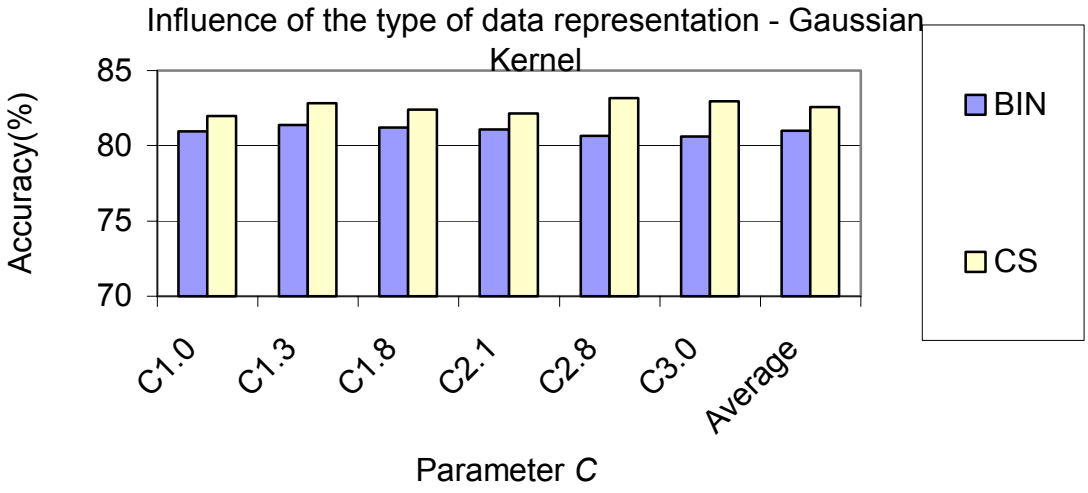

Figure 7.6 – Influence of C parameter (Gaussian kernel)

Figure 7.6 present results obtained using Gaussian kernel and several values for parameter C and two types of data representation. For Gaussian kernel in average with this new data I obtained for Cornell Smart data representation a average accuracy of 82.57% which is closer to average accuracy obtained with a set with 7053 samples 83.14%.

All the presented results are obtained using a reduced set with 65% comparatively with the initial set. I will not present here the time for training and testing because I used on the training part only the reduced set and I don't have results for the training time of the entire set. For the reduced set the training time are computed using the equation 7.4.

7.4 Splitting the Training and Testing Data Set

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

In this chapter I investigate if my algorithm was optimized only for this particular data set or this optimization is valid for any data set. In order to answer to this question I also created different sets that have the same dimension as the first sets but having other random method to split documents in training and testing sets. First random method chose almost randomly where the current document is put, in the training or in the testing set. I said almost randomly because there is one restriction, the training set needs to be larger then the testing set. For example this method assures that from three samples one of them (randomly chosen) is put in the testing set and other two are put in the training set. The second method splits the data in a different way, for example starts with nine samples, and puts three of them (randomly chosen) in the testing set and the others six in the training set. 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.

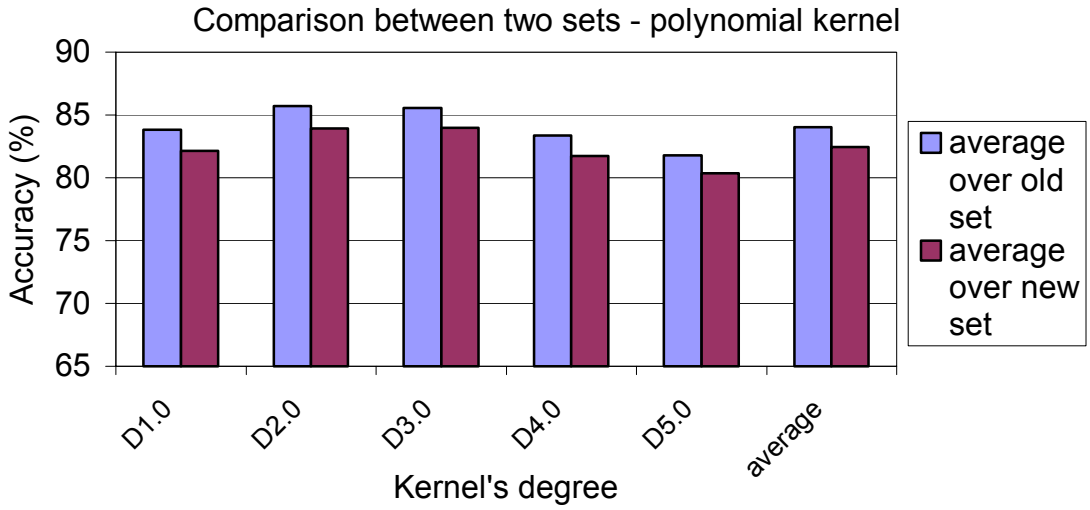

Figure 7.7 – Comparative results for a set with 475 features

I don't repeat all experiments with these new sets. I chose to repeat only experiments with SVM_FS method for different dimensions of the feature space. In the Figure 7.7 I show a comparison of results for Polynomial kernel and 475 features. I present here the average over all three methods of representing the input data.

The last column represents an average over all obtained 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. I obtain an average accuracy 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.

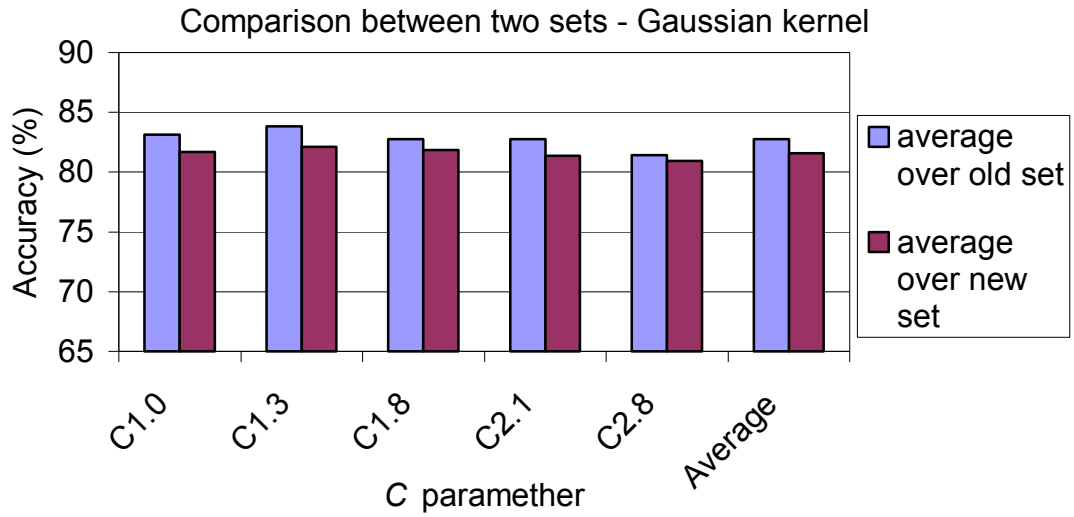

Figure 7.8 – Comparative results for a set with 475 features

When using the same dimension of the feature set but with the Gaussian kernel I 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 I have the same number of features those features are different. Because I splat the data differently my 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 I present results obtained for a dimension of 1309 features. For this dimension I 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).

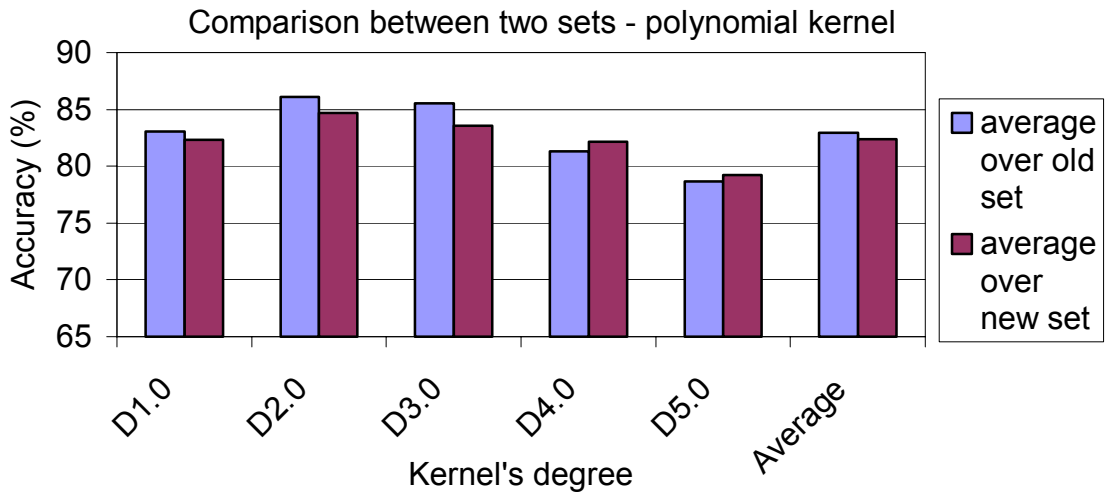

Figure 7.9 – Comparative results for a set with 1309 features

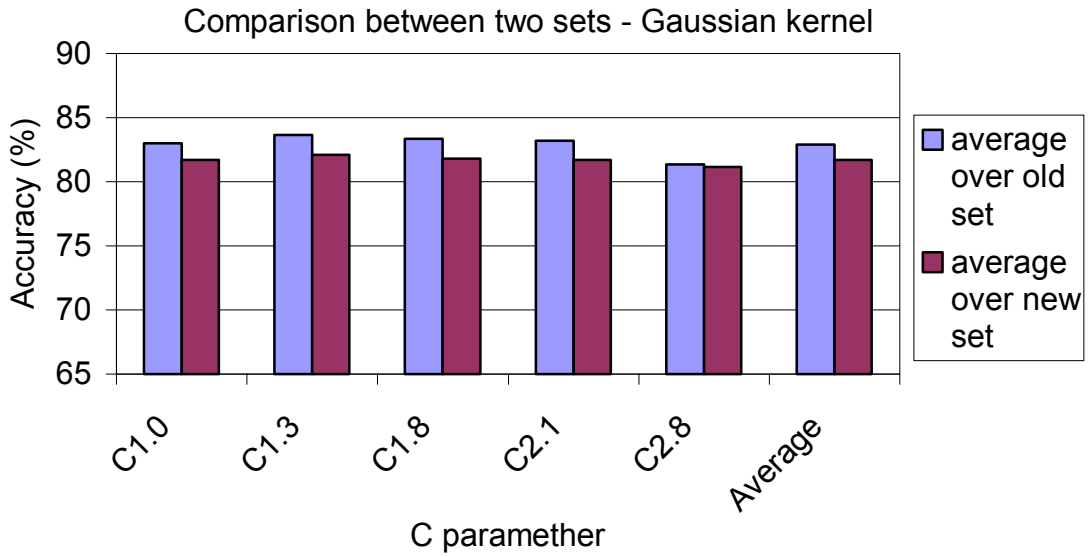

Figure 7.10 – Comparative results for a dimension of the feature space of 1309 features

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.

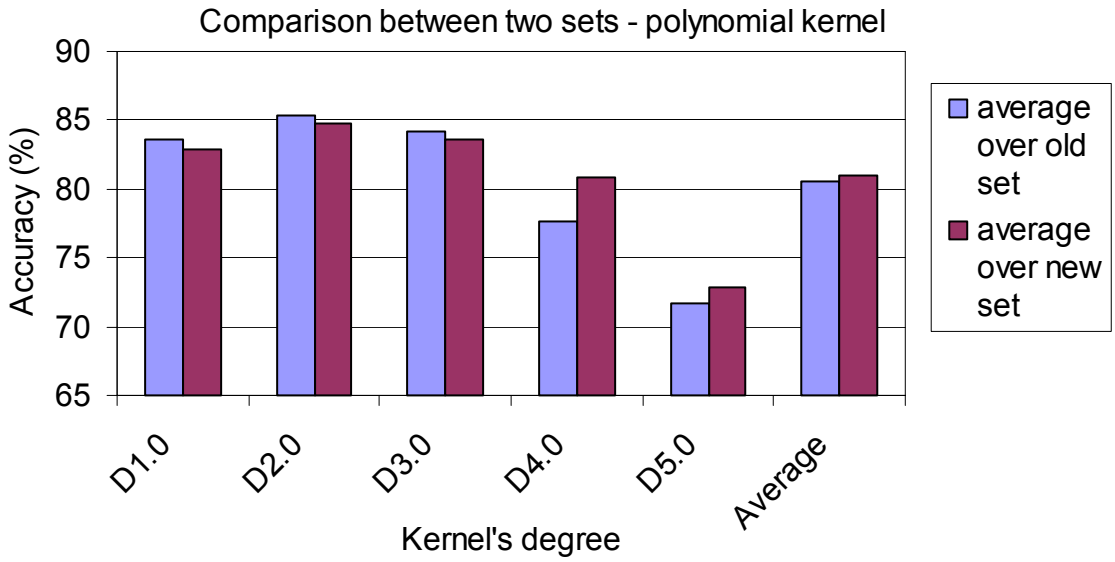

Figure 7.11 – Comparative results for a set with 2488 features

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

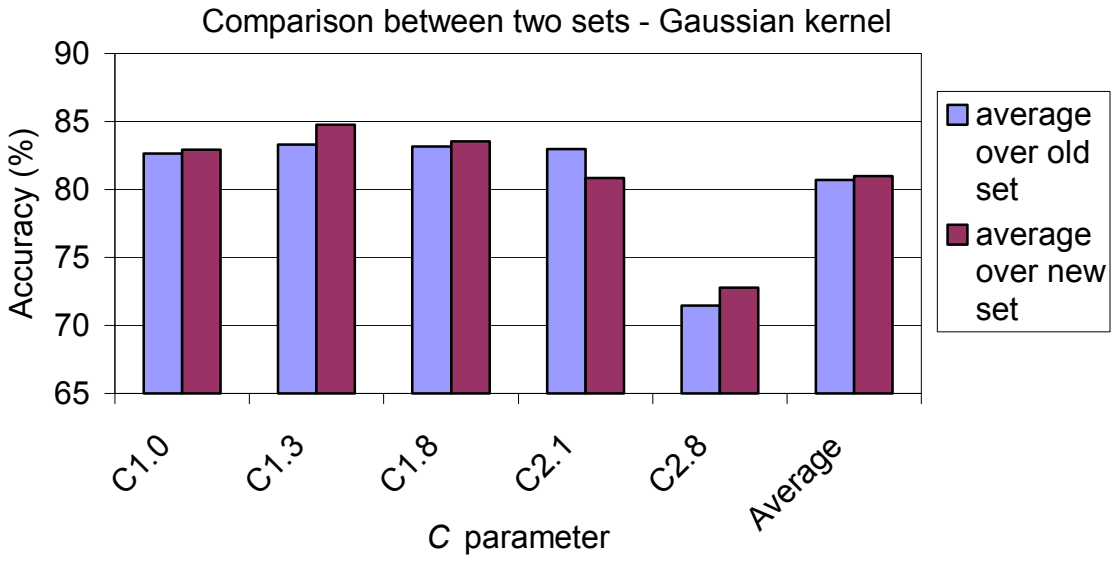

Figure 7.12 – Comparative results for a dimension of the feature space of 2488 features

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.

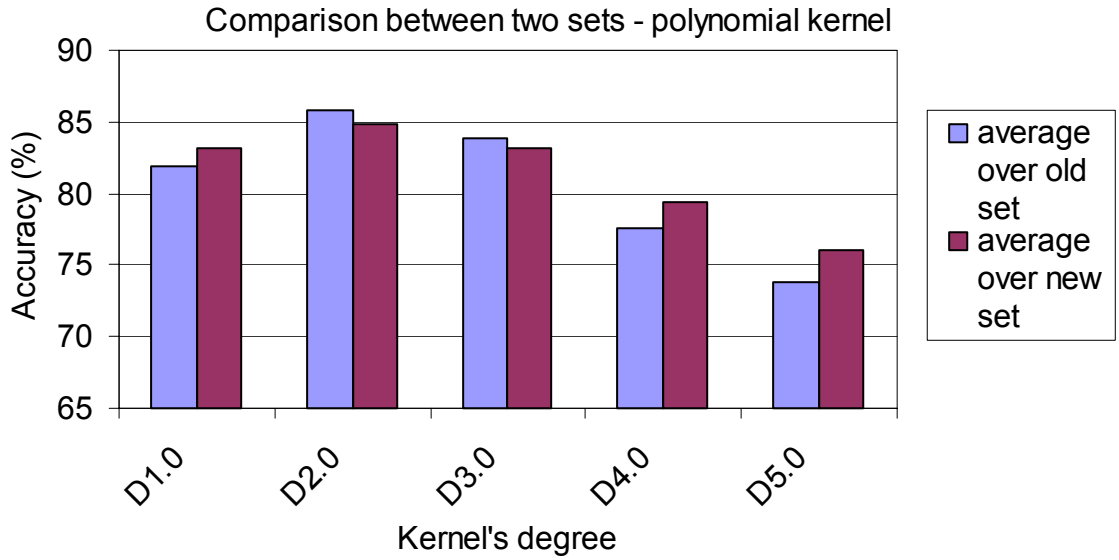

Figure 7.13 – Comparative results for a set with 8000 features

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. At average for all types of input data representation (Binary, Nominal or Cornell Smart representation) and all kernel degrees I obtain with newly selected sets slightly better results. For the Gaussian kernel, represented in Figure 7.14, 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. At average for the first sets I obtain 82.01%, and 81.59% for the second sets.

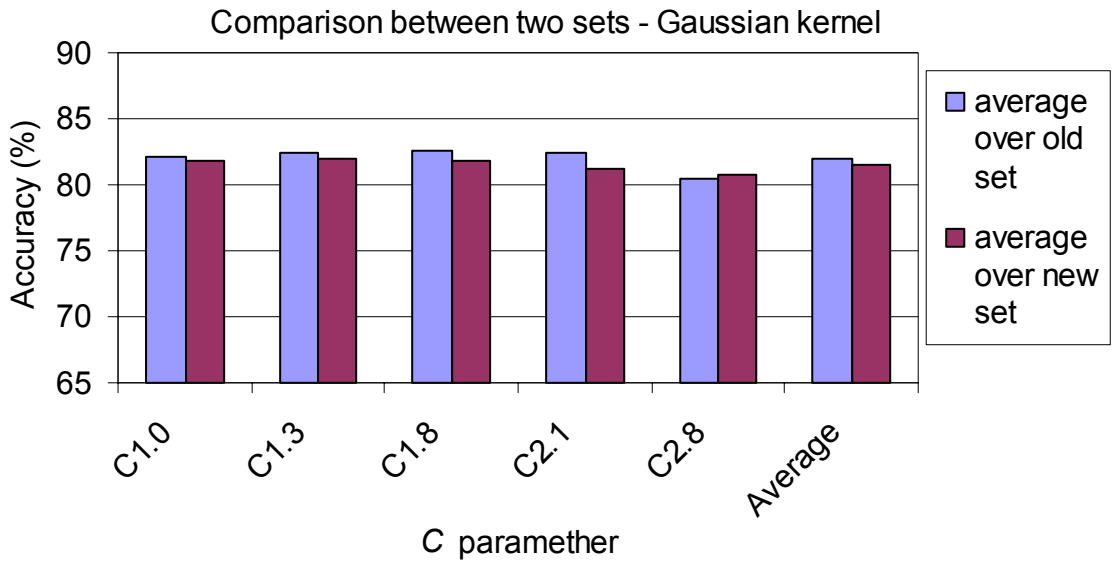

Figure 7.14 – Comparative results for a dimension of the feature space of 8000 features

Here I present results obtained using a features selection method based on Support Vector Machine and after selecting different feature dimension sets I test the accuracy of classification with my 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 my entire work are not influenced on the selected sets. With other sets my features selection methods and my classification algorithm obtain comparable results. This conclusion encourages us to use my classification in real life to classify Web documents.

The most credible results would be the average over each value obtained with each of the set. In order to have this kind of credibility I would have had to run all my 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 I resumed my experiments only to two different groups.

"An expert is a person who has made all the mistakes that can be made in a very narrow field.

Niels Bohr

8 Conclusions

8.1 Author's Original Contributions

This PhD thesis presents the author's work in the field of document classification especially in text document classification. From author's point of view, the original contributions developed in this domain can be presented as follows:

The author starts with **Chapter 1** where he presents an overview of this PhD thesis. The author is seeing the process of automatically documents classification as a flowchart briefly presenting the onset parts (see Figure 1.1).

In **Chapter 2** the author presents a state-of-the-art review of the methods used in database, text and Web documents classification putting the accent on text and Web documents classifying techniques. In this chapter are presented theoretical backgrounds of this domain and also some databases used for automatically document classification. There are also presented methods used for preparing dataset for text documents classification and dataset for Web documents classification.

The author presents in **Chapter 3** a mathematical background for the algorithm used for classification. The author also presents the modality used for implementing a powerful classifying algorithm – Support Vector Machine (SVM). Here it is broached a modality to make this algorithm to work with unlabeled documents. At the end of this chapter the author presents a new method for correlation of the SVM kernel's parameters that lead to improvements of the classification accuracy and simplifies the modality of selecting the parameter in order to hit all the time the best results. This original method correlates the degree of the Polynomial kernel with the bias, respectively correlates the parameter from the Gaussian kernel with a value that represents the number of distinct features that occur into the currently used vectors having weights greater than 0.

In **Chapter 4** the author presents four methods for selecting relevant features. First of the presented methods, Random Selection, is used due to its simplicity and to justify the necessity of using more powerful selection methods. The second method, Information Gain (IG), is a method usually presented in the literature and used here only as a comparison point with the last two developed methods. A powerful method based on SVM algorithm (SVM_FS) is the third method developed. This method is based on linear kernel and benefits of kernel's parameter correlation offering better and faster results. The last proposed method, is a new feature selection method that combines the tenseness and rigorous mathematical techniques based on kernels – SVM, with an algorithm inspired from evolution theory - the genetic algorithm (GA_FS). This new proposed method makes the process of feature selection faster without degrading the performance. Also the author tests the influence of representing the input data on classification accuracy. He tests thus three types of data representation: Binary, Nominal and Cornell Smart. At the end the author presents some comparative results which show that in the case of multi-class classification, the

best results are obtained when a small (but relevant) dimension of the dataset is chose. After selecting relevant features, the author shows that using between 2.5% and 7% from the total number of features, the classification accuracies are significantly better (with a maximum of 87.11% for SVM_FS method, polynomial kernel and Cornell Smart data representation). If the number of features is further increased more than 10%, the accuracy does not improve or even decreases (to 86.52% for 2488 and to 85.36% for 8000 features). When SVM_FS is used, better classification accuracy is obtained using a smaller number of features (85.28% for 475 features representing about 3% of the total number of features)-needing smaller training time too. Generally speaking, the SVM_FS and GA_FS methods are better than IG and Random methods and both obtain comparable results.

The author also showed that the polynomial kernel obtains at average better results when it is used with a nominal data representation and the Gaussian kernel obtains at average better results when it is used with Cornell Smart data representation. The best accuracy is obtained by the polynomial kernel with degree two and 1309 features (87.11% for Cornell Smart representation) in comparison with Gaussian kernel that obtained only 84.85% (for C=1.3 and Cornell Smart representation). The GA_FS method obtains the best results for a greater numbers of features (8000). Also the author 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 the number of features increases from 1309 to 2488. As far as he knows, the author is first one that proposes a feature selection method using Genetic Algorithms with SVM for calculating fitness function and a simplified chromosome structure.

At the end of this chapter the author presents a study realized using HTML Web documents. The author analyzes the influence of the number of features in improving the classification accuracy. The best results are also obtained for a small number of features. Tests were done on 500, 1000, 1500, 2000, 2500, 5000, 10000 and 25646 features. For Web document classification are obtained the same conclusions as those obtained for Reuters' documents classification. For example, for polynomial kernel with degree 1 and Cornell Smart data representation the classification accuracy has only a small decrease from 86.89% for 500 features, to 84.86% for 25646 features. For Gaussian kernel and Binary data representation the decrease is more significantly (from 86.63% for 500 features to 68.91% for 25646 features). For Web documents Gaussian kernel obtains a maximum of 87.70% comparatively with polynomial kernel that obtains only 87.01% but at average the polynomial kernel obtains better results. The maximum value is obtained for a small number of features (500) meaning 2% of total number of features and for 1000 features meaning 4% of total number of features.

The author demonstrated in **Chapter 5** that the proposed methods for correlating the kernel's parameters assure that all the time the best obtained values are simple to found. The author presents also a bi-dimensional intuitive visualization of the results obtained from the classifier in order to have a simple modality to view and analyze the classification process. In order to verify the implemented algorithm, the author presents a comparison between the implemented algorithm (SVM) and another implementation of the algorithm, frequently used in literature, called LibSVM. At the end of this chapter are presented some results obtained using a developed implemented clustering algorithm.

In the polynomial kernel case there are more values for which the best results are obtained but the author's proposed formula assures to hit in almost all cases the best value without the need of making more tests to find the optimal parameter for the bias. Thus the author propose that the bias of the kernel to be correlated with the degree of the kernel ($b=2*d$).

For Gaussian kernel the author proposed a formula that always assures the best results. The author also shows that in text classification problems it is not a good idea to use C parameter equal to the

number of features, as it is frequently used in the literature. Using his proposed correlation formula the author obtained at average results with 3% better for polynomial kernel and results with 15% better for Gaussian kernel. As far as he knows, the author is the first one that proposed a correlation between these two parameters for both Polynomial and Gaussian kernels.

Using the idea to modify the Gaussian kernel the results obtained using LibSvm with kernel's parameters correlation are better in comparison with results obtained using LibSvm with standard kernel (average accuracy classification gain of 24.26% for polynomial kernel, respectively 28.44% for Gaussian kernel). The author's "UseSvm" program obtains far better results than the well-known LibSvm (with an average gain of 25.57%). For the default parameter of LibSvm the author's application has also obtained better results (76.88% in comparison with 69.97% for LibSvm).

Chapter 6 ends the flowchart of automatically documents classification with a more powerful method based on multiple basic classifiers (hybrid classification). In this chapter the author designs two adaptive meta-classifiers based on SVM classifiers. So, in this chapter, three approaches to build an efficient meta-classifier are investigated. Based on his previous work 8 different SVM classifiers are carefully selected. For each of the classifiers, the kernel, the degree of the kernel and input data representation are modified. Based on these selected classifiers the upper limit of his meta-classifier is 94.21%. The author compares here one simple static method based on Majority Vote with two original adaptive methods.

With Majority Vote the classification accuracy is 86.38%. Obviously, the documents that are correctly classified by only one classifier can't be correctly classified through this method. This is why the results are considered poor.

The meta-classifier based on Euclidean distance (SBED) method obtains the best results, growing up to 92.04% after 14 learning steps. This value is with 2.17% smaller than the upper limit the meta-classifier can reach. This method is also the fastest because it chooses the first acceptable classifier that might be used. The last developed meta-classifier based on cosine (BC-SBCOS) tries to be the most rigorous because it finds the best component classifier. As a consequence, the training time for BC-SBCOS is greater on average with 20 minutes comparatively with SBED that gives the response in only 22 minutes.

Because in the real life when working with documents we need to work with huge sets of documents in **Chapter 7** the author develops a working strategy for large documents' sets. This strategy doesn't increase exponentially the training time and doesn't loose substantially in classification accuracy. In order to do this, a method to reduce in the first step the number of input vectors from the input set and make two learning steps in order to consider the learning step finished, is proposed and implemented. The author notices that the classification accuracy decreases at average with only 1% when the dataset is reduce at 60% (from the entire dataset).

At the end of chapter 7 there are presented some experiments showing that the presented work is not significantly influenced by the selected training and testing sets. So that the author selects other training and testing sets and repeats the tests. Only the results obtained using SVM_FS method are presented it. Can be observed that the results obtained with the new grouping of data in sets are quite equivalent with the results obtained with the first grouping of data in sets. Sometimes the first sets obtain results with 1% better; sometimes the new sets are obtaining better results. The difference between these sets it is never greater than 2%. As a conclusion the results presented in the authors' entire work are not influenced on the selected sets. With other data sets the presented features selection methods and the presented classifier algorithm obtain comparable results. This conclusion encourages the author to use further his implemented classifier in other interesting domains, like Web documents classification.

8.2 General Conclusions and Further Work

This PhD thesis, with the original contributions already presented, is useful in case we want to develop a system based on automatically Web classification, especially automatically reordering Web pages returned by classical search engines. Thus, reordering and grouping can be done using the content of the pages.

Also this thesis presents some solutions for selecting different kernels and data representation. Thus, when we work with text data it is indicated to use polynomial kernel with nominal data representation and a small kernel degree. If acceptable results are not obtained because the data are strongly overlapped the Gaussian kernel with Cornell Smart data representation can be used. In order to obtain better results faster it is indicated to use a kernel with correlated parameters, as the author proved.

It is recommendable to use more classifiers combined into an adaptive meta-classification method for increasing the results without increasing the time too much. An interesting idea that can be further address is to use the parallel computation on multiprocessor computing systems in order to reduce the classification time in the meta-classification context.

Also the thesis presents methods that make the classification algorithm to work faster with larger dimensions of the data set, without decreasing the classification accuracy.

In further experiments I will try to combine the classification methods with clustering methods in order to use labeled and unlabeled data into a hybrid classification algorithm. My idea is to change the primal step from clustering, when is chosen a percentage of the Lagrange multipliers α_i (initial hyperplane) which will be initialized with a value different from 0. In this step a small number of labeled data are presented to a classification algorithm in order to obtain the α_i coefficients that are used as initial values for clustering process. This showed offer 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 dealing 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 an empty set). There are definitely 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 because the training set can not contain the roots of all existing words. Feature selection methods choose only features that are relevant for the training set. As we have seen in this PhD thesis the algorithm obtains better results when it uses fewer but relevant features. Thus training with fewer features increases the number of documents that can not be classified after the learning step. As a further improvement I will try to develop tests with families of words and use as features only a representative of each family. This way 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 can increase the classification accuracy when it is used as a feature selection method. In order to achieve this we could use the [WordNet] database which contains a part of the families of words for the English language. Also, to increase the number of documents that can be further classified, methods based on synonymy and polysemy problems can be approached. Thus, I will keep only one word from more synonyms reducing the number of features and increasing the number of documents that can be used further. Also, detecting the sense of the word based on the context can avoid the classification of different documents in the same category. WordNet database can provide this, but right now only for a small number of words.

"Research is what I'm doing when I don't know what I'm doing."

Wernher von Braun

9 Glossary

I do not pretend to be extremely rigorous in the following definitions. The terms are defined in order to express, as good as possible, the concrete context in that they occur in my PhD thesis.

Aggregation – a method of reducing a database dimension by union general identical tuples.

Classification accuracy - the ability to predict the correct class labels, percentage of correct classified documents from all documents.

Clustering – an the unsupervised learning process for grouping the data into classes (or clusters) so that objects within a class have high similarity, but are very dissimilar in comparison with the objects from other classes.

Data cube – consists of lattice of cuboids each corresponding to a different degree of summarization of the given multidimensional data.

Data mining – the process of extracting knowledge from vast amounts of data.

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 document.

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; represents a set of points from the training set that meets a constraint expressed as a linear equation.

Information retrieval – the process concerned with the organization and retrieval of information from large number of text based documents.

Interested – a measurement of the request to produce description passes.

Kernel – a function that computes the dot product in the feature space directly as a function of the original input points.

Kernel trick – given an algorithm which is formulated in terms of a positive definite kernel k, one can construct an alternative algorithm by replacing k by another positive definite kernel $\bar{k}$.

Lattice of cuboids – represent a part from a data cub.

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

Meta-classification – a technique for combining the predictions obtained from the base-level models in order to improve the classification accuracy (used in hybrid classifier).

Normalizing - a technique for representing the data in a new, usually small, common value domain.

Noise in data – a random error or variance in a specific stochastic variable.

Ontology - a document or file of vocabularies that formally define the relations among terms based on own taxonomy.

Outlier - considered values out of range or noise in data.

Optimal hyperplane – the hyperplane distinguished by the maximum margin of separation between any training point and the hyperplane.

Positive defined kernel – into a nonempty set $\mathbf{X}$ is a function k on $\mathbf{X} \times \mathbf{X}$ which for all $m \in \mathbf{N}$ and all $x_1,...,x_m \in \mathbf{X}$ gives gain to a positive defined Gram matrix.

Relevant – in context represents a term (feature) that has a good influence in obtaining better classification results.

Semantic Web – the extension of current Web that allows to find, share, and combine information more easily; machine understandable contents.

System scalability – a system is considered scalable if when enlarging the number of input data *n* times, it doesn't take more than *n* training times to execute the learning process.

Stemming – the process of extracting the root of a word; in this case only for English language.

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

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

Support Vector Machine Method – a set of related supervised learning methods based on kernels used for classification and regression.

Supervised learning – in the learning process for each sample the class label the sample belongs to is presented.

Tree widget – a form of hierarchically organize obtained results.

Unsupervised learning – in the learning process, the training set is made only with inputs vectors without specifying the output (class label) of the input vectors.

Web Mining – the process of application of data mining techniques to discover patterns from the Web.

Web directory – a directory on the World Wide Web. It specializes in linking to other web sites and categorizing those links.

"The secret to creativity is knowing how to hide your sources."

Albert Einstein

10 References

[Ack97_1] Ackerman, M., *The DO-I-Care Agent: Effective Social Discovery and Filtering on the Web*, Proceedings of RIAO'97: Computer-Assisted Information Searching on the Internet, pages 17-31, 1997.

[Ack97_2] Ackerman, M., Starr, B., Pazzani, M., *DO I Care? – Tell Me What's Change on the Web*, In Proceedings of the American Association for Artificial Intelligence Spring Symposium on Machine Learning, 1997.

[Ack97_3] Ackerman, M, Billsus, D., Gaffney, S., Hettich, S., Khoo, G., Kim, D., Klefstad, R., Lowe, C.,. Ludeman, A., Muramatsu, J., Omori, K., Pazzani, M., Semler, D., Starr, B., Yap, P., *Learning Probabilistic User Profiles: Applications to Filtering Interesting Web Sites, Notifying Users of Relevant Changes to Web Pages, and Locating Grant Opportunities*, AI Magazine 18(2) pages 47-56, 1997.

[Alb04] Albanese, M., Picariello, A., Sansone, C., Sansone, L., *A Web Personalization System based on Web Usage Mining Techniques*, In Proceedings of the World Wide Web Conference 2004, New York pp. 288-289, 2004.

[And04] Andronico, P., Buzzi, M., Leporini, B., *Can I Find What I'm Looking For?*, In Proceedings of the World Wide Web Conference 2004, New York, pp. 430-431, 2004.

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

[Bar02] Barbat, B., *Agent oriented intelligent systems*, Romania Academy Publishing House, Bucharest, 467 pages, 2002 (in Romanian).

[Bar03] Barbu, C., Marin, S., *Information Filtering Using the Dynamics of the User Profile*, In Proceedings of The 16th International FLAIRS Conference, 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.

[Ber99] Berners-Lee, T., *Weaving the Web*, Orion Business Book, 1999.

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

[Bha00] Bhatia, S., *Selection of Search Terms Based on User Profile*, ACM Explorations, pages. 224-233, 1998.

[Bra94] 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.

[Bos92] Boser, B. E., Guyon, I. M., Vapnik, V., *A training algorithm for optimal margin classifiers*, in proceedings of the 5th Annual ACM Workshop on Computational Learning Theory, pages 144-152, Pittsburgh ACM Pres, 1992

[Cha00] Chakrabarti, S., *Data mining for hypertext: A tutorial survey*, Newsletter of the Special Interest Group (SIG) on Knowledge Discovery & Data Mining, ACM Explorations 1(2), pages. 1-11,2000.

[Cha03] Chakrabarti, S., *Mining the Web. Discovering Knowledge from Hypertext Data*, Morgan Kaufmann Publishers, USA, 2003.

[Cha05] Charlesworth, I., *Integration fundamentals*, Ovum, 2005.

[Che98] Chen, L., Sycara, K., *WebMate: A Personal Agent for Browsing and Searching*, Proceedings of the 2nd International Conference on Autonomous Agents, pages 132-139, 1998.

[Che00] Chen, H., Dumais, S., – *Bringing Order to the Web: Automatically Categorizing Search Results*, In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, pages 145-152, 2000.

[Chi03] Chih-Wei, H., Chih-Chang, C., Chih-Jen, L., *A Practical Guide to Support Vector Classification*, Department of Computer Science and Information Engineering National Taiwan University, 2003 (Available at http://www.csie.ntu.edu.tw/~cjlin/papers/ guide/ guide).

[Chr02] Christoph, K., Veit, B., *Visual Representation and Contextualization of Search Results*, *List and Matrix Browser*, In Proceedings of International Conference on Dublin Core and Metadata for e-Communities, pp. 229-234, 2002.

[Coo99] Cooley, R., Mobasher, B., Srivastava, J., *Data preparation for mining World Wide Web browsing patterns*, Journal of Knowledge and Information Systems 1(1), pages 5-32, 1999.

[Cov91] Cover, T. M.,. Joy, T. A., *Elements of Information Theory*, Jhon Wiley & Sons Interscience Publication, 1991.

[Cro99] *Introduction to Data Mining and Knowledge Discovery*, Tow Crows Corporation, a short introduction to a new book ,1999 (Available at http://www.twocrows.com/booklet.htm).

[Dav06] Davies J., Studer R., Warren P., *Semantic Web Technologies Trends and Research in Ontology-based Systems*, John Wiley & Sons, Ltd, 2006.

[Dee90] Deerwester, S., Dumais, S., Furnas, G., Landauer, T., Harshman, R., *Indexing by latent semantic analysis*, Journal of the American Society for Information Science, 41, pages. 391-407, 1990.

[Der00] Dertouzos, M., *What will be: How the New World of Information Will Change Our Lives*, Technical Publisher, Bucharest, 2000 (Translation in Romanian by F. G. Filip et al.).

[Die95] Dietterich, T. G., Bakiri, G., *Solving multi-class learning problems via error-correcting output codes*, Journal of Artificial Intelligence Research, 1995

[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.

[Dou04] Douglas, H., Tsamardinos, I., Aliferis C., *A Theoretical Characterization of Linear SVM-Based Feature Selection*, Proceedings of the 21st International Conference on Machine Learning, Banff, Canada,2004.

[Dum00] Dumais, S., Hao Chen, *Hierarchical Classification of Web Content*, Proceedings of the 23rd international ACM SIGIR Conference on Research and Development in Information Retrieval, pages 256-263, 2000.

[Fen04] Fensel, D., *Ontologies: A Silver Bullet for Knowledge Management and Electronic Commerce*, Second Edition, Springer –Verlag Berlin Heidelberg, 2004.

[For04] Forman, George, *A Pitfall and Solution in Multi-Class Feature Selection for Text Classification,* Proceedings of the 21st International Conference on Machine Learning, Banff, Canada, 2004.

[Gab04] Gabrilovich, E., Markovitch S., *Text Categorization with Many Redundant Features Using Aggressive Feature Selection to Make SVM Competitive with C4.5*, Proceedings of the 21st International Conference on Machine Learning, Banff, Canada,2004.

[Geo99] Goecks, J., Shavilk, J., *Automatically Labeling Web Pages Based on Normal User Action*, Proceedings of the International Joint Conference on Artificial Intelligence. Workshop on Machine Learning for Information Filtering, pages 573-580, 1999.

[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.

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

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

[Gru93] Gruber, T., *A Translation approach to portable ontologies*. Knowledge Acquisition 5, pages 199-220, 1993.

[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.

[Hof99] Hofmann, T., *Probabilistic Latent Semantic Analysis*, In Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence, Stockholm, pages. 289-296, UAI 1999.

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

[Ian00] Ian H., Witten, E. F., *Data Mining, Practical Machine Learning Tools and Techniques with Java implementation*, Morgan Kaufmann Press, 2000.

[Jai00] Jaideep, S., Cooley, R., Mukund, D., Pang Ning Tan, *Web - Usage Mining: Discovery and Applications of Usage Patterns from Web Data*, Technical Report, Department of Computer Science and Engineering University of Minnesota, 2000 (Available at http://maya.cs.depaul.edu/~classes/ect584/papers/srivastava.pdf).

[Jai01] Jaiwei, H., Micheline K., *Data Mining. Concepts and techniques*, Morgan Kaufmann Press, 2001.

[Jeb00] Jebara, T., 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.

[Jim04] Romng, J., Huan, L., *Robust Feature Induction for Support Vector Machine*, Proceedings of the 21st International Conference on Machine Learning, Banff, Canada, 2004.

[Joa99] Joachims, T., *Making large-scale support vector machine learning practical*, In B. Scholkopf, C. J. C. Burges, A.J. Smola (Eds): *Advances in kernel methods: Support vector learning*, MIT Press, 1999, pp. 169-184.

[Jur03] Jurafsky, D, Pradhan, S, Ward, W., Hacioglu, K., Martin, J., – *Shallow Semantic Parsing using Support Vector Machines*, Proceedings of the Human Technology Conference / North America chapter of the Association of Computational Linguistics, Boston, 2003.

[Kai02] Kai, Yu, Schwaighofer, A., Volker, T., Wei-Ying, M., HongJing, Z., *Collaborative Ensemble Learning: Combining Collaborative and Content Based Information Filtering,* Technical Report Siemens AC CT IC, Munich, 2002.

[Kai03] Kai, Yu, Schwaighofer, A., Volker, T., Wei-Ying M., HongJing Z., C*ollaborative Ensemble Learning: Combining Collaborative and Content Based Information Filtering via Hierarchical Bayes*, Proceeding of the 19th Conference, Uncertainty in Artificial Intelligence, pages 616-623, 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.

[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.

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

[Koh97] Kohonen, T., *Self-Organizing Maps*, Second edition, Springer Publishers, 1997.

[Kum01] Kummamuru, K., Krishnapuram, *A clustering algorithm for asymmetrically related data with its applications to text mining*, In Proceedings of CIKM, pages 571-573, 2001.

[Kum04] Kummamuru, K., Lotilikar, R., Roy, S., Singal, K., Krishnapuram, R., *A Hierarchical Monothetic Document Clustering Algorithm for Sumarization and Browsing Search Results*, In Proceedings of the Thirteenth International World Wide Web Conference, pages 658-665, 2004.

[Law99] Lawrence, S., Giles, C. L., *Accessibility of information on the Web*,Nature,400, pages 107-109, 1999.

[Law01] Lawrie, D., Croft, W. B., Rosenberg, A., *Finding topic words for hierarchical summarization*, In proceedings of SIGIR, pages 349-357, 2001.

[Lin02_1] Lin, W.-H., Houptmann, A., *News Video Classification Using SVM-based Multimodal Classifier and Combination Strategies*, International Workshop on Knowledge Discovery in Multimedia and Complex Data, Taiwan, 2002.

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

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

[Lym03] Lyman, P., *How Much Information?* School of Information Management and System, University of California at Berkeley, http://www.sims.berkeley.edu/research/projects/how-much-info-2003.

[Mit97] Mitchell, T., *Machine Learning*, McGraw Hill Publishers, 1997.

[Mla98] Mladenic, D., *Feature Subset Selection in Text Learning*, Proceedings of the 10th European Conference on Machine Leraning (ECML-98), pages 95-100, 1998.

[Mla99] Mladenic, D., Grobelnik, M. *Feature selection for unbalanced class distribution and naïve bayes*, In Proceedings of the 16th International Conference on Machine Learning ICML, p.258-267,1999.

[Mla04] Mladenic, D., Brank, J., Grobelnik, M., Milic-Frayling, N., *Feature Selection Using Support Vector Machines* The 27th Annual International ACM SIGIR Conference (SIGIR2004), pp 234-241, 2004.

[Mob96] Mobasher, B., Jain, N., Han, E.-H., Strivastava, J., *Web Mining: Pattern Discovery from World Wide Web Transactions*, Technical Report, 1996.

[Mor03_1] **Morariu, D.**, Curea, G., *Learning using the Support Vector Concept*, Proceedings of Scientific Workshop „The Science Challenge in XXI Century", organized by Land Forces Military Academy Sibiu, ISBN 973-8088-85-2, pages 29-36, Sibiu, December 2003.

[Mor03_2] Curea, G., **Morariu D**., *Structure of Web data using XML*, Proceedings of Scientific Workshop „The Science Challenge in XXI Century", organized by Land Forces Military Academy Sibiu, ISBN 973-8088-85-2, pages 21-28, Sibiu, December 2003.

[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.

[Mor05_2] **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

[Mor06_1] **Morariu, D.**, Vintan, L., *A Better Correlation of the SVM Kernel's Parameters*, Proceedings of the 5th RoEduNet IEEE International Conference, ISBN (13) 978-973-739-277-0, Sibiu, June, 2006.

[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 (ICIS'06), ISSN 1503-5313, vol. 14, pp. 83-89, Prague, August, 2006.

[Mor06_3] **Morariu, D.**, Vintan, L., Tresp, V., *Evolutionary Feature Selection for Text Documents using the SVM*, Proceedings of the 3rd International Conference on Machine Learning and Pattern Recognition (MLPR'06), ISSN 1503-5313, vol.15, pp. 215-221, Barcelona, Spain, October, 2006.

[Mor06_4] **Morariu, D.**, Vintan, L., Tresp, V., *Meta-classification using SVM classifier for Text Document*, Proceedings of the 3rd International Conference on Machine Learning and Pattern Recognition (MLPR'06), ISSN 1503-5313, vol. 15, pp. 222-227, Barcelona, Spain, October, 2006.

[Mor06_5] **Morariu, D.**, *Relevant Characteristics Extraction*, 3rd PhD Report, University „Lucian Blaga" of Sibiu, October, 2006, http://webspace.ulbsibiu.ro/daniel.morariu /html/Docs/Report3.pdf

[Mor06_6] **Morariu, D.**, Vintan, L., Tresp, V., *Evaluating some Feature Selection Methods for an Improved SVM Classifier*, International Journal of Intelligent Technology, Volume 1, no. 4, ISSN 1305-6417, pages 288-298, December 2006

[Mor07_1] **Morariu, D.**, Vintan, L., *Kernel's Correlation for Improving SVM in Text Documents' Classification*, (Accepted) "Acta Universitatis Cibiniensis", Technical Series, "Lucian Blaga" University of Sibiu, 2007

[More05] Morello, D., *The human impact of business IT: How to Avoid Diminishing Returns*, published in Information Age magazine, Australian Computer Society, 2005.

[Nel00] Nello C., Shawe-Taylor, J., *An Introduction to Support Vector Machines and other kernel-based learning methods*, Cambridge University Press, 2000.

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

[Pla99_1] 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.

[Pla99_2] Platt J., *Probabilistic Outputs for Support Vector Machines and Comparisons to Regularized Likelihood Methods*, In Advances in Large Margin Classifiers, A. Smola, P. Bartlett, B. Scholkopf, D. Schuurmans, eds., MIT Press, Cambridge, pages 61-74, 1999.

[Sch02] Schoslkopf Bernhard, Smola Alexander, *Learning with Kernels, Support Vector Machine*, MIT Press, London, 2002.

[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.

[Str00] Strivastava, J., Cooley R., Deshpande M., Tan Pang-Ning, *Web Usage Mining: Discovery and Applications of Usage Patterns from Web Data*, ACM SIGKDD Explorations, pages 12-23, 2000.

[Ray00] Raymond, K., Hendrik, B., *Web mining research: A survey*, In SIGKDD Explorations: Newsletter of the Special Interest Group (SIG) on Knowledge Discovery & Data Mining, ACM Press, pages 1-15,2000.

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

[Vap01] Vapnik, V., Asa, Ben-Hur, Horn, D., Sieglmann H. T., *Support Vector Clustering*, Journal of Machine Learning Research 2, pages 125-137, 2001.

[Yan97] Yang, Y., Pedersan, J.O., *A Comparative Study on Feature Selection in Text Categorization*, Proceedings of ICML, 14th International Conference of Machine Learning, pages 412-420, 1997.

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

[Wah99] Wahba, G., *Support Vector Machine, reproducing kernel Hilbert space and the randomized GACV*. In B. Scholkopf, C. J. C. Burgers, and A. J. Smol;a, editors, Advances in Kernel Methods – Support Vector Learning, pages 69-88, Cambridge, MA, 1999, MIT Press.

[Whi94] Whitely, D., *A genetic Algorithm Tutorial, Foundations of Genetic Algorithms*, Morgan Kaufmann Publishers, 1994.

[Wiz04] Wojciech, W., Krzysztof, W., Wolciech, C., *Periscope – A System for Adaptive 3D Visualization of Search Results*, Association for Computing Machinery , p.29-40, 2004.

Web References

[Aol] search.aol.com (Web directories, accessed in December 2006).

[Dmoz] www.dmoz.org (Open Directory Project - Web directories, accessed in December 2006).

[Excite] www.excite.com (Web directories, accessed in December 2006).

[Google] www.google.com (search engine, accessed in December 2006).

[Ir] http://www.cs.utexas.edu/users/mooney/ir-course/ Information Retrieval java Application, accessed in December 2006.

[kartoo] www.kartoo.com (graphical representation of search results and monitoring a specific site, accessed in December 2006).

[LibSvm] http://www.csie.ntu.edu.tw/~cjlin/libsvm - accessed in December 2006.

[LookSmart] http://www.looksmart.com/ (Web directories with preclassified Web pages) – accessed in December 2006

[SurfMind] www.surfmind.com/Web (Web directories and searching into a specific domain, accessed in November 2006).

[Surfwax] www.surfwax.com (help user to find different synonyms grouped after domain for every search word, accessed in December 2006).

[SparseMatrix] www.cs.utk.edu/~dongarra/etemplates/node372.html - presenting an optimal modality to store sparse matrix.

[Periscope] http://periscope.kti.ae.poznan.pl/ (graphical representation of the search results) – accessed in December 2006.

[Reu00] Misha Wolf and Charles Wicksteed- Reuters Corpus: http://www.reuters.com/researchandstandards/corpus/ Released in November 2000 accessed in June 2005

[Rdf] www.w3.org/rdf.

[Yahoo] www.yahoo.com (search engine and Web directory, accessed in December 2006).

[Vivisimo] www.vivisimo.com (2 levels hierarchical representation of the search results, accessed in December 2006).

[WebCr] www.Webcrawler.com (one level hierarchical representation of the search results, accessed in December 2006).

[WebMate] www.cs.cmu.edu/~softagents/Webmate (the proactive agent that learn the user profile to increase the quality of the search results) – accessed in December 2006.

[Weka] www.cs.waikato.ac.nz/~ml/weka/ - accessed in December 2006.

[WordNet] http:\\www.cogsci.princeton.edu\obtain – online lexical system – accessed in December 2006.