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

Contributions to Automatic Knowledge Extraction from Unstructured Data

PhD Thesis

(abstract)

Author:
Ionel Daniel MORARIU, MSc

PhD Supervisor:
Professor Lucian N. VINȚAN, PhD

SIBIU, 2007

Contents

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**.

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

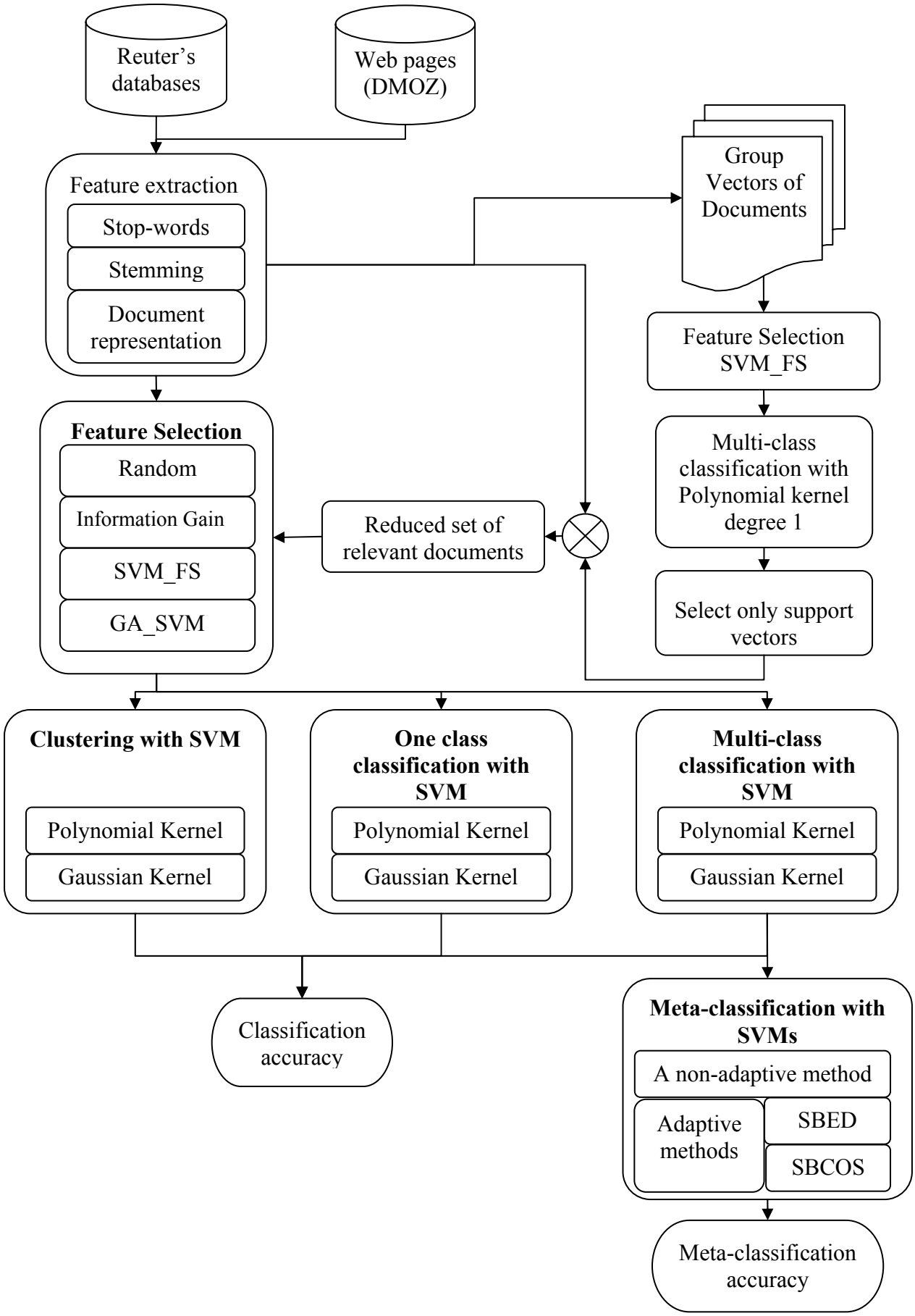

Figure 1.1 – Documents classification flowchart. My view

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.

2 Methods for Knowledge Discovery in Data

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.

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. 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. The process of knowledge discovery in database has more steps as preprocessing data, data mining, patter evaluation and 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, data integration, data selection, data transformation and data reduction.

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.

Data mining can be classified into *descriptive data mining* and *predictive data mining*. In the data mining step the users have only a rough idea of what the interesting attributes for exploration might be.

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. For mining association rules there are some classical algorithms like Apriori, Frequent Pattern Growth, Multilevel Association Rules and Constraint Based Rule Mining.

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.

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.

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.

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. In this step the documents that are always in an understandable format for the user are transformed into a format that is more accessible for that computer and after that are used some technique for analyzing and extract relevant information.

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

There are some methods for measure the relevance degree in a information retrieval system. 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: as *Precision* that represents the percent of documents that are [Relevant] and [Retrieved] from

[Retrieved] documents and *Recall* that represents the percent of documents that are [relevant] and [Retrieved] from al [Relevant] documents

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".

The information retrieval systems 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 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. This represents the methods used in this thesis.

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.

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.

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.

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.

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.

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.

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.

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.

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 because: the users are unable to formulate the right query and restrict the result set, the user interface is difficult to be used, the ranking function is applied statically and because the information on the Internet is rarely structured and organized for fast retrieval.

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.

To solve the problems presented earlier this research field grown rapidly in areas such as algorithms, strategies and architecture. Hundreds of ideas were tried, some were implemented and some are only prototypes. Some of those ideas are: organizing documents in predefined categories, improve the way of presenting results, monitoring a specified pages, help the user to formulate correct query and programs that filter the search results.

Organizing documents in predetermined categories, usually named *web directories* have a static structure that is very difficult to be updated. 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.

An 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. 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. Organizing web search results into a hierarchy of topics and subtopics facilitates browsing the collection and locating more easily the interesting results.

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.

Single keywords are usually ambiguous, or to 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.

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 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.

Ontology represents the heart of all semantic web applications. A commonly agree definition of ontology is: "ontology is an explicit and formal specification of a conceptualization of a domain of interest". Ontologies are used to organize knowledge into a structure way in many areas – from philosophy to knowledge management and the Semantic Web.

2.4 The Data Sets Used

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. 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 algorithm.

In the extraction phase from 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.

For each word that remained after the elimination of stopwords I have extracted the root of the word (stemming) and counted its occurrences. Thus I have created an vector for each document from Reuters with the remaining roots (called later features). This vector is the individual characterization of each document and 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 Reuter's classification topics.

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). 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.

2.4.1 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

2.4.2 Choosing a Larger Dataset

In this thesis I make an expectation when use larger dataset. In what follows I am presenting the method of obtaining this data set. Thus form 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 extract a number of 28624 features after eliminating the stopwords and extracting the root of the word (steaming). 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.

2.4.3 Preprocessing the Used Web Dataset

In this paragraph I want to preset the method used for build the database used for web mining. I am interested in this moment only in web content mining without being interested in the thesis 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 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 subcategory. 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. A number of 745 html files were selected from those categories and were extracted 26998 roots of words. The selected documents were split randomly in a training set that contain 445 documents and a testing set of 377 documents.

2.4.4 Type of Data Representation

Because there are many ways to define the term-weight (features), I represent 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:

$$TF(d,t) = \frac{n(d,t)}{\max_{\tau} n(d,\tau)} \tag{2.1}$$

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 the most 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:

$$TF(d,t) = \begin{cases} 0 & if\ n(d,t) = 0 \\ 1 + \log(1 + \log(n(d,t)) & otherwise \end{cases} \tag{2.2}$$

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).

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.

3.1 SVM Technique for Binary Classification

Support Vector Machine (SVM) is a classification technique based on statistical learning theory [Sch02], [Nel00] that was applied with great success in many challenging non-linear classification problems and was successfully applied to large data sets.

The SVM algorithm finds a hyperplane that optimally splits the training set. The optimal hyperplane can be distinguished by the maximum margin of separation between all training points and the hyperplane (a practical idea about implementation of this algorithm can be found in [Chi03]). Looking at a two-dimensional problem we actually want to find a line that "best" separates points in the positive class from points in the negative class. The hyperplane is characterized by a decision function like:

$$f(x) = \mathrm{sgn}\left(\langle \mathbf{w}, \mathbf{x} \rangle + b\right) \tag{3.1}$$

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, "*Φ(x)*" is a function that transforms the input data into a higher dimensional feature space and $\langle \cdot, \cdot \rangle$ representing the dot product. *Sgn* is the signum function that returns 1 if the value is greater or equal to 0 and -1 otherwise. If **w** has unit length, then $<\mathbf{w}, \Phi(x)>$ is the length of $\Phi(x)$ along the direction of **w**. Generally **w** will be scaled by $\|\mathbf{w}\|$. The training part the algorithm needs to find the normal vector "**w**" that leads to the largest "*b*" of the hyperplane.

The problem seems very easy to be solved 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 one. This learning algorithm can be performed in a dot product space and for data which is linear separable, 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.

For training data which is not separable by a hyperplane in the input space the idea of SVM is to map the training data into a *higher-dimensional feature space* via Φ, and construct a separating hyperplane with the maximum margin there. This yields a non-linear decision boundary into the input space. By the use of a kernel function $\langle \mathbf{w}, \phi(x) \rangle$ it is possible to compute the separating hyperplane without explicitly carrying out the map into the feature space [21].

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

$$\begin{aligned} &\underset{\mathbf{w}\in H, b\in\Re}{\text{minimize}}\ \tau(\mathbf{w}) = \frac{1}{2}\|\mathbf{w}\|^2 \\ &\textit{subject to } y_i\left(\langle \mathbf{w}, \mathbf{x}_i\rangle + b\right) \geq 1 \textit{ for all } i = 1,\ldots,m \end{aligned} \tag{3.2}$$

The constraints ensure that $f(x_i)$ will be +1 for y_i=+1 and -1 for y_i=-1. This problem is computationally attractive because it can be constructed by solving a quadratic programming problem for which efficient algorithms already exist. Function τ is called objective function with inequality constrain. Together, they form a so-called *primal optimization problem*. To solve this type of problems it is more convenient to deal with the dual problem by introducing the Lagrange multipliers $\alpha_i \geq 0$ and the Lagrangian [21] which lead to the so-called *dual optimization problem*:

$$L(\mathbf{w}, b, \alpha) = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{m} \alpha_i (y_i (\langle \mathbf{x_i}, w\rangle + b) - 1) \tag{3.3}$$

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. This leads to:

$$\begin{aligned} &\mathbf{w} = \sum_{i=0}^{m} \alpha_i y_i x_i \\ &\sum_{i=0}^{m} \alpha_i y_i = 0 \end{aligned} \tag{3.4}$$

The solution vector thus has an expansion in terms of training examples. 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 that are precisely met. Formally, for all i=1,…,m, it can be written:

$$\alpha_i \left[y_i \left(\langle \mathbf{w}, \mathbf{x}_i \rangle + b \right) - 1 \right] = 0 \textit{ for all } i = 1,\ldots,m \tag{3.5}$$

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, witch is the problem that one usually solves in practice.

$$\underset{\alpha\in\Re^m}{\text{maximize}}\, W(\alpha) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} \alpha_i \alpha_j y_i y_j \langle x_i, x_j \rangle \tag{3.6}$$

this is called the target function

$$\text{subject to } \alpha_i \geq 0 \textit{ for all } i = 1,\ldots,m \textit{ and } \sum_{i=1}^{m} \alpha_i y_i = 0 \tag{3.7}$$

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

$$f(x) = \text{sgn}\left(\sum_{i=1}^{m} y_i \alpha_i \langle x_i, x \rangle + b \right) \tag{3.8}$$

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.

Everything was formulated in a dot product space. On the practical level, changes have to be made to perform the algorithm in a higher-dimensional feature space. Thus the new patterns $\Phi(x_i)$ can equally well be the result of mapping the original input patterns x_i into a higher dimensional feature space using function Φ. Maximizing the target function and evaluating the decision function involve the computation of dot products $\langle \phi(x), \phi(x) \rangle$ in a higher dimensional 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* is used to extend 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* as the following form:

$$f(x) = \operatorname{sgn}\left(\sum_{i=1}^{m} y_i \alpha_i k(x, x_i) + b \right) \tag{3.9}$$

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, 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

$$\arg\max_{j=1,..,M} g^j(x), \quad \textit{where } g^j(x) = \sum_{i=1}^{m} y_i \alpha_i^j k(x, x_i) + b^j \tag{3.10}$$

and

$$f^j(x) = \operatorname{sgn}(g^j(x)) \tag{3.11}$$

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

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] modification 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 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:

$$\Phi(x, x') = e^{-\frac{\|x-x'\|^2}{2\cdot\sigma^2}} \tag{3.12}$$

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.

3.4 SMO - Sequential Minimal Optimization

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 generate 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. 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.

3.5 Types of Kernels

In this PhD thesis I tested a new idea to correlate this 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:

$$k(x, x') = \left(2 \cdot d + \langle x \cdot x' \rangle\right)^d \tag{3.13}$$

  - *d* being the only parameter to be modified.

- Gaussian (radial basis function RBF):

$$k(x, x') = \exp\left(-\frac{\|x - x'\|^2}{n \cdot C}\right) \tag{3.14}$$

  - *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**.

3.6 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.

3.6.1 Polynomial Kernel Parameters Correlation

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

$$k(\mathbf{x}, \mathbf{x}') = \left(\langle \mathbf{x} \cdot \mathbf{x}' \rangle + b\right)^d \tag{3.15}$$

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.

3.6.2 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:

$$k(x,x') = \exp\left(-\frac{\|x-x'\|}{C}\right) \tag{3.16}$$

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.

4 Feature Selection Methods Developed

4.1 Data Reduction

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!

4.2 Random Selection (RAN)

In this feature selection method random weights between 0 and 1 are assigned to each feature. We chose this simple method just to have a base (lower limit) in evaluating the performance gains introduced by the other three methods. 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. We repeat this process for three times. After doing this we classify all of the sets and then we compute the average classification accuracy. This value will be considered the classification accuracy for random selection.

4.3 Entropy and Information Gain (IG)

Information Gain and Entropy [Jeb00], [Mit97] are functions of the probability distribution that underlie the process of communications. The entropy is a measure of uncertainty of a random variable. 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 proportion of *S* belonging to class *i*.

Based on entropy an attribute effectiveness a measure is defined in features selection. The measure is called *Information Gain*, and is the expected reduction in entropy caused by partitioning the samples according to this attribute. More precisely, the information gain of an attribute relative to a collection of samples *S*, is defined as:

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

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.

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.

Forman in [For04] reported that Information Gain failed to produce good results on an industrial text classification problem, as Reuter's database. The author attributed this to the property of many feature scoring methods to ignore or to remove features needed to discriminate difficult classes.

4.4 Feature Subset Selection Using SVM (SVM_FS)

For this method I have used my implemented SVM algorithm with linear kernel in the step of features selection. 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 3.6 and as I showed this correlation produces good results. Because I use more than two topics, for each topic a decision function 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.

4.5 Features Selection Using Genetic Algorithms (GA_FS)

Genetic algorithms encode a potential solution to a specific problem on a simple chromosome-like data structure and apply genetic operators to these structures so as to preserve critical information [Gol89], [Lug98]. In our feature selection problem the chromosome is considered to be of the following form:

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

where $w_i, i = \overline{1,n}$ represent the weight for each feature, and *b* represent the bias of the hyperplane of SVM. We consider that the training set has the form $\left\{\langle \vec{x}_i, y_i \rangle, i = 1,...,m\right\}$, where y_i represents the output of the input sample $\vec{x}_i$, and it can only take -1 and +1. We chose this form of chromosome to facilitate using of SVM for fitness function, keeping into the chromosome the parameters that are modified into SVM decision function. Thus potential solutions to the problem encode the parameters of the 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*. Following the idea proposed for multi-class classification ("one versus the rest"), I try to find the best chromosome for each of the 24 Reuters topics. For each topic we start with a generation of 100 chromosomes, each of them having values randomly generated between -1 and 1.

Using the SVM algorithm with linear kernel $\langle w,x \rangle + b$ we can compute the fitness function for each chromosome. 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 selection, crossover or mutation [Whi94]. The evolutionary process stops after a predefined number of steps are taken or when in the last 20 steps no change occurs.

At the end of the algorithm, we obtain for each topic the best chromosome that represents the decision function. We then normalize each weight vector in order to obtain all weights between 0 and 1. For selecting the best features we make an average over all those 24 obtained weight vectors and select the features according to their descending weights. The developed method is detailed in [Mor06_5].

4.6 Feature Subset Selection. A Comparative Approach

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.

4.6.1 Number of Features 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.

In Figure 4.1 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.

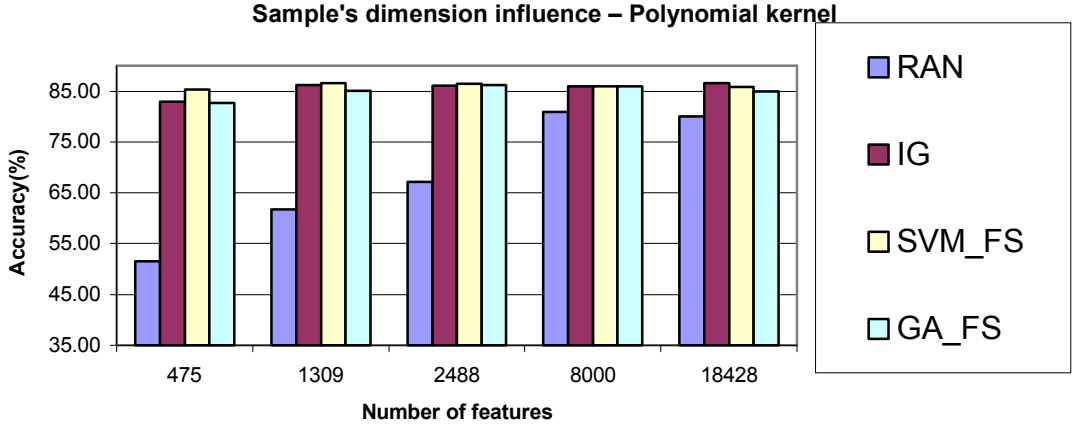

Figure 4.1 - Influence of the number of features on the classification accuracy using Polynomial kernel with degree equal to 2 and Binary data representation

The classification performance, as it can be observed from Figure 4.1 and Figure 4.2 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 and Gaussian kernels and obtains best results using a small number of features.

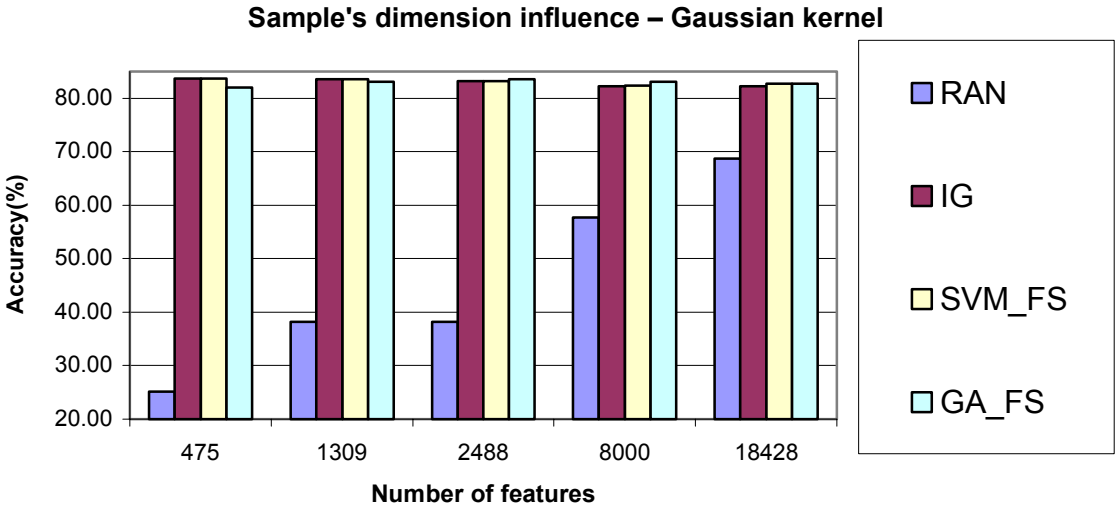

Figure 4.2 - Influence of the number of features on the classification accuracy using Gaussian kernel with parameter *C* equal to 1.3 and binary data representation

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 4.3 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 teste) 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.

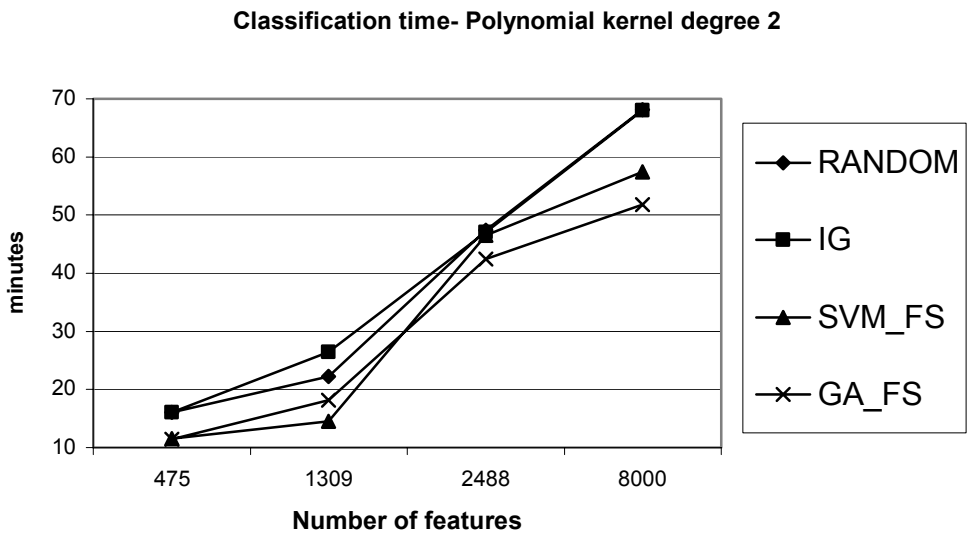

Figure 4.3 - Learning classification time function of selected numbers of features

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 result only for nominal data representation here because as it will be shown in section 5.5 it obtains at average the best results. Also Gaussian kernel with binary data representation obtains at average the best results. In Figure 4.4 are presented results obtained for a set with 475 features selected with all presented methods and polynomial kernel.

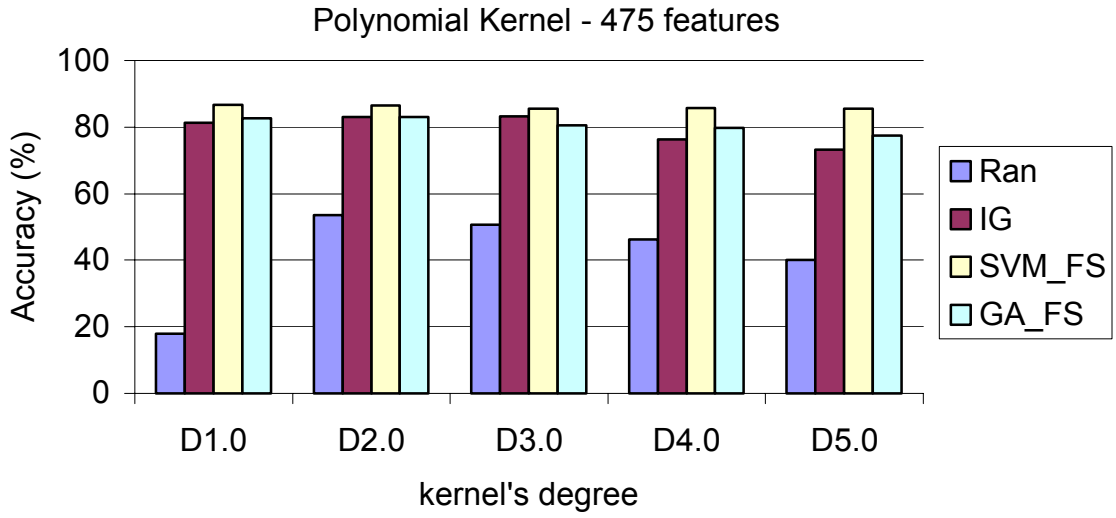

Figure 4.4 – Influence of feature selection method and kernel degree on classification accuracy

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.

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.

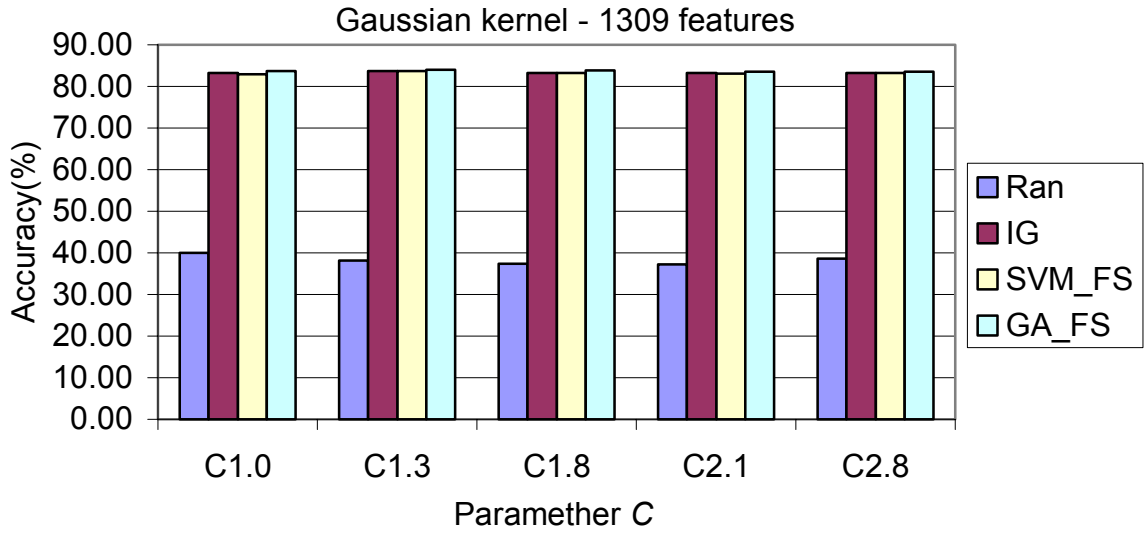

Figure 4.5 – Comparison between feature selection methods for 1309 features

When the number of features increases to 1309 (Figure 4.5) 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%.

When the number of features increases more the maximum accuracy obtained doesn't increase. 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.

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).

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 4.1 I presented an average of classification accuracy over all results obtained for each feature set dimension and 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.

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 4.1 – Averages over all tests made for each feature dimension and each kernel type

4.6.4 The Influence of Features Selection on Web Documents

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 2.4.4, 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 4.6.

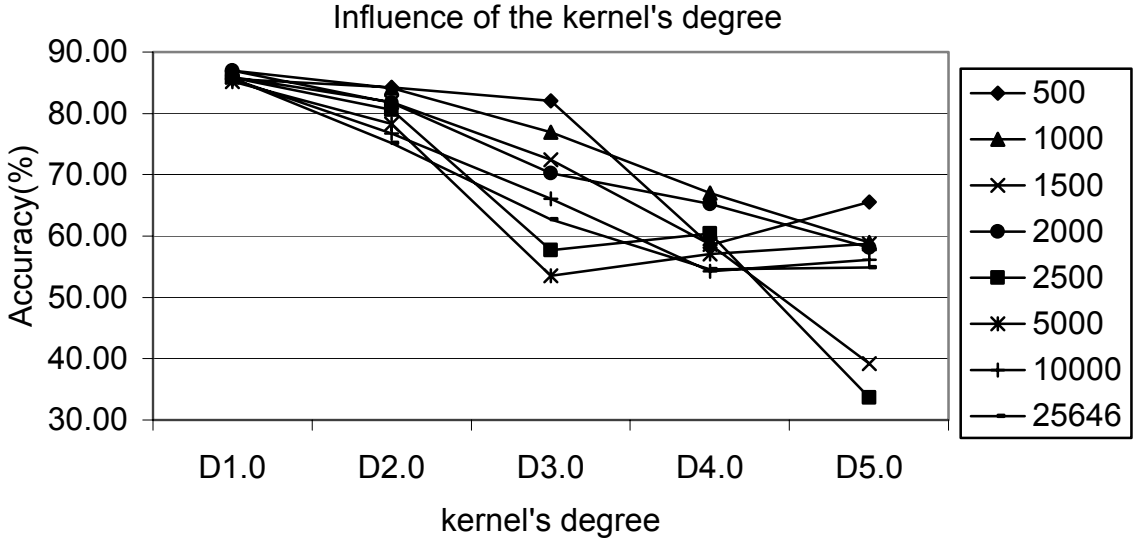

Figure 4.6 – Influence on kernel's degree for Polynomial kernel and Binary data representation

In Figure 4.6 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 linearly separable, conclusion that was also obtained for Reuters text file.

In Figure 4.7 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 decreases tendency of the classification accuracy when the number of features is increasing.

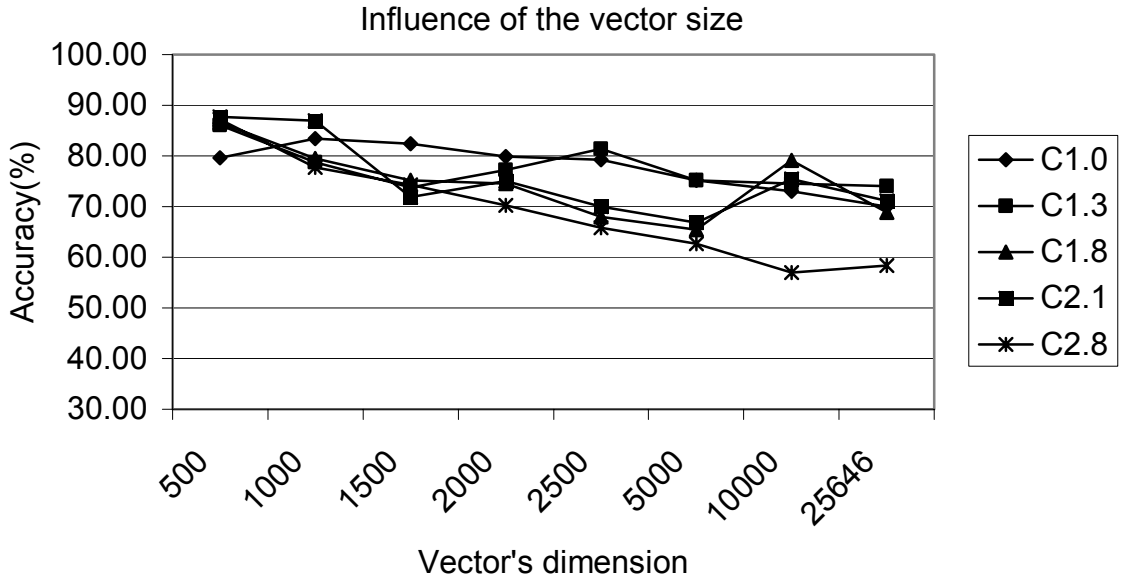

Figure 4.7 – The influence of the vector size for Gaussian kernel and Cornel Smart data representation

Analyzing all figures it 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 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.

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 3.6.1. 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.

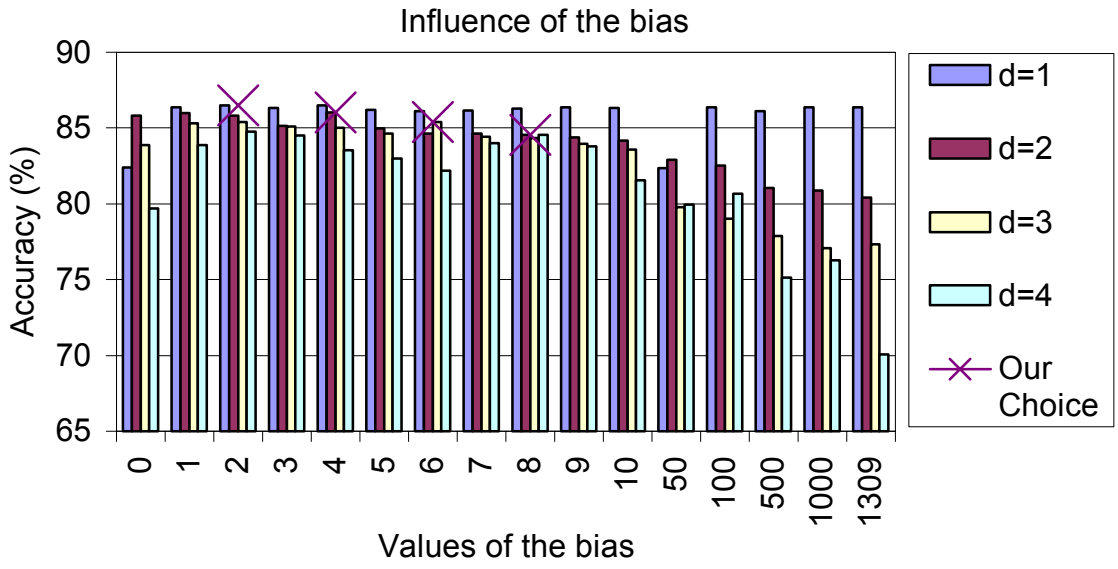

Figure 5.1 – Influence of bias for Nominal representation of the data

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 3.13) 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%).

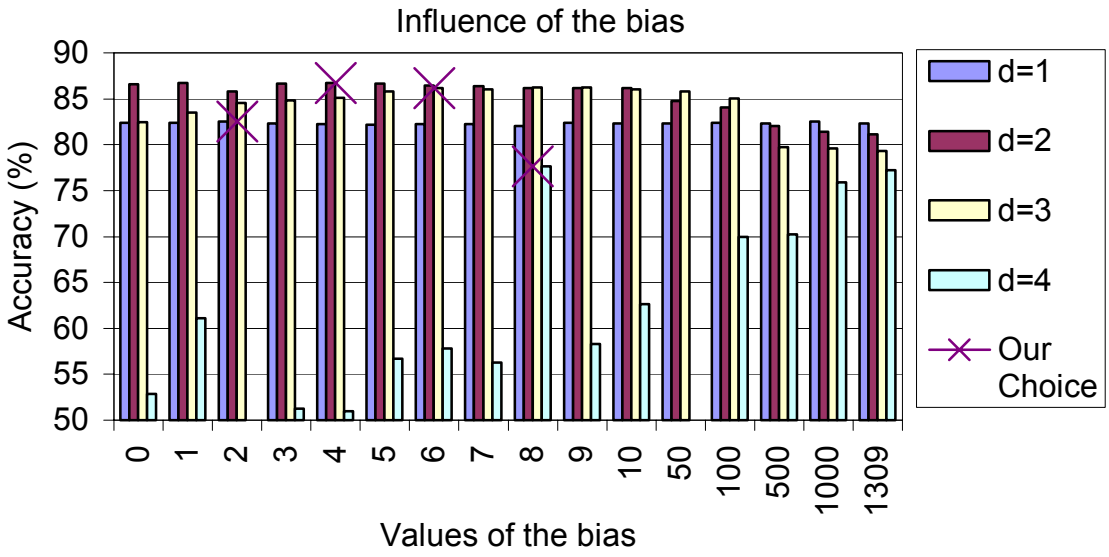

Figure 5.2 – Bias influence for Binary representation of the input data

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 3.14) 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 3.14). 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.

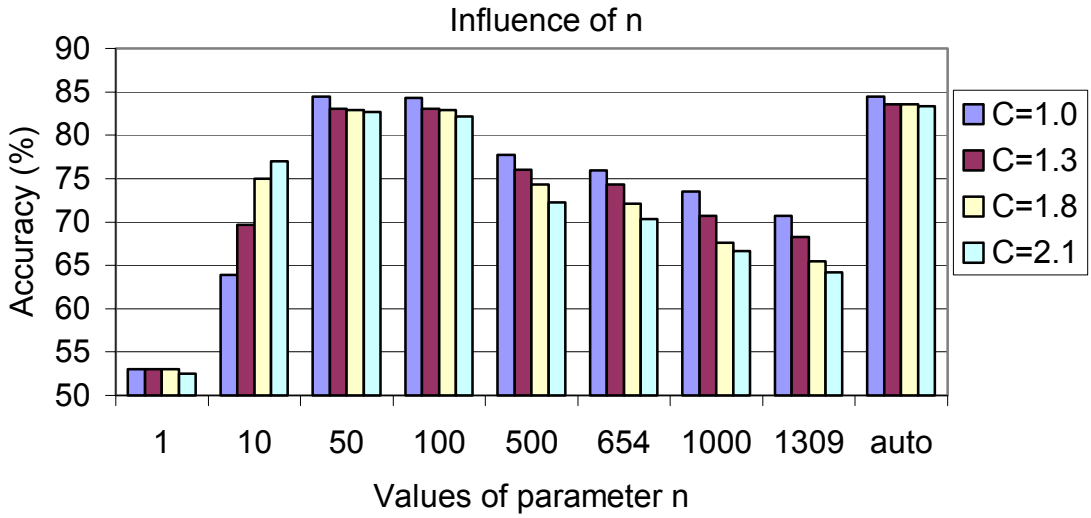

Figure 5.3 – Influence of the parameter "*n*" for Gaussian kernel and Binary data representation

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). In contrast with the polynomial kernel, in the Gaussian kernel case I obtained best results only with my proposed formula to compute the parameter "*n*".

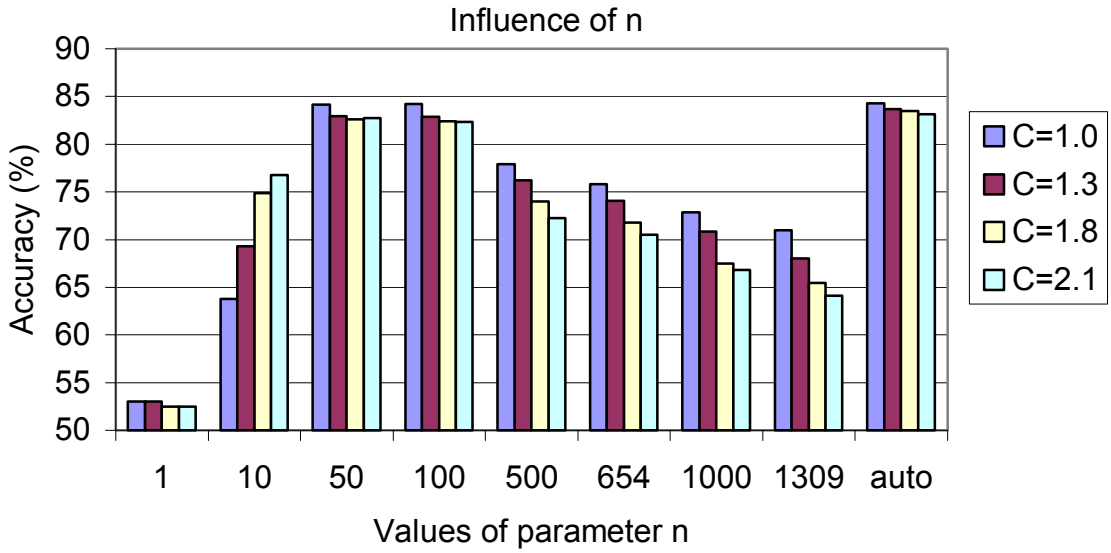

Figure 5.4 – Influence of the parameter *n* for Gaussian kernel and Cornell Smart data representation

5.3 A "de Facto" Standard: LibSVM

Almost all researchers present their results using an implementation of support vector machine technique called "LibSvm" 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 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 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 3.6.1). 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.

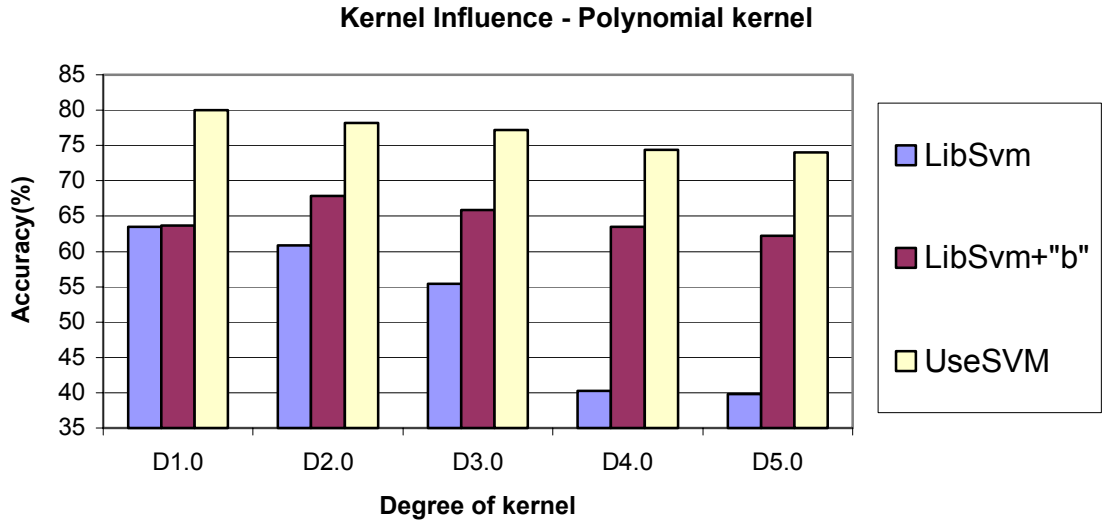

Figure 5.5 – Influence of correlation between parameters from polynomial kernel

As it can be observed from Figure 5.5, 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.

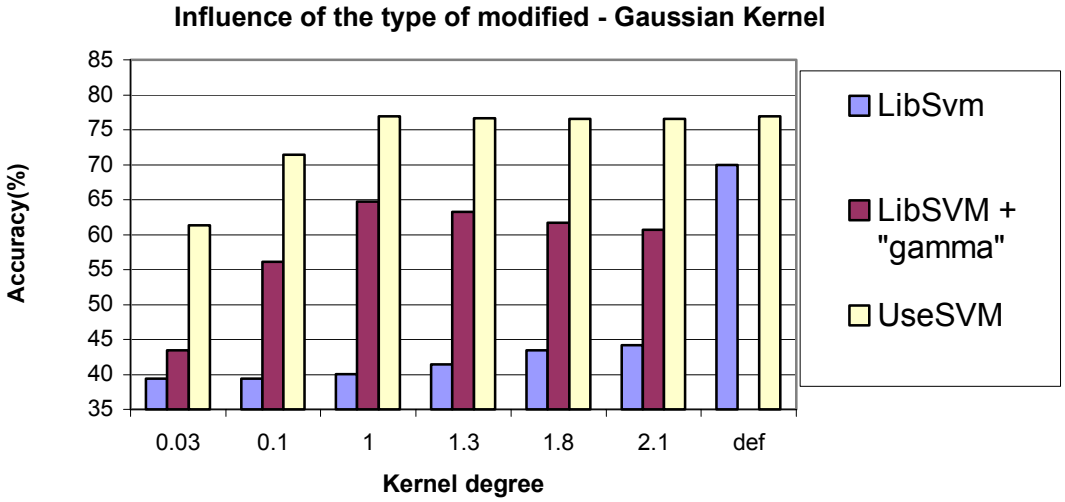

Figure 5.6 – Influence of modified about Gaussian kernel

For the Gaussian kernel simulations, presented in Figure 5.6, my suggestion was to multiply the parameter C with a parameter *n* (like I already explained in Section 3.6.2). 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 5.6.

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).

5.5 Multi-class Classification - Quantitative Aspects

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 function 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.

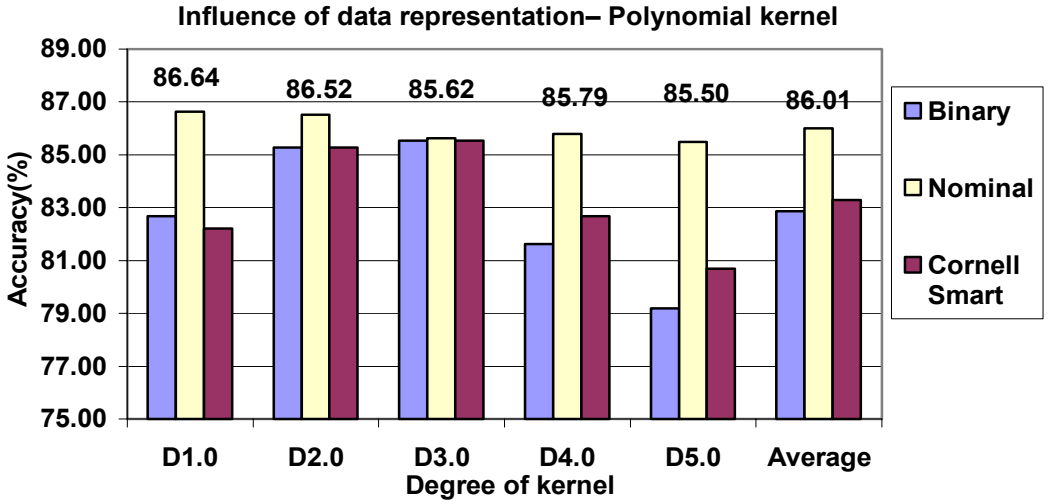

Figure 5.7 – Influence of data representation and degree of polynomial kernel

Figure 5.7 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%).

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 3.2GHz and 1Gb memory with WinXP operating system. More time results were presented in Figure 4.3.

In Figure 5.8 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).

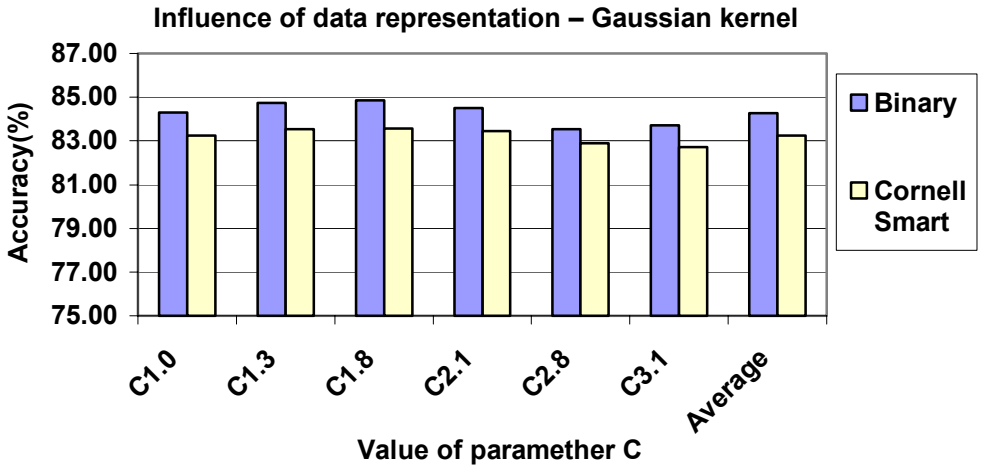

Figure 5.8 – Influence of data representation and parameter *C* for Gaussian kernel

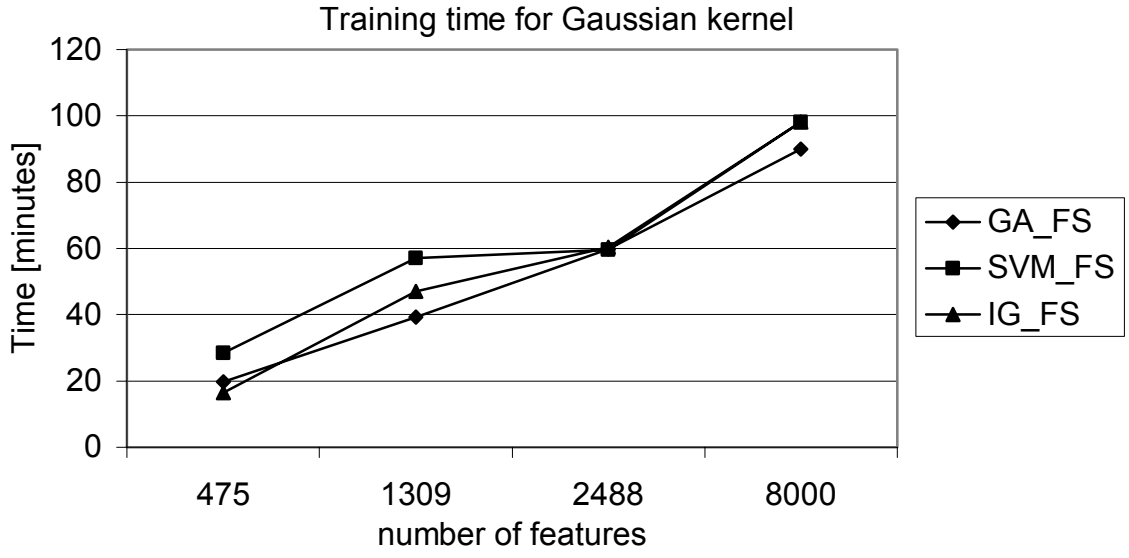

Figure 5.9 – Training time for Gaussian kernel with *C*=1.3 and binary data representation

The training times for the Gaussian kernel are given for parameter C=1.3 and Binary data representation in Figure 5.9. 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.

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

6.1 Selecting Classifiers

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 classifier. 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. After analyzing test results for each (Table 6.1) studied classifier using the same training and testing data set I selected 8 different classifiers as components of the developed meta-classification system.

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 6.1 – The selected classifiers

6.2 The Limitations of the Developed Meta-Classifier System

Another interesting question that occur offer chosen 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 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.

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.

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

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.

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.

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.

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 those 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 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

As I already mentioned the two presented methods SBED and SBCOS 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.

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.

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, SBCOS 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.

In Figure 6.1 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].

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 6.2 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.

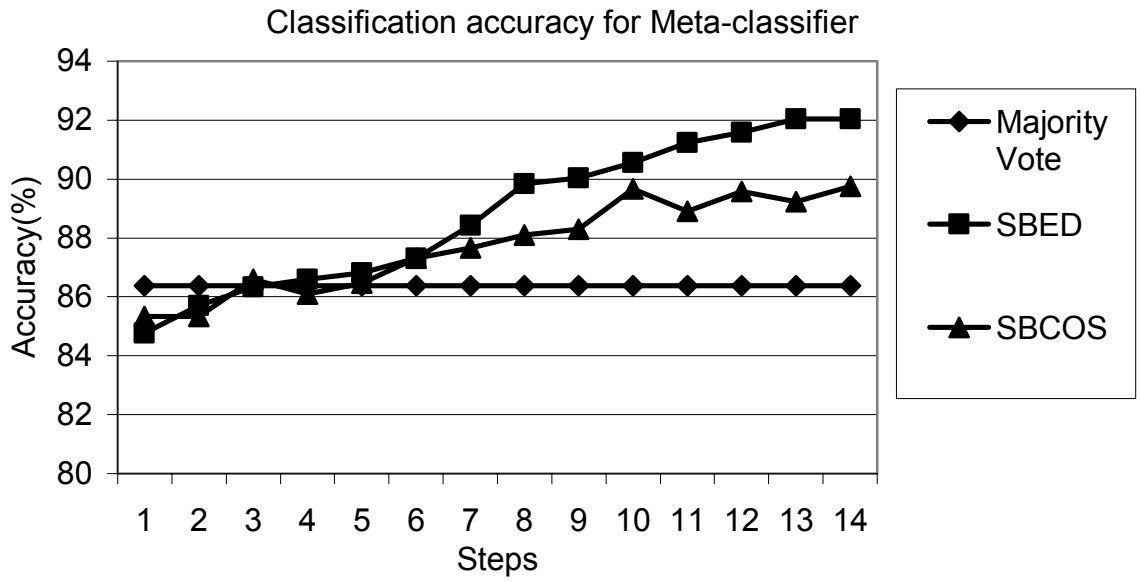

Figure 6.1 – Classification Accuracy for my meta-classifier

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.

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.

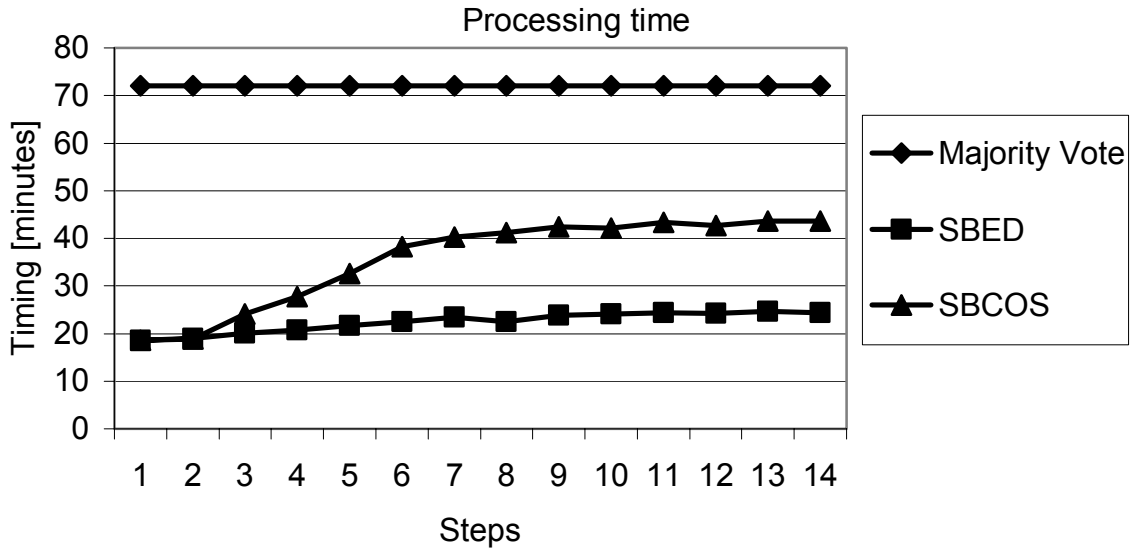

Figure 6.2 – Processing time – comparison between SBED and SBCOS

7 Research Regarding the Methods' Scalability

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 Algorithms' Scalability

In this chapter I want 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 were 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.

Depending of the method used for compute the representative vector I use two type of representing this vector. First of them contain the sum over all elements that are included and a value that represent the total number in order to compute the arithmetical average. The second method each sample is computed using equation 7.1.

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

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.

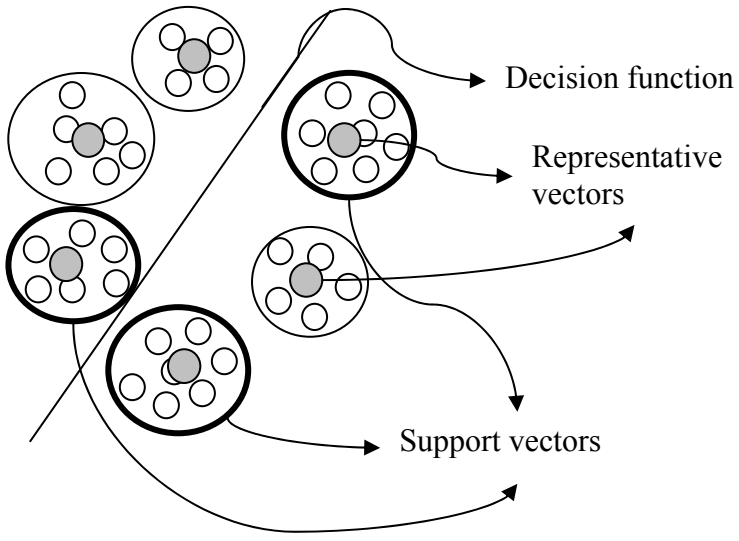

Figure 7.1 – Selecting of support vectors

The entire process (those three stages presented before) is presented in 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:

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

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 (Figure 7.1 the gray small circles) 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.

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

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 large circles from 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 we are 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 we use a classification method, that needs a topic for each vector from the set, we need to specify a topic. This topic is specified automatically as being the most frequent topic that occurs in all input vectors that was grouped together.
4. On this reduced set, as 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 in my second PhD report, too [Mor05_2]. After computing all weights I select only 1309 features because as I showed that offer good results.
5. The resulted smaller vectors are used in a learning step. For this step I use polynomial kernel with degree equal to 1 and nominal data representation. I use polynomial kernel because it usually obtains a small number of support vectors in comparison with Gaussian kernel. I use the kernel's degree equal to 1 because in almost all previously tests I obtained better results.
6. After SVM learning step I chose only those vectors that are support vectors. In the SVM theory the support vectors are that vectors heaving 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 make a new set 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 number of vectors) and containing only vectors that can really contribute at the decision making.

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 try 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 that 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. In this step I select more samples comparatively with the first method presented because in the first step of this method I selected a smaller number of vectors. 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 Algorithms' Scalability - Quantitative Aspects

I make tests only for one reduced vector dimension, for a number of features equal with 1309. I will use only this dimension because I were interested to see if the classification accuracy will go down excessively when I apply a method to select a small (but relevant) number of vectors from the entire set. For example to see much will decrease the classification accuracy, if the original set is reduced with 40%. With the entire set, for this dimension of the features, I obtain the best results.

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

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.

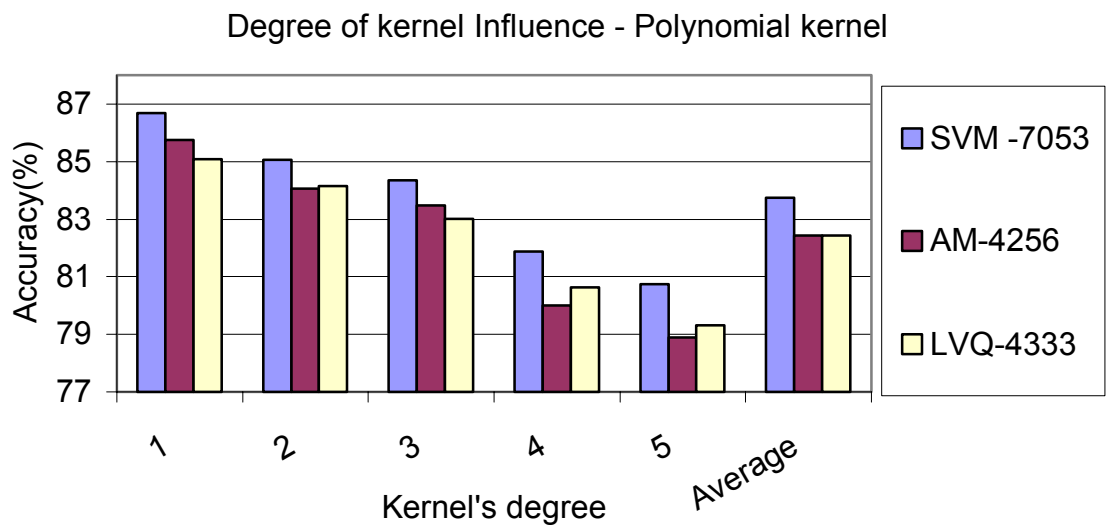

Figure 7.2 – Comparative results for different set dimensions – polynomial kernel

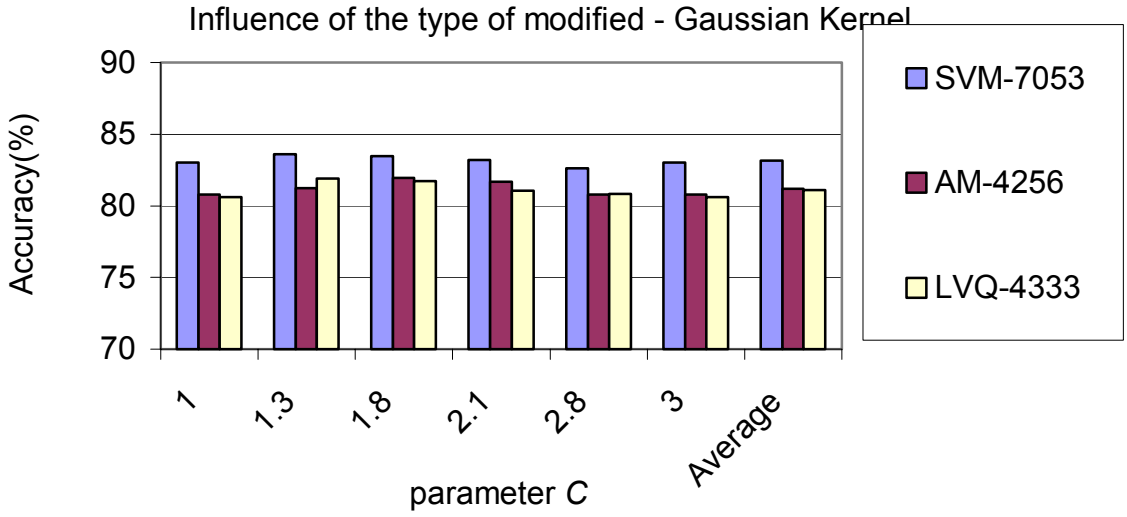

Figure 7.3 – Comparative results for different set dimensions – Gaussian kernel

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.

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.

7.3 Expected Results using a Larger Data Set

My intention is not to classify directly this large set. I first tried to create small groups in order to reduce dimension following steps presented at the beginning of this chapter (section 7.1) and depicted in Figure 7.1. For creating 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 only for selecting 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, when 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 better with 1-2% than results presented.

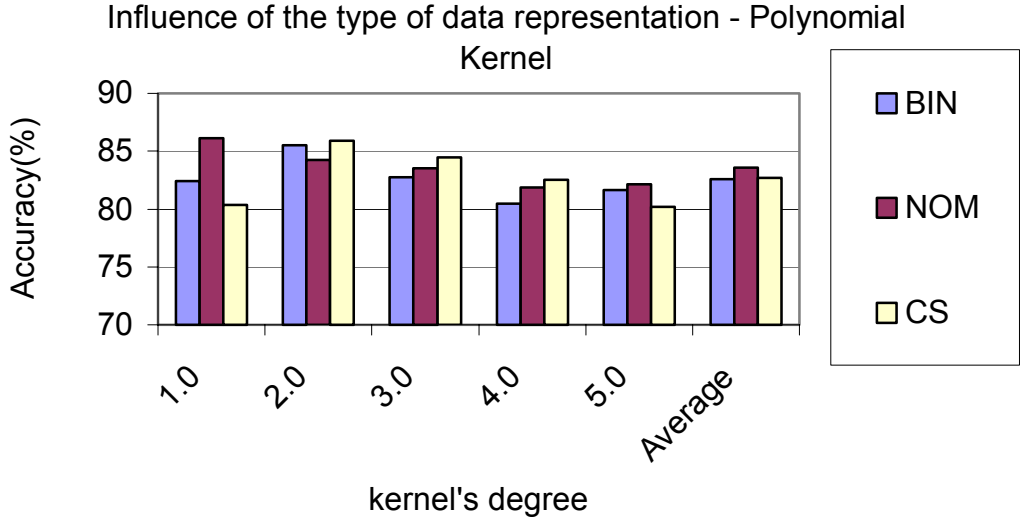

Figure 7.4 – Influence on classification accuracy of type of data representation for Polynomial kernel

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 7.4 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 7.4 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.

8 Thesis 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.

9 Thesis 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.