Advanced Data Mining Techniques for Compound Objects

Dissertation im Fach Informatik
an der Fakultät für Mathematik, Informatik und Statistik
der Ludwig-Maximilians-Universität München

von

Matthias Schubert

Tag der Einreichung: 7. Oktober 2004
Tag der mündlichen Prüfung: 9. November 2004

Berichterstatter:
Prof. Dr. Hans-Peter Kriegel, Ludwig-Maximilians-Universität München
Prof. Dr. Martin Ester, Simon Fraser University, British Columbia (Kanada)

Acknowledgement

There are many people who supported me while I was working on my thesis and I am sorry that I cannot mention all of them in the following. I want to express my deep gratitude to all of them.

First of all, I would like to thank Prof. Dr. Hans-Peter Kriegel, my supervisor and first referee. He made this work possible by offering me the opportunity to work on my own choice of problems in his excellent research group. I benefitted a lot from the opportunities he provided for all of us and enjoyed the inspiring working atmosphere he created.

I want to extend my warmest thanks to Prof. Dr. Martin Ester. He not only willingly agreed to act as my second referee but also shared a lot of his knowledge about scientific work and data mining with me. His encouragement during our cooperation helped me a lot in doubtful times.

Most of the solutions in this thesis were developed in a team and I want to especially thank the people I published with. I know that working with me sometimes demands a lot of endurance and often the willingness to follow my rather broad excursions. I am trying to improve. I especially want to mention Alexey Pryakhin. The cooperation with him during the supervision of his diploma thesis and afterwards as a member of our group was a major influence on the second part of this thesis which I do not want to miss. Scientific research lives in discussions and therefore I want to thank all of my colleagues for many interesting conversations and arguments, not to mention the good times we had.

I would also like to express my deep gratitude to Susanne Grienberger who was a big help in writing down this thesis. She aided me a lot by carefully reading the thesis and offering useful hints for polishing my English. Furthermore, she often shouldered the administrative burdens for me that are part of working at an university. An invaluable assistance for technical problems

I received from Franz Krojer. He always came up with fast solutions if more computing power or additional disc space was needed. So, thank you for always providing running systems in critical times.

I want to thank my parents for their affection and their help for managing my life in busy times. Without you, it would have been very difficult to focus on my research. At last, I want to thank the rest of my family and my friends. Their belief in me was a driving force behind my efforts.

September 2004,

Matthias Schubert

Abstract

Knowledge Discovery in Databases (KDD) is the non-trivial process of identifying valid, novel, potentially useful, and ultimately understandable patterns in large data collections. The most important step within the process of KDD is data mining which is concerned with the extraction of the valid patterns. KDD is necessary to analyze the steady growing amount of data caused by the enhanced performance of modern computer systems. However, with the growing amount of data the complexity of data objects increases as well. Modern methods of KDD should therefore examine more complex objects than simple feature vectors to solve real-world KDD applications adequately. Multi-instance and multi-represented objects are two important types of object representations for complex objects. Multi-instance objects consist of a set of object representations that all belong to the same feature space. Multi-represented objects are constructed as a tuple of feature representations where each feature representation belongs to a different feature space.

The contribution of this thesis is the development of new KDD methods for the classification and clustering of complex objects. Therefore, the thesis introduces solutions for real-world applications that are based on multi-instance and multi-represented object representations. On the basis of these solutions, it is shown that a more general object representation often provides better results for many relevant KDD applications.

The first part of the thesis is concerned with two KDD problems for which employing multi-instance objects provides efficient and effective solutions. The first is the data mining in CAD parts, e.g. the use of hierarchic clustering for the automatic construction of product hierarchies. The introduced solution decomposes a single part into a set of feature vectors and compares them by using a metric on multi-instance objects. Furthermore, multi-step query processing using a novel filter step is employed, enabling the user to efficiently process similarity queries. On the basis of this similarity search system, it is possible to perform several distance based data mining algorithms like the hierarchical clustering algorithm OPTICS to derive product

hierarchies.

The second important application is the classification and search for complete websites in the world wide web (WWW). A website is a set of HTML-documents that is published by the same person, group or organization and usually serves a common purpose. To perform data mining for websites, the thesis presents several methods to classify websites. After introducing naive methods modelling websites as webpages, two more sophisticated approaches to website classification are introduced. The first approach uses a preprocessing that maps single HTML-documents within each website to so-called page classes. The second approach directly compares websites as sets of word vectors and uses nearest neighbor classification. To search the WWW for new, relevant websites, a focused crawler is introduced that efficiently retrieves relevant websites. This crawler minimizes the number of HTML-documents and increases the accuracy of website retrieval.

The second part of the thesis is concerned with the data mining in multi-represented objects. An important example application for this kind of complex objects are proteins that can be represented as a tuple of a protein sequence and a text annotation. To analyze multi-represented objects, a clustering method for multi-represented objects is introduced that is based on the density based clustering algorithm DBSCAN. This method uses all representations that are provided to find a global clustering of the given data objects. However, in many applications there already exists a sophisticated class ontology for the given data objects, e.g. proteins. To map new objects into an ontology a new method for the hierarchical classification of multi-represented objects is described. The system employs the hierarchical structure of the ontology to efficiently classify new proteins, using support vector machines.

Zusammenfassung

Knowledge Discovery in Datenbanken (KDD) ist der nicht-triviale Prozess, neues, gültiges und bisher unbekanntes Wissen aus großen Datenmengen zu extrahieren. Der wichtigste Schritt im KDD Prozess ist das Data Mining, das die in den Daten geltenden Muster findet. KDD ist notwendig, um die stetig wachsenden Datenmengen zu analysieren, die durch die wachsende Leistungsfähigkeit moderner Rechensysteme entstanden sind. Allerdings steigt auch die Komplexität der Objektdarstellung einzelner Datenobjekte an. Moderne KDD Verfahren sollten daher auch mit komplexeren Objekten als einfachen Merkmalsvektoren umgehen können, um reale KDD Applikationen adäquat zu lösen. Zwei wichtige Arten von komplexen Datenmodellierungen sind mengenwertige und multirepräsentierte Objekte. Mengenwertige Objekte bestehen dabei aus einer Menge von Objektrepräsentationen, die alle demselben Vektorraum angehören. Multirepräsentierte Objekte sind durch einen Tupel von Objektrepräsentationen gegeben, die jeweils aus unterschiedlichen Merkmalsräumen stammen.

Das Ziel dieser Doktorarbeit ist es, neue KDD-Verfahren im Bereich Clustering und Klassifikation von komplexen Objekten zu entwickeln. Ausgehend von der Modellierung der Daten als mengenwertige und multirepräsentierte Objekte, werden Lösungen zu realen Anwendungen vorgestellt. Anhand dieser Lösungen wird gezeigt, dass eine allgemeinere Datenmodellierung für viele relevante Anwendungen zu besseren Ergebnissen führt.

Der erste Teil der Doktorarbeit beschäftigt sich mit zwei KDD Problemen, die unter Verwendung von mengenwertigen Datenobjekten besser als durch etablierte Verfahren gelöst werden können. Das erste Problem ist Data Mining von CAD-Bauteilen, wie z.B. das automatische Erstellen von Produkthierarchien mit Hilfe des Clustering. Hierzu werden eine Zerlegung der Bauteile in Mengen von Merkmalsvektoren, eine Metrik auf Vektormengen und passende Methoden zur Ähnlichkeitssuche eingeführt. Auf Basis dieses Suchsystems sind dann viele distanzbasierte Data Mining Algorithmen anwendbar, wie zum Beispiel der Clustering-Algorithmus OPTICS zur Erstellung von Teilhierarchien. Die zweite Anwendung ist die

Kategorisierung und Suche von kompletten Websites im World Wide Web (WWW). Eine Website stellt dabei eine Menge von HTML-Dokumenten dar, die von der gleichen Personengruppe mit demselben Zweck im WWW publiziert wurde. Zunächst werden mehrere Methoden zur Klassifikation von Websites vorgestellt. Die einfachsten versuchen dabei Websites genauso wie einzelne HTML-Dokumente zu klassifizieren. Desweiteren werden zwei fortgeschrittenere Ansätze von Klassifikatoren für Websites eingeführt. Der erste Ansatz verwendet einen Vorverarbeitungsschritt, der die einzelnen HTML-Dokumente auf sogenannte Seitenklassen abbildet. Der zweite Ansatz vergleicht Websites direkt als Mengen von Wortvektoren und wendet Nächste-Nachbar-Klassifikation an. Zur Suche neuer Websites im WWW wird ein fokusierter Webcrawler vorgestellt, der möglichst schnell große Mengen relevanter Websites findet. Das vorgestellte Verfahren minimiert dabei die Anzahl der geladenen Einzeldokumente und erhöht die Trefferrate unter den gefundenen Ergebnissen.

Der zweite Teil der Arbeit beschäftigt sich mit dem Data Mining multirepräsentierter Objekte. Eine wichtige Anwendung für diesen Bereich ist das Data Mining von Proteinen, die durch Aminosäuresequenzen und Text beschrieben werden. Zunächst, wird eine Clustering-Methode vorgestellt, die auf dem dichtebasierten Clustering-Algorithmus DBSCAN basiert. Dabei werden alle vorhandenen Repräsentationen verwendet, um eine globale Einteilung der Proteine zu finden. Häufig besteht allerdings schon eine von Experten erstellte Kategorisierung in so genannten Ontologien. Um bisher noch nicht eingeordnete Objekte in diese Ontologien einzusortieren, wird ein Verfahren zur hierarchischen Klassifikation von multirepräsentierten Objekten vorgestellt. Dieses System nutzt die hierarchische Struktur einer gegebenen Ontologie aus, um die Objekte einer Menge von Klassen mit Hilfe von Support Vektor Maschinen zuzuordnen.

Contents

Part I

Preliminaries

Chapter 1

Introduction

In recent years the amount of data that is collected by advanced information systems has increased tremendously. To analyze these huge amounts of data, the interdisciplinary field of Knowledge Discovery in Databases (KDD) has emerged. The core step of KDD is called Data Mining. Data Mining applies efficient algorithms to extract interesting patterns and regularities from the data. Besides the sheer size of available data sources, the complexity of data objects has increased as well. Thus, new data mining methods are necessary to draw maximum benefit from this additional information. In this chapter the KDD process is introduced and described. Then data mining and its key tasks are surveyed. Clustering and classification are discussed in detail, because these are the tasks the thesis deals with. Afterwards the idea of using compound data objects for the representation of complex objects is introduced. Finally the chapter concludes with an outline of the thesis, offering a brief overview of the introduced solutions.

1.1 Knowledge Discovery in Databases

In recent years the amount of data collected and stored by electronic devices has risen tremendously. For example, earth observation satellites retrieving images, bar code scanners collecting costumer data, and companies mapping costumer preferences in data warehouses are generating gigabytes of data every day. Another rapidly growing information collection is the World Wide Web (WWW). Currently the web provides more than 4 billions [Cen] webpages containing information about almost any imaginable topic.

All of these data collections are far to large to be examined manually and even the methods for automatic data analysis based on classical statistics and machine learning often face problems when processing large, dynamic data collections consisting of complex objects. To analyze these large amounts of collected information, the area of Knowledge Discovery in Databases (KDD) provides techniques which extract interesting patterns in a reasonable amount of time. Therefore, KDD employs methods at the crosspoint of machine learning, statistics and database systems. In [FPSS96] KDD is defined as follows :

Knowledge Discovery in Databases *is the non-trivial process of identifying valid, novel, potentially useful, and ultimately understandable patterns in data.*

According to this definition, data is a set of facts that is somehow accessible in electronic form. The term "patterns" indicates models and regularities which can be observed within the data. Patterns have to be valid, i.e. they should be true on new data with some degree of certainty. A novel pattern is not previously known or trivially true. The potentially usefulness of patterns refers to the possibility that they lead to an action providing a benefit. A pattern is understandable if it is interpretable by a human user. At last KDD is a process, indicating that there are several steps that are repeated in several iterations.

Figure 1.1 displays the process of KDD in its basic form. The process

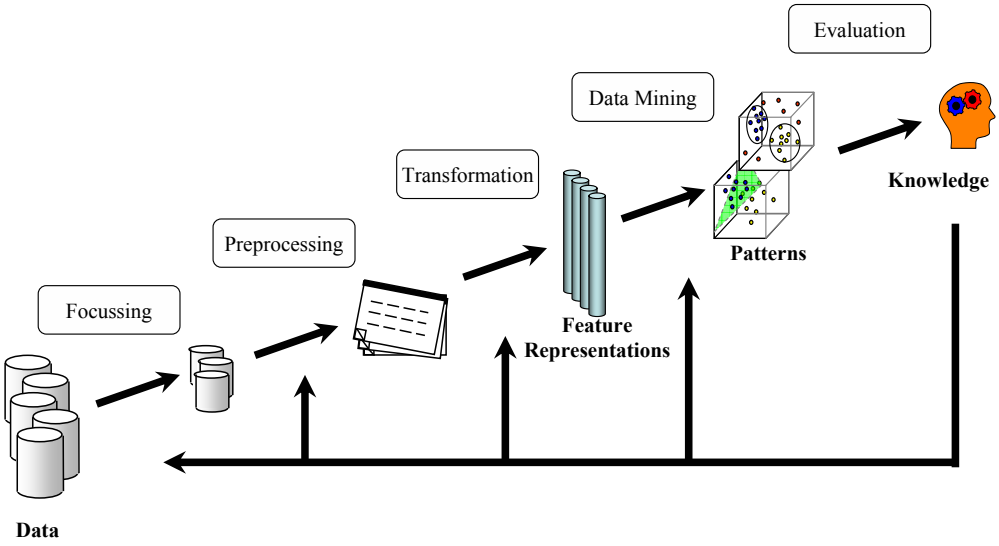


Figure 1.1: The process of KDD.

comprises the following steps:

- **Focussing**
  The first step is to define the goal of the particular KDD task. Another important aspect of this step is to determine the data to be analyzed and how to obtain it.

- **Preprocessing**
  In this step the specified data has to be integrated, because it is not necessarily accessible on the same system. Furthermore, several objects may be described incompletely. Thus, the missing values need to be completed and inconsistent data should be corrected or left out.

- **Transformation**
  The transformation step has to assure that each data object is represented in a common form which is suitable as input in the next step. Thus, certain attributes are selected and others are left out. Furthermore, some attribute values have to be discretized, depending on the

algorithms used in the next step. Note that the chosen features and the type of object representation can have a major influence on the resulting patterns. Prominent examples of object representations that are results of the transformation step are feature vectors or itemsets.

- **Data Mining**
  Data mining is the application of efficient algorithms to detect the desired patterns contained within the given data. Thus, the data mining step is responsible for finding patterns according to the predefined task. Since this step is the most important within the KDD process, we are going to have a closer look at it in the next section (1.1.1).

- **Evaluation**
  At last, the user evaluates the extracted patterns with respect to the task defined in the focussing step. An important aspect of this evaluation is the representation of the found patterns. Depending on the given task, there are several quality measures and visualizations available to describe the result. If the user is satisfied with the quality of the patterns, the process is terminated. However, in most cases the results might not be satisfying after only one iteration. In those cases, the user might return to any of the previous steps to achieve more useful results.

The quality of the results varies, depending on the right choice of feature transformations, feature selections and data mining algorithms. A user must decide which techniques should be employed and select the step where the KDD process should be modified to improve the result after each iteration. Thus, the number of iterations and the quality of the results strongly depend on the user directing the KDD process.

1.1.1 Data Mining

Since data mining is the most important step within the KDD process, we will treat it more carefully in this section. In [FPSS96] Data Mining is defined as follows:

Data mining *is a step in the KDD process consisting of applying data analysis and discovery algorithms that, under acceptable computational efficiency limitations, produce a particular enumeration of patterns over the data.*

According to this definition data mining is the step that is responsible for the actual knowledge discovery. To emphasize the necessity that data mining algorithms need to process large amounts of data, the desired patterns has to be found under acceptable computational efficiency limitations. Let us note that there are many other definitions of data mining and that the term data mining and KDD are often used in a synonymous way.

In the following, we will describe the most important data mining methods with respect to the kind of knowledge they mine:

- **Classification** (also called supervised learning)
  Classification is the task of learning a function that maps data objects to one or several classes in a predefined class set. To learn this function, classification methods need a training set, containing data objects that are already mapped to the class they belong to. After analyzing the training set, classification methods can map new unknown objects to the classes. A second purpose of classification is, deriving class models to explain why the objects are mapped in this way. In section 1.1.2 and chapter 2.3, classification is treated more precisely.

- **Clustering** (also called unsupervised learning)
  Clustering is the task of identifying a finite set of categories (or clusters) to describe the data. Thus, similar objects are assigned to the same category and dissimilar ones to different categories. Clustering is also

called unsupervised learning because the data objects are mapped to a set of clusters which can be interpreted as classes as well. In section 1.1.2 and chapter 2.2, clustering and cluster algorithms will be discussed more exhaustive.

- **Association Rules**
  Finding Association rules is the task of identifying rules that express co-occurrences within transaction databases. A transaction is a set of items where each item has a different type. Association rules express that in the given database a specified set of items appears together in the same transaction with a certain support/probability. The most important example of transaction data is market basket data.

- **Data Generalization**
  Data Generalization derives compact representations for a subset of data objects.

- **Regression**
  The task of regression is to learn a function which maps data objects to a real value. To find a regression function, a training set of data objects that are already mapped to a real value is necessary. An additional goal of regression is to discover functional relationships between the feature values of the underlying training objects. Regression is related to classification, since both tasks learn functions from a training set.

- **Dependency Modelling**
  Algorithms in this category are designed to find a model which describes significant dependencies between variables, e.g. learning or believe networks.

A second important categorization is to distinguish the databases a data mining algorithm is build on. The characterization of the database can have a major effect on the data mining algorithm, since data objects in varying

kinds of databases usually are represented in a different way and contain different kinds of information. Important groups of databases are relational, object-relational, spatial and deductive databases. Most of the so far proposed data mining methods are developed for spatial and pure relational databases. [CHY96] surveys data mining techniques for relational databases and [KAH96] contains an overview of spatial data mining techniques.

1.1.2 Clustering and Classification

The main contribution of this thesis is the development of new methods for clustering and classification. Therefore, these very important data mining tasks and their relation to each other are described more closely in the following.

Clustering

Clustering is the process of grouping the data records into meaningful subclasses (clusters) in a way that maximizes the similarity within clusters and minimizes the similarity between two different clusters [KHK99].

Other names for clustering are unsupervised learning (machine learning) and segmentation. Clustering is used to get an overview over a given data set. A set of clusters is often enough to get insight into the data distribution within a data set. Another important use of clustering algorithms is the preprocessing for some other data mining algorithm. In chapter 2.2.2 a general categorization and important examples of clustering algorithm are described more closely.

Example applications of clustering are the grouping of costumers for target marketing, grouping HTML-documents to order the answer sets of search engines [BGG+99a, BGG+99b] or finding clusters of proteins and genes having a similar function [HKKM98, SCC+95, NRS+95].

Classification

Classification is the process of learning a function that maps data objects to a subset of a given class set. Therefore, a classifier is trained with a labelled set of training objects, specifying each class. There are two goals of classification:

- Finding a good general mapping that can predict the class of so far unknown data objects with high accuracy. For this goal, the classifier is a mere function. To achieve this goal, the classifier has to decide which of the characteristics of the given training instances are typical for the complete class and which characteristics are specific for single objects in the training set.

- The other goal of classification is to find a compact and understandable class model for each of the classes. A class model should give an explanation why the given objects belong to a certain class and what is typical for the members of a given class. The class model should be as compact as possible because the more compact a model is, the more general it is. Furthermore, small and simple class models are easier to understand and contain less distracting information.

Of course, a good classifier should serve both purposes, but for most practical applications finding an accurate mapping is more important than developing understandable class models. Thus, multiple techniques are used to classify objects that do not offer an understandable class model. A categorization of different kinds of classification methods is given in chapter 2.2.3.

Example applications of classification methods are mapping emails into one out of a determined set of folders, predicting the functional class of proteins [DK02, JH98], finding relevant information in the WWW [CDI98, CvdBD99a, CvdBD99b], and predicting the costumer class for a new customer.

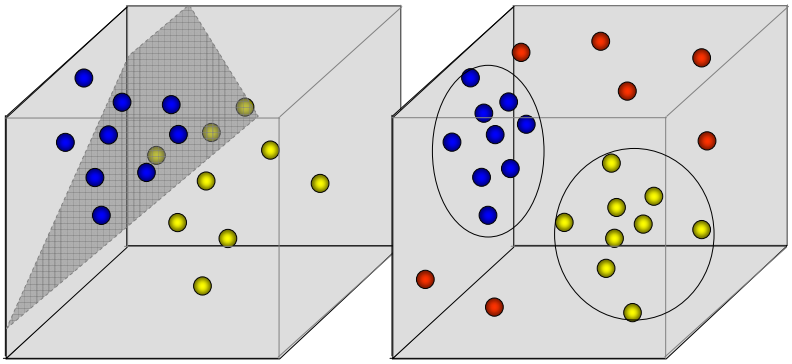


Figure 1.2: Classification separates the data space (left) and clustering groups data objects (right).

Classification and Clustering are strongly connected. Classification tries to learn the characteristics of a given set of classes, whereas clustering finds a set of classes within a given data set. An important feature of clustering is that it is not necessary to specify a set of example objects. Therefore, clustering can be applied in applications where there is no or little prior knowledge about the groups or classes in a database. However, the usefulness of a found clustering is often subject to individual interpretation and strongly depends on the selection of a suitable similarity measure. In applications for which the existence of a dedicated set of classes is already known, the use of classification is more advisable. In these cases providing example objects for each class is usually much easier than constructing a feature space in which the predefined classes are grouped into delimited clusters. Furthermore, the performance of a classifier can easily be measured by the percentage of correct class predictions it achieves. To conclude, clustering and classification are related data mining tasks that are used in different situations. Figure 1.2 displays class separation by a classifier on the left side and the grouping of two clusters in a noisy data set on the right side.

1.1.3 Data Mining and Complex Objects

For many standard applications, like market basket analysis, constructing a usable KDD process is a rather well determined task. However, the data to be processed in real-world applications is getting more and more complex and is yielding more potential knowledge. With advancing processors, memory and disc space, the detail level of objects is increasing as well as their plain numbers. For example, companies acquire more detailed information about their costumers, sky telescopes offer pictures with higher resolutions and HTML-documents use structural tags, embedded multimedia content and hyperlinks which makes them much more complicated than ordinary text documents.

All these additional information yields new challenges to KDD. Though it is basically desirable to have more information about given data objects, the selection of characteristics that are used in data mining gets more difficult. Additionally, many complex objects provide structural information as well as plain features. For example, a gene sequence is characterized by the order of nucleotides instead of their plain appearance in the gene.

To analyze complex objects, the most established way is to map any complex object to a feature vector. The idea is to span a vector space in which each relevant object characteristic or feature provides a dimension. Thus, an object is represented by the vector of its feature values. Since this is the most common feature representation, there is a wide variety of data mining algorithms that can process vectors as input representation. Though this method offers good results in many application areas, the data transformation becomes more and more difficult with increasing object complexity. Since data transformation usually is not informed about the purpose of the KDD task, it is difficult to decide which characteristic of an object should be preserved and which can be neglected. Furthermore, structural information is very difficult to express using a single feature vector. For example, it is not possible to model an arbitrary sized set within a feature vector without loos-

ing information. Thus, transforming complex objects into a feature vector and employing vector-based data mining often spends large efforts for data transformation and provides suboptimal results.

For several applications, it is more beneficial to employ specialized data mining algorithms that can process more complex input representations than plain feature vectors. Employing structured object representations like graphs, sequences or relational data, often provides an more natural view on real-world complex objects. The type of data representation discussed in this thesis is called compound object representation and is also capable to model structural information. A compound data object is defined in the following way:

Compound Data Objects *are built of concatenations and sets of other compound data objects. Basic compound objects can consist of any object representation that can be processed by a data mining algorithm.*

The most simple type of compound object is a value v for a given object domain D. A concatenation of this basic type is a feature vector $(v_1, \ldots, v_d) \in D_1 \times \ldots \times D_d$. However, the idea of compound objects is not limited to this kind of basic objects. Other examples are trees, graphs, and sequences that could be used as basic objects, too. Figure 1.3 illustrates two basic types of compound objects, concatenated or multi-represented objects and set-valued or multi-instance objects. The following directions are capable to process compound objects for data mining.

Distance based data mining algorithms like the density-based clustering algorithms DBSCAN, OPTICS, k-medoid clustering or k nearest neighbor (kNN) classification can process any data objects as long as there is a suitable distance function. For this approach, the treatment of the structural information completely relies on the distance function. For some kinds of compound objects, there already exist several distance functions [EM97, RB01], each offering a different interpretation of the structure. The challenge of this approach is to find out which distance function is suited best for a given application.

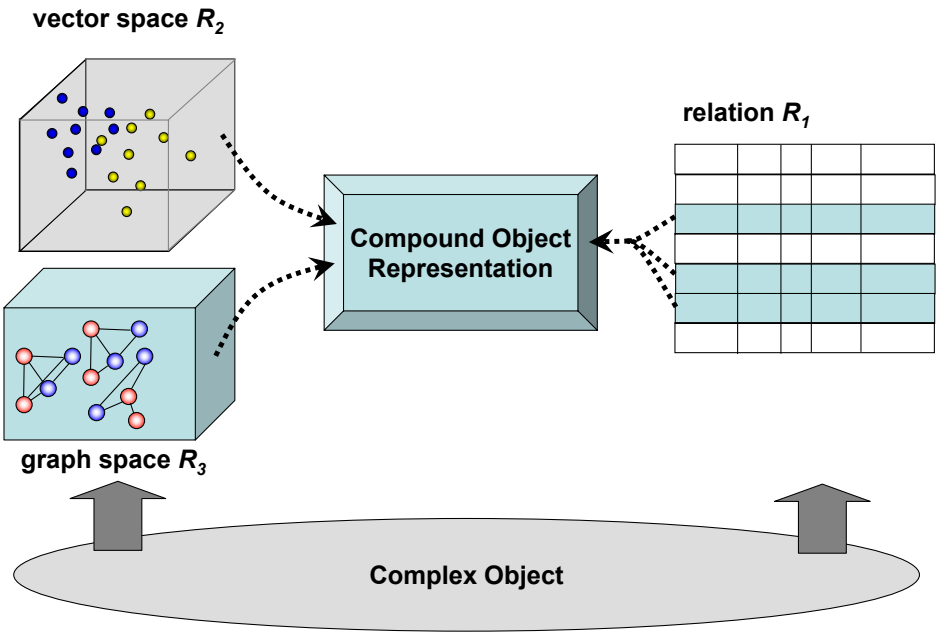


Figure 1.3: Multi-represented and multi-instance objects are basic types of compound objects.

Another approach considering compound data is multi-instance learning. A multi-instance learner tries to solve a two class classification problem in which each training object is given by a bag or set of feature vectors [DLLP97]. An object is relevant, if there is at least one relevant instance within the set. An object is irrelevant, if there is none relevant instance within the set. Thus, the problem of the classifier is to find out which kind of instance makes an object relevant in order to decide the class of new sets of instances. There have been several methods developed so far to solve this problem [DLLP97, WZ00, CZ00, GFKS02]. However, this direction treats only a very specialized interpretation of data mining in multi-instance objects. Other directions that are related to data mining in compound objects are generalization [HNKW98], ensemble learning [VM02] and relational data mining [DL01].

1.2 Outline of the Thesis

This thesis is aimed at the development of methods for classification and clustering of two basic types of compound objects: multi-instance objects and multi-represented objects.

Multi-instance objects are given by a set of object representations similar to the objects mined in multi-instance learning. However, the goal of multi-instance data mining is much more general. Therefore, multi-instance data mining algorithms should consider a wide variety of dependencies between the instances of two objects. An example for multi-instance objects are websites which can be described as sets of HTML-documents. Therefore, websites can be represented as sets of feature vectors.

Multi-represented objects are concatenations of several different kinds of object representation. Unlike modelling an object by spanning a single feature space, multi-represented objects are given by a tuple of feature representations. Each of these feature representations exists in a separated feature space. For some objects, there exists no feature representation in some of these data spaces. Thus, methods of multi-represented data mining have to cope with missing object representations. An example for data objects that can be transformed into multi-represented form are proteins which can be described by text annotations, the amino acid sequence and structural feature representations.

The use of data mining methods using compound objects depends on the given application. For many applications, data mining based on feature vectors offers very good results because the structural information is not relevant for the given task. To demonstrate the need for data mining using compound objects, this thesis introduces new methods of clustering and classification of compound objects that offer advanced solutions to important data mining applications. Each of the introduced solutions is capable to exploit the structure of the given input data and to outperform conventional approaches using feature vectors.

The thesis consists of four parts. The rest of the first part (chapter 2) contains a brief overview of feature transformations, classification and clustering algorithms that are necessary to understand the methods introduced in the next two parts of the thesis.

The second part describes methods employing multi-instance objects to offer solution to two important applications. The part starts with a chapter (3) that formally introduces multi-instance data mining, describes several important applications, and distinguishes the introduced solution from the area of classical multi-instance learning.

Chapter 4 is concerned with shape-based data mining in CAD parts. It starts by introducing a representation of CAD parts as sets of covers. To measure the similarity between two multi-instance objects, the "minimal matching distance" is used which is a metric on sets of vectors that is computable in polynomial time. Based on this representation and distance function, an efficient similarity search system is introduced that uses multi-step query processing. Given the similarity search system, various distance based data mining algorithms are applicable.

In chapter 5 we introduce data mining aimed at the efficient retrieval of relevant websites or so-called website mining. A website is a set of HTML-documents that is published by the same person group or institution and is usually serving a common purpose. Websites are interesting objects for data mining, since there are many important applications directed at websites instead a single HTML-documents. A basic task of website mining is the classification of websites. After discussing naive approaches, we introduce two more sophisticated solutions to website classification. The first approach employs a preprocessing step that maps the webpages of a website to a predefined set of page classes. Based on this preprocessing step, multiple methods of website classification are discussed. The second approach to website classification does not employ any preprocessing step, but directly compares websites as multi-instance objects of text feature vectors. To further improve the performance of website classification, a method for reducing the number

of webpages that are needed for the classification of a website is introduced. The last and most important contribution of this chapter is a focused crawler that is specialized on retrieving new relevant websites from the WWW while reading as few webpages as possible. The idea of the crawler is to employ a two level architecture. The first level, called the external crawler, ranks candidates for relevant websites and decides which of these candidates has to be explored next. The second level, called the internal crawler, examines these candidates and decides if they are indeed relevant.

Chapter 6 concludes the part about data mining in multi-instance objects. Since the last two chapters contain solutions to real-world problems containing aspects that are not specific to multi-instance objects, the solutions to multi-instance data mining are especially summarized. Furthermore, the chapter draws general conclusion from the introduced techniques.

The third part of the thesis is concerned with data mining in multi-represented objects. The first chapter (7) of this part gives an introduction to multi-represented objects and names several important applications.

Chapter 8 introduces a method for density-based clustering of multi-represented objects. Therefore, two new methods are introduced that are based on the density-based clustering algorithm DBSCAN. Both methods are capable to find more meaningful clusters based on more than one object representation. The introduced clustering methods are applied to a protein database, consisting of text annotations and amino acid sequences. Furthermore, the usefulness of the method is demonstrated on an image database where each image is described by a color histogram and a segmentation tree.

In chapter 9, we introduce a solution to ontology-based data integration of biomolecular databases. Therefore, we introduce a classification method for mapping multi-represented objects into large class ontologies. The introduced classification system is based on support vector machines (SVM) and utilizes the inheritance relationships within the given class ontology. Furthermore, the system is able to predict a set of classes for each data object which is an important requirement for protein classification. To draw max-

imum knowledge from multiple representations, a technique called "object-adjusted weighted"(OAW) is introduced. OAW recombines the classification results achieved in each representation for each object and uses the reliability of the results for finding a global class prediction. The system is evaluated by mapping entries of the SWISS-PROT protein database [BBA+03] to the classes of Gene Ontology [Con00].

The part of the thesis that deals with multi-represented data objects is concluded by chapter 10. This chapter contains a brief summary of the introduced data mining techniques with respect to multi-represented data mining. Furthermore, a categorization of problems in multi-represented data mining is provided.

The fourth and final part contains a summary of the thesis. The last chapter (11) sums up the introduced methods and draws general conclusions. Afterwards ideas for future work are presented that contain several interesting directions for data mining of compound objects and for the introduced KDD applications.

Chapter 2

Basic Techniques of Knowledge Discovery and Data Mining

There are many established data mining algorithms that can be applied to a given data mining task. In this chapter well established data transformations and data mining techniques that are used by the novel solutions are described. The first section illustrates the basic idea of data transformation more formally. Additionally, similarity measures and basic similarity queries are described. Finally, data transformation, feature selection and distance functions for text documents are introduced. The next section gives a brief overview and categorization of clustering algorithms. To understand the contribution of the thesis, it is necessary to discuss the clustering algorithms DBSCAN and OPTICS more closely. The last section deals with classification algorithms. It begins with a small introduction about the general characteristic of classification and its evaluation. Afterwards important directions, like kNN classification, statistical classifiers, support vector machines and decision trees, are surveyed.

2.1 Feature Spaces and Data Transformation for KDD

2.1.1 Feature Transformation and Similarity Search

This section describes the idea of data transformation. Additionally, similarity measures are described to compare the given data objects. Finally, similarity search and basic types of similarity queries are explained.

In most KDD applications, the data objects are not provided in a form that can be processed by a data mining algorithm. Therefore, the data objects have to be transformed into a more meaningful representation that directly expresses the potentially relevant characteristics. For example, images are transformed into feature vectors that describe shape, texture or color [HSE+95, NBE+93, SH94] or text documents in word vectors [Sal89]. In general, a feature transformation is a function of the following form.

Definition 2.1 (feature transformation)
Let $o \in O$ be an object of the object space O and let F be a feature space. Then, feature transformation is a function, mapping the objects of O to F:

$$FT : O \rightarrow F$$

The object space O can be given by any object representation that is processable by a computer. The feature space F might be a vector space, a compound object or any other arbitrary form that is processable by the chosen data mining algorithm. An important assumption about feature spaces is that object similarity is reflected within the distance of feature representations. In other words, two objects that are similar should be transformed to feature representations that are close with respect to a suitable distance function. On the other hand, the feature representations of dissimilar objects should be far away from each other. To fulfill this assumption, a suitable distance measure has to be selected. For most types of feature spaces there exist multiple distance measures that are suitable for certain applications,

but do not reflect object similarity in other cases. Formally, distance based similarity can be defined as follows:

Definition 2.2 (distance based similarity)
Let O be the set of data objects and let $FT : O \Rightarrow F$ be a feature transformation into the feature space F. Furthermore, let $dist : F \times F \rightarrow \mathbb{R}$ be a distance measure in the feature space F. The similarity of objects obj_1, $obj_2 \in O$ is defined as follows:

$$simdist(obj_1, obj2) = dist(FT(obj_1), FT(obj_2))$$

The most established type of distance functions are the l_p distance functions which are defined as follows:

Definition 2.3 (l_p-distance metrics)

$$l_p : \mathbb{R}^d \times \mathbb{R}^d \rightarrow \mathbb{R} \ : \ l_p(x,y) = \left(\sum_{i=1}^{d} |x_i - y_i)^p \right)^{\frac{1}{p}}, \ p > 1$$

For $p = 2$ the l_p is called Euclidian distance function and is used as the default distance function for most applications using vector spaces. The l_p distance functions are metric which means that they fulfill the following conditions:

Definition 2.4 (Metric Distance Function)
Let $dist : F \times F \rightarrow \mathbb{R}$ be a distance function. dist is called a metric iff

1. $dist(x,y) = 0 \Leftrightarrow x = y \ \forall x, y \in F$
2. $dist(x,y) = dist(y,x) \ \forall x, y \in F$
3. $\forall x, y, z \in F : dist(x,z) \leq dist(x,y) + dist(y,z)$

Using metric distance functions provides many benefits like the use of efficient indexing and data structures [BKK96, CPZ97, LJF94, BBJ+00], but it is not mandatory for achieving good data mining results.

Having a notion of similarity that is processable by a computer, many data mining algorithms become applicable. On the other hand, it is not necessary to define an explicit distance measure for a wide variety of data mining algorithms like kernel based methods or decision trees. To distinguish these two directions of data mining algorithms, we call all data mining algorithms that demand the selection of a distance function *distance based data mining algorithms.*

Distance based data mining algorithms often use similarity queries as basic operations. The two most important types of similarity queries are:

1. **Definition 2.5 (ε-range queries)**
   Let $DB \subset F$ be a set of objects that is given in feature space F and let $simdist : F \times F \rightarrow \mathbb{R}$ a similarity measure. Then, the result of an ε range query to the set DB with respect to an query object $q \in F$ is defined as follows:

   $$\mathcal{N}_{\varepsilon}(q) = \{x | x \in DB \wedge simdist(q, x) < \varepsilon\}$$

   The result of an ε range query is the subset of a data set that is contained within the hypersphere around the querypoint q having the radius ε.

2. **Definition 2.6 (k nearest neighbor (kNN) queries)**
   Let $DB \subset F$ be a set of objects that is given in feature space F and let $simdist : F \times F \rightarrow \mathbb{R}$ a similarity measure, then the result of an k nearest neighbor query to the set DB with respect to an query object $q \in F$ is defined as follows:

   $$NN_k(q) = \{x_1, \ldots, x_k | x \in DB \wedge \nexists\, x' \in DB \setminus \{x_1, \ldots, x_n\}$$
   $$\wedge \nexists\, i,\ 1 \leq i \leq k : simdist(x_i, q) > simdist(x', q)\}$$

   In other words, a kNN query for the query object q returns k objects having the minimal distance to q.

In the following, we will call a system that enables us to process similarity queries a *similarity search system*.

2.1.2 Data Transformation for Text Documents

An important part of this thesis deals with processing text data. There are several established techniques that are aimed at increasing the performance for this important application area [Sal89, YL99, DMS98, Lar98]. The following section gives an introduction to data transformation methods for text documents which we use in this thesis.

The most established way to process text documents is to treat each document as a so-called "bag of words". In this representation, each word occurring in any document provides a dimension for a vector space. A document can be represented as a feature vector where each dimension provides the number of occurrences of the word corresponding to this dimension. A more general approach to text mining is to use terms instead of words. A term can be a small sequence of words or a single word. The techniques processing more complicated terms are the same as for single words.

Since most documents contain only a limited number of terms, the feature vectors that are build in this way contain an extraordinary high percentage of zero-valued dimensions. Therefore, the feature vectors are called sparse and need to be processed by methods that can cope with this extraordinary characteristic. An example effect that occurs data spaces of sparse feature vectors is that usually all objects are placed on the surface of the feature space, since there is no document that contains all words.

Another problem of text feature spaces is the number of possible dimensions. Obviously, it does not make sense to use every possible term. Most of the dimensions would be used for almost no text document and the extremely high dimensionality of the feature space would make efficient calculation very difficult.

There are several techniques that reduce the number of terms that have to

be considered. A method that can be used especially for words is stemming which is the reduction of each word to its basic word stem. Using stemming reduces the number of terms by mapping several words into the same dimension. There are two directions to achieve stemming: lexical and algorithmic stemming. Algorithmic stemming employs rules to reduce a word, whereas lexical stemming does rely on dictionaries to find out if a word is derived from a particular word stem. Since algorithmic stemming is not applicable for all languages and large dictionaries are hard to acquire, both directions have their drawbacks. Professional text mining tools usually use a mixture of both. [Tom04] compares stemming for nine European languages.

Another established approach to reduce the dimensionality of a feature space is the deletion of stop words like "to", "like" or "would". These words usually do not give a hint to the topic of a given text document. Stop words lists for common languages can be found on the WWW, e.g. [Con].

Though stemming and the removal of stop words reduce the potential candidates for spanning a feature space, the number is usually still very big. Thus, most text transformations use methods of feature selection to select the most meaningful terms for building a feature space. The research community has spent a lot of attention on feature selection and there are several techniques that have proven to be useful. [YP97] contains an comparative study of some of the most established methods of feature selection for text mining.

In the following, we will introduce the feature selection methods that achieved the best results in [YP97].

The first is called "feature selection by document frequency". For each term t the number of text documents in the database DB that the term occurs in, is counted denoted as $cont(t, DB)$. Afterwards, the terms are sorted descending with respect to $cont(t, DB)$ and each term is given a rank with respect to its order. Thus, the term occurring in the most documents, receives the top rank (rank 1) and so on. To find the final score for each term, the rank is multiplied with $cont(t, DB)$. To select k features, the k

terms providing the highest values of $(k \cdot contt, DB)$ are chosen. The idea is that terms that occur in almost any given document are not well suited to distinguish them. On the other hand, terms that occur in only a view documents are too special to be used for a general comparison. Though this technique is very simple, it achieves remarkable results and is applicable for text clustering since it does not rely on the existence of document classes like the following two approaches.

The following feature selection techniques are for text classification only since they weight the significance of each term for a given class. Afterwards the top k significant terms for each class are chosen as dimensions of the resulting feature space.

To decide the significance of a given term for a class, there exist multiple solutions. All of them begin with counting the number of documents of class c_i containing a particular term w_j denoted as $occ_{w_j}^{c_i}$. Example methods to calculate the significance of a term w_j with respect a class c_i are:

χ^2 statistic

This method measures the level of independence between a term t and a class c_i and is computable in quadratic time.

Definition 2.7 (χ^2 statistics)

Let DB a labelled text collection and let $C_i \subset DB$ be the documents of DB belonging to class c_i. Furthermore let $cont(t, S) = \{d \in S | t \in d\}$ denote the subset of set S that contains a term t and let $\overline{cont(t, S)}$ denote the $S \setminus cont(t, S)$. Then the χ^2 statistics for class C_i and term t is defined as follows:

$A = |cont(t, C_i)|$

$B = |\bigcup_{l \neq i} cont(t, C_l)|$

$C = |\overline{cont(t, C_i)}|$

$D = |\bigcup_{l \neq i} \overline{cont(t, C_l)}|$

$$\chi^2(t, C_i) = \frac{|DB|(A \cdot D - C \cdot B)^2}{(A+B)(B+D)(A+B) \cdot (C+D)}$$

Information Gain

The information gain is measure for the degree a given term is capable to distinguish the classes. It is based on the entropy as a measure of pureness with respect to set of classes C.

Definition 2.8 (entropy)

Let DB be a set of documents and let $C = \{C_1, ..C_m\}$ with $DB = \bigcup_{1 \leq i \leq m} C_i$ be a disjunctive partitioning of DB. Then the entropy of DB with respect to the partitioning C is defined as follows:

$$entropy(C) = -\sum_{i=1}^{m} Pr[C_i] \cdot \log PR[C_i]$$

where $Pr[C_i]$ denotes $\frac{|C_i|}{|DB|}$.

Definition 2.9 (Information Gain)

Let t be a term and let $cont(t, S) = \{d \in S | t \in d\}$ denote the subset of set S that contains a term t and let $\overline{cont(t, S)}$ denote $S \setminus cont(t, S)$. The information gain of t with respect to the disjunctive partitioning C is:

$$G_C(t) = entropy(C) - \frac{|cont(t, DB)|}{|DB|} \cdot entropy(cont(t, DB) - \frac{|\overline{cont(t, DB)}|}{|DB|} \cdot entropy(\overline{cont(t, DB)})$$

The idea of information gain is to split a given set according to the occurrence of a given term and afterwards compare the weighted average of the resulting subsets to the entropy of the unsplitted set. If the entropy in the subsets decreases significantly, the term provides a higher information gain and is better suited as a feature. We will need the information gain again in section 2.3 for constructing decision tree classifiers.

To conclude, there are multiple techniques for reducing the dimensionality of text feature spaces. However, to find a suitable document representation it is often necessary to still employ between 5,000 and 20,000 terms to achieve good results.

After introducing techniques of feature selection, we will turn to the feature values that are stored in a feature vector. The most common approach is to count the appearances of the terms within a document to determine the feature value. This approach is called term frequency (TF) and is denoted by $n(t, d)$. However, for text collections containing documents that strongly vary in size, the appearance of a word should be measured by the length of the document. A word appearing in a small document should have more impact to the document length than the same word appearing in a large document containing a large number of words. Therefore, the document vector is often normalized with respect to the document length $f = \frac{n(t,d)}{\sum_{w \in d} n(w,d)}$.

Another commonly used technique is to additionally consider the inverse document frequency (IDF) of a term. The IDF factor is defined as follows:

$$IDF(t) = \frac{|DB|}{|cont(t,DB)|}$$

where DB is a data collection and $cont(t, DB)$ denotes the set of elements in DB that contains a given term t. The idea of IDF is to treat each dimension with a different weight. If a term appears very often, its weight is reduced. This way, terms that appear very often in all kinds of documents are considered as less important than terms appearing in specific documents only. Often TF and IDF are used together. The TDIDF score of a term appearance in a document is calculated by:

$$TDIDF_D(t) = TF_D(t) \cdot IDF(t)$$

Though text documents are reduced to feature vectors, comparing those vectors using the established l_p distance measures does not perform very well.

The sparseness of feature spaces has the effect that the feature values in the majority of the dimension is equally set to zero. Thus, l_p distance measures tend to consider the distance between most documents as very small. However, this effect is not wanted, since two documents should be compared by counting the words that appear in both of them. Therefore, there exists several special distance function that are suitable for sparse vectors and especially for text documents. [Dun03] lists some of the most common distance functions for text documents. The most established distance function is the cosine coefficient which is defined as follows:

Definition 2.10 (cosine coefficient)

$$dist_{cos} : I\!R^d \times I\!R^d \rightarrow I\!R : dist_{cos}(x, y) = 1 - \frac{\sum_{i=1}^{d} x_i \cdot y_i}{\sqrt{\|x\|} \cdot \sqrt{\|y\|}}$$

The idea of the cosine coefficient is to consider the sum of the products of the term frequencies of each document. This way, only words that occur in both documents are accounted for an increasing similarity. To consider the document size, the product is divided through the product of the length of each feature vector. Thus, the fraction has a value between one and zero. To turn the similarity measure to a distance function, the fraction is subtracted from one. The cosine coefficient is metric and has proven to be a suitable distance for many text mining application like clustering and classification.

2.2 Clustering Algorithms

Clustering or cluster analysis is one of the data mining tasks in this thesis. Therefore, this section provides a brief review about the purpose of clustering, a categorization of directions in clustering, and a discussion of the density-based clustering algorithms.

2.2.1 The Task of Clustering

Clustering is the process of grouping a given data set into classes or clusters. The objects within a cluster should be similar to each other and the objects of different clusters should be dissimilar to each other. Thus, similarity is a very important aspect for clustering objects. In most cases, clustering relies on distance based similarity as is described in the previous section. There are multiple algorithms for clustering data and the choice of a suitable one depends on both, the type of data and the application. An overview about clustering algorithms can be found in [HK01].

2.2.2 Directions of Clustering Algorithms

Over the last decades the research community proposed a variety of clustering algorithms which can be categorized with respect to varying aspects. In the following, we will use the categorization provided by [HK01].

1. **Partitioning Methods**

   Partitioning clustering algorithms partition a database of n objects into k disjunctive clusters. Each cluster contains at least one object and each object belongs to exactly one cluster. In order to find a good clustering, partitioning methods divide the data set into k initial partitions and afterwards optimize the cluster quality in several iterations. In each iteration some objects are moved from one cluster to another one, improving the quality of the clusterig. If it is not possible to augment the quality by relocating any object, the algorithm terminates. Note that partitioning clustering algorithms are often heuristic and usually not guarantee to find the clustering having the maximal quality. The most prominent examples of this direction are k-Means clustering and k-Medoid methods like PAM or CLARANS [HK01]. Partitioning clustering algorithms are well suited for small to medium sized databases and are useful to find k spherical clusters to describe a data set.

However, partitioning clustering algorithms need a predefined number of clusters to be found and do not find arbitrary shaped clusters in large databases without extension.

2. **Hierarchical Methods**
   Hierarchical methods create a hierarchical decomposition of the data set. A hierarchical clustering allows that smaller clusters are part of bigger clusters which are more general. There are two approaches to hierarchical clustering, agglomerative and divisive clustering. Agglomerative clustering follows a bottom-up approach. Each object is a cluster of its own. In the next step, the two clusters are merged that are closest to each other. The merging of clusters is repeated until either a complete tree of clusters is built or a termination condition is reached. This cluster tree is called dendrogram and contains clusters of varying size. Divisive methods follow a top-down approach and successively divide already found clusters into smaller clusters. The clustering terminates if each cluster consists of exactly one object or a termination condition is reached. In basic hierarchical clustering algorithms the decision that an object belongs to a cluster cannot be revised. Though this characteristic helps to find clusters efficiently, the quality of the clustering might suffer. Thus, there are more sophisticated approaches that analyze the object linkages like in CURE [GRS98] or Chameleon [KHK99] or additionally use iterative relocation like BIRCH [ZRL96].

3. **Density-Based Methods**
   Density-based clustering algorithms define clusters according to the dense areas in the database. They are capable of finding an arbitrary number of arbitrary shaped clusters. Another advantage of density-based clustering is its capability to consider noise which is not to be counted to any cluster. In order to identify dense areas in the database, density-based algorithm employ ε-range queries to decide if the ε-neighborhood of a data object contains a sufficient number of

other data objects. Clusters are defined by connected areas where the data objects contain objects with dense ε-neighborhoods. In the next section, we will have a closer look at the most prominent examples for density-based clustering algorithms DBSCAN [EKSX96] and OPTICS [ABKS99].

4. **Grid-Based Methods**
   Grid-based methods divide the object space into a finite number of cells that form a grid structure. All clustering operations are performed on this grid structure. The advantage of grid-based clustering is the fast computation. The drawback of these approaches is that the result is strongly dependent on the size and placement of the grid. Important examples for grid-based clustering algorithms are Sting [WYM97] and CLIQUE[AGGR98].

5. **Model-Based Methods**
   This approach is based on the construction of a (statistical) model that explains the data distribution in a given data set as good as possible. The common way to construct such a model is using a mixture of several density functions, containing a density function for each Cluster. Often an additional uniform distributed density function is added to consider noise objects. A further extension determines the most likely number of clusters in the database. Examples for model based clustering algorithms are COBWEB [Fis87] and expectation maximization clustering(EM Clustering) [DLR77]. Though model-based clustering algorithms create very expressive models, their computational costs are often very high. Thus, model-based clustering algorithms are usually not applicable to very large databases.

Though the given categorization surveys the most important directions of clustering, many clustering algorithms employ ideas of more than one of the mentioned categories. For example, the density-based algorithm OPTICS

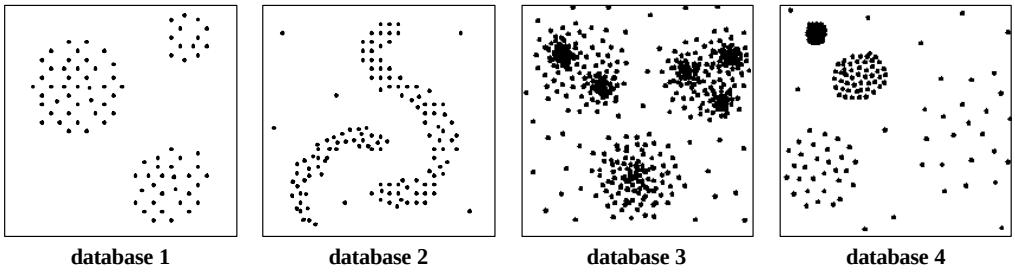


Figure 2.1: Sample databases.

provides a cluster hierarchy as a result and EM Clustering has a similar algorithmic scheme as partitioning clustering algorithms, e.g. k-Means.

2.2.3 Density-Based Clustering

Density-based clustering algorithms are the foundation of the clustering solutions that are proposed in this thesis. Additionally, to the advantages of density-based clustering named above, this approach is very general and therefore well suited for clustering complex and compound objects. As mentioned above, the basic idea of density-based clustering is the observation that inside a cluster the density of points is considerably higher than outside a cluster. Furthermore, different clusters are separated by areas of noise. This observation can be validated by looking at the two dimensional sample databases displayed in figure 2.1).

DBSCAN

The key idea of density-based clustering is that for each member of a cluster the neighborhood of a given radius has to contain a minimum number of objects, i.e. the density in the neighborhood has to exceed a density threshold. This threshold is determined by two user defined input parameters ε specifying the size of the neighborhood and *MinPts* specifying the minimum number of objects the neighborhood must contain.

Definition 2.11 (ε-neighborhood)

Let $\varepsilon \in \mathbb{R}$. The ε-neighborhood *of an object $p \in \mathcal{D}$, denoted by $\mathcal{N}_\varepsilon(p)$, is defined by*

$$\mathcal{N}_\varepsilon(p) = \{o \in \mathcal{D} \mid dist(p, o) \leq \varepsilon\}$$

As claimed above, an object should be inside a cluster if its neighborhood contains at least a given number of objects.

Definition 2.12 (core object)

An object $q \in \mathcal{D}$ is a core object *w.r.t. $\varepsilon \in \mathbb{R}$ and $MinPts \in \mathbb{N}$, denoted by $\textsc{Core}_{den}(q)$, if its ε-neighborhood contains at least MinPts points, formally:*

$$\textsc{Core}_{den}(q) \Leftrightarrow |\mathcal{N}_\varepsilon(p)| \geq MinPts$$

Let us note that the acronym *den* in the definition refers to the density parameters ε and *MinPts*. In the following, we omit the parameters ε and *MinPts* wherever the context is clear and use *den* instead. The core point concept is visualized in Figure 2.2(a).

A naive approach could require the core object property for each member of a cluster. However, this approach fails because there are some objects on the border of the cluster (- so called *border objects*) that do not fit the core point property but are intuitively part of a cluster. In fact, a cluster has two properties: density and connectivity. The first one is captured through the core object property. The second one is captured through the following concepts.

Definition 2.13 (direct density-reachable)

An object $p \in \mathcal{D}$ is direct density reachable *w.r.t. $\varepsilon \in \mathbb{R}$ and $MinPts \in \mathbb{N}$ from $q \in \mathcal{D}$, denoted by $\textsc{DirReach}_{den}(q,p)$, if q is a core object and p is in the ε-neighborhood of q, formally:*

$$\textsc{DirReach}_{den}(q, p) \Leftrightarrow \textsc{Core}_{den}(q) \ \wedge \ p \in \mathcal{N}_\varepsilon(q).$$

The concept of direct density reachability is depicted in Figure 2.2(b). Obviously, directly density reachable is a symmetric relation for pairs of core objects. However, it is not symmetric in general.

Definition 2.14 (density-reachable)
An object $p \in \mathcal{D}$ is density-reachable *from $q \in \mathcal{D}$ w.r.t. $\varepsilon \in \mathbb{R}$ and MinPts $\in \mathbb{N}$, denoted by* $\text{REACH}_{den}(q,p)$*, if there is a chain of objects $p_1, \ldots, p_n \in \mathcal{D}$, $p_1 = q$, $p_n = p$ such that p_{i+1} is directly density reachable from p_i, formally:*

$$\begin{aligned}&\text{REACH}_{den}(q,p) \Leftrightarrow \\ &\quad \exists p_1, \ldots, p_n \in \mathcal{D} : p_1 = q \ \wedge \ p_n = p \ \wedge \\ &\quad \forall i \in \{1, \ldots, n-1\} : \text{DIRREACH}_{den}(p_i, p_{i+1}).\end{aligned}$$

Density reachability is illustrated in Figure 2.2(c). It is the transitive enclosure of direct density reachable but it is not symmetric in general (again only for pairs of core points). Thus, we have captured the connectivity of core points so far. But two border objects of the same cluster $\mathcal{C}$ are not density reachable from each other. However, there must be a core point in $\mathcal{C}$ from which both border points are reachable. Therefore, the following definition captures general connectivity of points within a cluster.

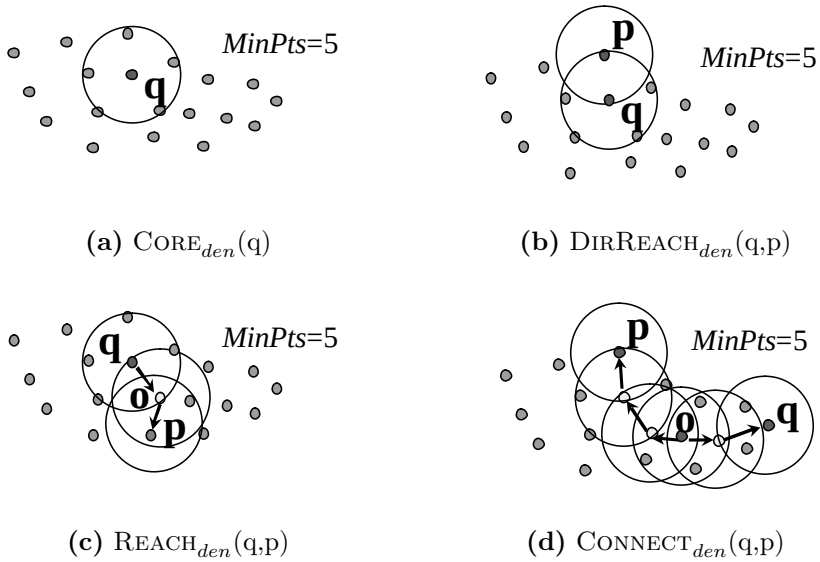
Definition 2.15 (density-connected)
An object $q \in \mathcal{D}$ is density-connected *to another object $p \in \mathcal{D}$ w.r.t. $\varepsilon \in \mathbb{R}$ and MinPts $\in \mathbb{N}$, denoted by* $\text{CONNECT}_{den}(q,p)$*, if there is an object $o \in \mathcal{D}$ such that both p and q are density reachable from o, formally:*

$$\begin{aligned}&\text{CONNECT}_{den}(q,p) \Leftrightarrow \\ &\exists o \in \mathcal{D} : \text{REACH}_{den}(o,q) \ \wedge \ \text{REACH}_{den}(o,p).\end{aligned}$$

Density connected is in general a symmetric relation. The concept is visualized in Figure 2.2(d).

Now, the density-based notion of a cluster can be defined using the introduced concepts. Intuitively, a cluster is defined to be a set of density connected objects which is maximal w.r.t. density- reachability. The objects in $\mathcal{D}$, not belonging to any of its density-connected sets are defined as *noise*.

Definition 2.16 (density-connected cluster)
A non-empty subset $\mathcal{C} \subseteq \mathcal{D}$ is called a density connected cluster *w.r.t. $\varepsilon \in \mathbb{R}$*



(a) CORE$_{den}$(q) **(b)** DIRREACH$_{den}$(q,p)

(c) REACH$_{den}$(q,p) **(d)** CONNECT$_{den}$(q,p)

Figure 2.2: Illustration of density-based clustering concepts

and MinPts $\in \mathbb{N}$*, denoted by* CONSET$_{den}(\mathcal{C})$*, if* $\mathcal{C}$ *is a density connected set and* $\mathcal{C}$ *is maximal w.r.t. density-reachability, formally:*

CONSET$_{den}(\mathcal{C}) \Leftrightarrow$

(1) *Connectivity:* CONSET$_{den}(\mathcal{C})$

(2) *Maximality:* $\forall p, q \in \mathcal{D} : q \in \mathcal{C} \wedge \text{REACH}_{den}(q, p) \Rightarrow p \in \mathcal{C}$.

The algorithm DBSCAN is proposed in [EKSX96] and computes all density-based clusters w.r.t. the user-specified parameters ε and *MinPts* by one single pass over the data. For that purpose, DBSCAN uses the fact that a density connected set can be detected by finding one of its core objects p and computing all objects which are density reachable from p. The pseudo code of DBSCAN is depicted in Figure 2.3. The method ExpandCluster which computes the density connected cluster, starting from a given core point, is given in Figure 2.4.

The correctness of DBSCAN can be formally proven (cf. Lemmata 1 and 2 in [EKSX96], proofs in [SEKX98]). Although DBSCAN is not in a strong sense deterministic (the run of the algorithm depends on the order in which

```
algorithm DBSCAN(SetOfObjects 𝒟, Real ε, Integer MinPts)
  // each point in 𝒟 is marked as unclassified
  generate new clusterID cid;
  for each p ∈ 𝒟 do
    if p.clusterID = UNCLASSIFIED then
      if ExpandCluster(𝒟, p, cid, ε, MinPts) then
        cid := cid + 1
      end if
    end if
  end for
```

Figure 2.3: The DBSCAN algorithm.

the points are stored), both the run-time as well as the result (number of detected clusters and association of core objects to clusters) are determinate. The worst case time complexity of DBSCAN is $O(n \log n)$ assuming an efficient spatial index (e.g. [BKK96] or [BBJ+00]) and $O(n^2)$ if no index exists.

OPTICS

DBSCAN computes a flat density-based decomposition of a database. It detects each density connected set w.r.t. a global density parameter specified by ε and *MinPts*. However, there may be clusters of different density and/or nested clusters in the database (cf. "database 3" and "database 4" in Figure 2.1). If the densities of different clusters vary significantly, the parameterization of DBSCAN is problematic. A less strict density threshold would detect also the clusters of lower density but may merge clusters of higher density. On the other hand, a more strict density threshold would partition the denser clusters but would miss clusters with lower density. In addition, the information of nested clusters, i.e. denser clusters within less dense clusters, may be missed.

In [ABKS99] the density connected clustering notion is extended by hierarchical concepts. Based on these concepts, the algorithm OPTICS is presented. The key idea is that (for a constant *MinPts*-value) density-based clusters w.r.t. a higher density, i.e. a lower value for ε, are completely con-

```
boolean ExpandCluster(SetOfObjects 𝒟, Object start, Integer cid, Real ε, Integer MinPts)
  SetOfObjects seeds := 𝒩_ε(start);
  if |seeds| < MinPts then
    start.clusterID := NOISE;
    return false;
  end if
  for each o ∈ seeds do
    o.clusterID := cid;
  end for
  remove start from seeds;
  while seeds ≠ ∅ do
    o := first point in seeds;
    neighbors := 𝒩_ε(o);
    if |neighbors| ≥ MinPts then
      for each p ∈ neighbors do
        if p.clusterID ∈ {UNCLASSIFIED, NOISE} then
          if p.clusterID = UNCLASSIFIED then
            insert p into seeds;
          endif
        p.clusterID := cid;
        endif
      end for
    end if
    remove o from seeds;
  end while
  return true;
```

Figure 2.4: Method ExpandCluster.

tained in density-based clusters w.r.t. a lower density, i.e. a higher value for ε. Figure 2.5 illustrates this observation: C_1 and C_2 are density-based clusters w.r.t. $eps_1 < eps_2$ and C is a density-based cluster w.r.t. eps_2 completely containing C_1 and C_2.

The algorithm OPTICS extends DBSCAN by computing the density connected clusters w.r.t. all parameters ε_i that are smaller than a generic value ε. In contrast to DBSCAN, OPTICS does not assign objects to clusters, but stores the order in which the data objects are processed and the information which would be used by an extended DBSCAN algorithm to assign objects to clusters. This information consists of only two values for each object, the *core distance* and the *reachability distance*.

The core distance is based on the concept of k-nearest neighbor distances.

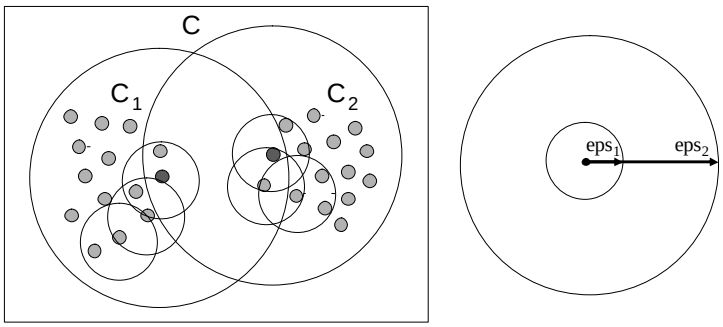


Figure 2.5: Nested clusters of different density.

Definition 2.17 (k-nearest neighbor distance)
The k-nearest neighbor distance *of* p*, denoted by* $nn\text{-}dist_k(p)$*, is defined as follows:*

$$nn\text{-}dist_k(p) = \max\{dist(o,p) \mid o \in NN_k(p)\}.$$

Let us note that in Definition 2.17 it is implicitly assumed that $\mathcal{D}$ contains at least k elements, i.e. $k \leq n$.

Definition 2.18 (core distance)
The core distance *of an object* $q \in \mathcal{D}$ *w.r.t.* $\varepsilon \in \mathbb{R}$ *and* $MinPts \in \mathbb{N}$ *is defined as*

$$\textsc{CoreDist}_{den}(q) = \begin{cases} nn\text{-}dist_{MinPts}(q) & \text{if } |\mathcal{N}_\varepsilon(q)| \geq MinPts \\ \infty & \text{else.} \end{cases}$$

The core distance of an object q is the smallest threshold $\hat{\varepsilon} \leq \varepsilon$ such that q is a core object w.r.t. $\hat{\varepsilon}$ and *MinPts*. If $\hat{\varepsilon}$ would be greater than the generic ε value, the core distance of q is set to ∞.

Definition 2.19 (reachability distance)
The reachability distance *of a point* $p \in \mathcal{D}$ *relative from another object* $q \in \mathcal{D}$ *w.r.t.* $\varepsilon \in \mathbb{R}$ *and* $MinPts \in \mathbb{N}$ *is defined as*

$$\textsc{ReachDist}_{den}(q,p) = \max(\textsc{CoreDist}_{den}(q), dist(q,p)).$$

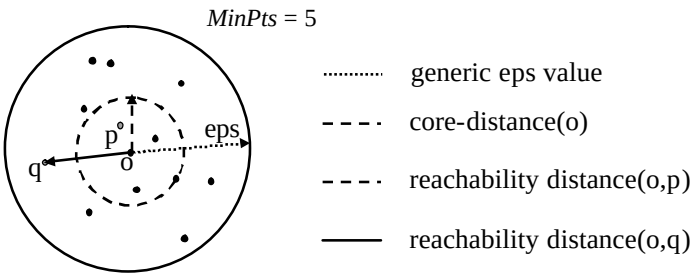


Figure 2.6: Illustration of core distance and reachability distance.

The reachability distance of an object p w.r.t. another object q is the smallest threshold $\hat{\varepsilon} \leq \varepsilon$ such that p is directly density reachable from q. Obviously, to achieve this relation, q has to be a core object. Thus, the reachability distance cannot be smaller than the core distance of q. As a consequence, if $dist(q,p) \leq \text{CoreDist}_{den}(q)$, the reachability distance of p w.r.t. q is set to $\text{CoreDist}_{den}(\text{q})$. Otherwise, the smallest threshold $\hat{\varepsilon} \leq \varepsilon$, where p is directly density reachable from q, is exactly $dist(q,p)$. Let us note that if q is not a core point w.r.t. the generic ε-value, i.e. $\text{CoreDist}_{den}(q) = \infty$, we get $\text{ReachDist}_{den}(q,p) = \infty$ indicating that the smallest threshold $\hat{\varepsilon}$ is in fact greater than ε, i.e. p cannot be directly reached from q w.r.t. the generic threshold ε.

Both the core distance of an object o and the reachability distances of the objects p and q relative to o are illustrated in Figure 2.6.

The OPTICS algorithm computes a so-called *cluster ordering* of a database w.r.t. the two input parameters ε and *MinPts*. In addition, the core distance and a "suitable" reachability distance is stored for each object. The pseudo code of the OPTICS algorithm is depicted in Figure 2.7. It starts with an arbitrary object $o \in \mathcal{D}$, assigns a reachability distance of ∞ to o and expands the cluster order if the core distance of o is smaller than the generic input parameter ε. The expansion is done by inserting each object $p \in \mathcal{N}_\varepsilon(o)$ into a seed list OrderedSeeds. This seed list is organized as a heap, storing that object q, having the minimum reachability distance to the already processed objects as first object in the list. The heap structure is maintained by the

```
algorithm OPTICS(SetOfObjects 𝒟, Real ε, Integer MinPts)
  CO := empty cluster ordering;
  while |CO| < n do
    o := arbitrary not yet handled point in 𝒟;
    neighbors_o := 𝒩_ε(o);
    o.R := ∞;
    o.C := CORE_den(o);
    mark o as handled;
    append o to CO;
    if o.C ≠ ∞ then
      OrderedSeeds.update(neighbors_o, o);
      while OrderedSeeds ≠ ∅ do
        p := OrderedSeeds.first();
        neighbors_p := 𝒩_ε(p);
        p.C := CORE_den(p);
        mark p as handled;
        append p to CO;
        if p.C ≠ ∞ then
          OrderedSeeds.update(neighbors_p, p);
        end if
      end while
    end if
  end while
```

Figure 2.7: The OPTICS algorithm.

procedure OrderedSeeds::update (cf. Figure 2.8) which updates the reachability distances of the objects that are already in the seed list if their according values decrease. The next object to be inserted in the cluster ordering is always the first object in the seed list. If the core distance of this object is smaller or equal to ε, all objects in the ε-neighborhood are again inserted into or updated in the seed list. If the seed list is empty and there are still some not yet processed objects in $\mathcal{D}$, we have a so-called "jump". OPTICS selects another arbitrary not yet handled object in $\mathcal{D}$ and proceeds extending the cluster ordering for the remaining objects.

Definition 2.20 (cluster ordering)

Let $MinPts \in \mathbb{N}$, $\varepsilon \in \mathbb{R}$, and CO be a permutation of the objects in $\mathcal{D}$. Each $o \in \mathcal{D}$ has additional attributes $o.P$, $o.C$ and $o.R$, where $o.P \in \{1, \ldots, n\}$ symbolizes the position of o in CO. We call CO a cluster ordering *w.r.t. ε*

```
method OrderedSeeds::update(SetOfObjects neighbors, Object center)
  cdist := center.C;
  for each o ∈ neighbors do
    if o is not yet processed then
      rdist := max{cdist, dist(o, center)};
      if o is already in OrderedSeeds then
        if o.R > rdist then
          o.R := rdist;
          decrease(o);
        end if
      else
        o.R := rdist;
        insert(o);
      end if
    end if
  end for
```

Figure 2.8: Method OrderedSeeds::update.

and MinPts if the following three conditions hold:

(1) $\forall p \in CO : p.C = \text{CoreDist}_{den}(p)$

(2) $\forall x, y \in CO : 1 < x.P < y.P \Rightarrow$

$$\exists o \in CO : o.P < x.p \wedge \text{ReachDist}_{den}(o, x) \leq \text{Reach}_{den}(o, y)$$

(3) $\forall p \in CO :$

$$p.R = \min\{\text{ReachDist}_{den}(o, p) \mid o \in CO \wedge o.P < p.P\},$$

where $\min \emptyset = \infty$

Condition (2) states that the cluster ordering CO is built by selecting the object o for position i that yields the smallest reachability to each of the objects that are already ordered so far, i.e. all objects in position $j < i$. $o.C$ symbolizes the core distance of an object o in CO whereas $o.R$ is the reachability distance assigned to the object o during the generation of CO.

A cluster ordering contains sufficient information to extract all density-based clusterings w.r.t. any $\varepsilon' \leq \varepsilon$. The density-based clustering w.r.t. a particular $\varepsilon' \leq \varepsilon$ can be extracted by scanning the cluster ordering and checking the reachability distance and the core distance of each object. If

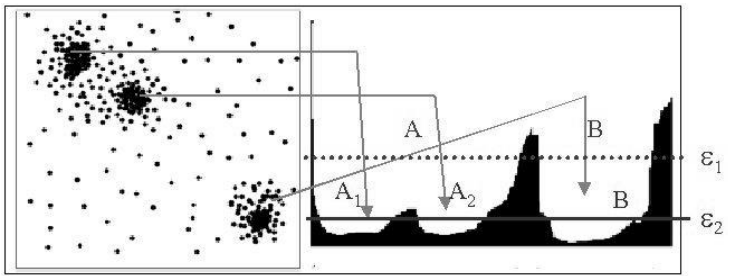


Figure 2.9: Reachability plot (right) computed by OPTICS for a sample two dimensional data set (left).

the reachability distance of the current object is larger than ε', we have to check its core distance. If the core distance of this object is also larger than ε', this object is assigned to noise. Ohterwise, the object is a core object and we start a new cluster. If the reachability of the current object is smaller than ε', it can be assigned to the current cluster because it is density reachable from a preceding core object in the cluster ordering. Let us note that the resulting clusters may miss some border objects, because border objects can belong to multiple density connected clusters.

A breakthrough advantage of OPTICS is that the resulting cluster ordering can be visualized very intuitively using a so-called *reachability plot*. A reachability plot is a two dimensional visualization of a cluster ordering where the objects are plotted according to the sequence specified in the cluster ordering along the x-axis and the y-values are given by the reachability distances. Figure 2.9 (right) depicts the reachability plot based on the cluster ordering computed by OPTICS for the sample two dimensional data set in Figure 2.9 (left). Intuitively, clusters are "valleys" or "dents" in the plot because sets of consecutive objects with a lower reachability value are packed more densely. In particular, to manually obtain a density-based clustering w.r.t. any $\varepsilon' \leq \varepsilon$ by visual analysis, one simply has to cut the reachability plot at y-level ε' i.e. parallel to the x-axis. The consecutive valleys in the plot below this cutting line contain the according clusters. An example is presented in Figure 2.9: For a cut at the level ε_1, we find two clusters de-

noted as A and B. Compared to this clustering, a cut at level ε_2 would yield three clusters. The cluster A is split into two smaller clusters denoted by A_1 and A_2 and cluster B decreased its size. This illustrates how the hierarchical cluster structure of a database is revealed at a glance and could be easily explored by visual inspection.

2.3 Classification Algorithms

This section contains a brief introduction to classification which is the other data mining task that is discussed in this thesis. First of all classification will be described in general and methods to evaluate the quality of a classifier are discussed. Afterwards main directions of classification are surveyed, including the classification methods used as foundation for the novel techniques of this thesis.

2.3.1 General Aspects of Classification

The task of classification is to learn a function that maps data objects to their correct class(es) in a predefined class set. A classifier learns from a so-called training set, containing a sufficient number of already mapped objects for each class. The training objects are considered to be "labelled" with the name of the class they belong to. Classification is also called supervised learning because it is directed by these labelled objects. Formally, a classifier is a function of the following form:

Definition 2.21 (Classifier)
Let O be the set of objects, C the set of classes and let $G : O \rightarrow C$ be the true mapping of all objects o to their correct class c_o. Furthermore, let $T \subset O \times C$ with $T = \{(o, c) | o \in T_o \subset O \wedge G(o) = c\}$ be the set of already labelled training objects. Then a classifier is a function $CL_T : O \rightarrow C$ that maps the objects of O to a class $c_i \in O$.

G is also called the ground truth. The goal of classification is to train F_T in a way that CL_T can reproduce the ground truth G as good as possible.

One of the most important aspects for evaluating classifiers is that the quality of prediction for the objects $o_i \in T_o$ is not significant for the performance observed for the objects $o_j \in O \setminus T_o$. Since the correct class for the objects in T_o is already known, it is easy to find a classifier that achieves

maximum performance on the training data. However, for reliable class predictions on unknown data objects $o \in O \setminus T_o$ a classifier has to offer good generalization. The key to build up a good classifier is to find out which of the characteristics are significant for a class and which are typical to individual data objects. If the classifier is based on too many individual characteristics, it will fit too accurately to the elements of T_o and the performance for new data objects $o \in O \setminus T_o$ degenerates. This effect is known as *overfitting* and is one of the central problems of classification. A good theoretical description to this problem is found in the introduction of [Bur98].

To measure classification performance without overfitting, the set of objects where the correct class is already known T_o, is split into a training set TR and a test set TE. The training set is used to train the classifier. Afterwards the elements of the test set are classified and the following measures for the classification performance can be calculated:

- **Classification Accuracy**

$$Acc(F_{TR}) = \frac{|\{o|G(o) = F_{TR}(o) \wedge o \in TE\}|}{|TE|}$$

- **Precision**

$$Precision(F_{TR}, c) = \frac{|\{o|G(o) = F_{TR}(o) = c \wedge o \in TE\}|}{|\{o|CL_T(o) = c\}|}$$

- **Recall**

$$Recall(F_{TR}, c) = \frac{|\{o|G(o) = F_{TR}(o) = c \wedge o \in TE\}|}{|\{o|G_T(o) = c\}|}$$

- **F-Measure**

$$F - Measure(F_{TR}, c) = \frac{2 \cdot Precision(F_{TR}, c) \cdot Recall(F_{TR}, c)}{Precision(F_{TR}, c) + Recall(F_{TR}, c)}$$

The classification accuracy is a performance measure considering all classes. It is the percentage of correct predictions for the test set TE. However, if the number of test objects for each class varies very strongly, considering the accuracy tends to be misleading. Consider a test set for two classes A and B that consists of 95 % objects for class A and only 5% percent of the objects belong to class B. By always predicting class A, it is possible to achieve 95 % classification accuracy for this test set without having a reasonable classifier. This example illustrates that considering the accuracy is only advisable if the number of test objects in TE is approximately similar for each of the classes. Additionally to the accuracy, the most important measures for classification performance are precision and recall. The precision for a class c indicates the percentage of correct classified objects among the objects that where predicted to belong to class c. The recall for class c is the percentage of correctly classified objects among all objects that really belong to class c. Naturally, there is a trade-off between precision and recall. Most classifiers can be adjusted to increase the precision of a class c while decreasing its recall or the other way around. To have a measure considering both aspects, the f-measure was introduced. The f-measure is the harmonic mean value of precision and recall and reaches its maximum when both measures reach the same value.

Another problem of testing classifiers is that the set of already labelled instances T usually tends to be very limited. Thus, training the classifier with only a subset of T will decrease the classification performance. On the other hand, it is not possible to measure the classification performance correctly without already labelled data that are not part of the training set. To limit this problem, the technique of stratified k-fold cross validation was introduced. First of all, stratified k-fold cross validation divides T into k stratified folds. Each stratified fold contains approximately the same percentage of objects from each of the classes as it is found in the complete set T. Now the classifier is trained k times with $k-1$ folds and each time another fold is left out for testing. Thus, for each run and for each fold there is an

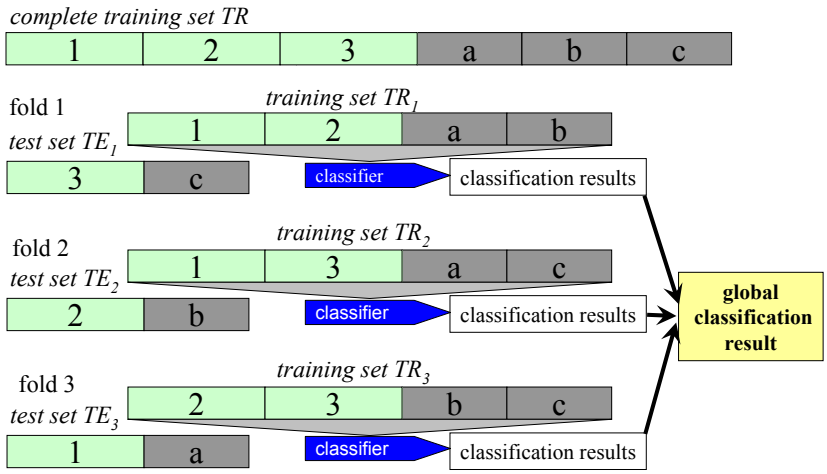


Figure 2.10: Illustration for 3-fold cross validation.

classification result that can be used to calculate the introduced performance measures for the complete data set T. Figure 2.10 illustrates the building of stratified folds.

Additional to the quality of prediction, there are other important aspects of a classifier. Especially for database application efficiency is an important aspect. The efficiency of a classifier is measured by the time it takes to classify a new unlabelled object. This so-called classification time is important, since a classifier is considered to be applied to large numbers of objects, without being modified. The time that is spent for the training of a classifier, the so-called training time, is considered as less important in most cases, because the training set is considered to consist of a minor number of objects only. However, if the training of a classifier is very time consuming, the complete KDD process is slowed down. Therfore, the time to train a classifier has to be considered for several application as well. The last important aspect of classification is the understandability of the found class model. Though accurate classification is the primary goal of classification, for many applications explicit knowledge about characteristics of the treated classes are needed. Thus, providing class models that are easily understood by human users is

an important feature of a classifier. Unfortunately, many of the established methods do not provide this feature.

2.3.2 Important Directions of Classification

The following section gives a brief overview of established methods of classification. Though the methods to achieve classification are strongly varying, all classifiers have something in common. They divide the given object space into disjunctive sections that are associated to a given class.

In the following, the most important approaches to classification are surveyed:

Bayes Classifiers

Statistical or Baysian classifiers are based on the assumption that the objects of a class can be modelled by a statistical process. Each data object o has its origin in the process of a given class c_i with a certain probability $Pr[o|c_i]$ and each process of a class c_i generates objects with a certain probability $Pr[c_i]$ called *a priori probability*. To decide which class is to be predicted for object o, it is necessary to determine the probability $Pr[c_i|o]$ called *a posteriori probability*. It describes the probability that an object o has its origin in class c_i. To determine the a posteriori probability based on $Pr[c_i]$ and $Pr[o|c_i]$, the rule of Bayes is used:

Definition 2.22 (Rule of Bayes)
Let $o \in O$ be an object and let $c_i \in C$ be a class. Then the a posteriori probability $Pr[c_i|o]$ can be calculated by:

$$Pr[c_i|o] = \frac{Pr[c_i] \cdot Pr[o|c_i]}{\sum_{j=1}^{|C|} Pr[c_j] \cdot Pr[o|c_j]}$$

where $Pr[c_i]$ denotes the a priori probability of class c_i and $Pr[o|c_i]$ denotes the probability that o was generated by the model of class c_i.

The class providing the maximum likelihood for an object o is predicted. Formally the decision rule of this so-called maximum likelihood classifier is the following:

$$argmax_{c_i \in C} Pr[c_i] \cdot Pr[o|c_i]$$

Note that the denominator is equal for each of the classes and can therefore be neglected when calculating the most likely class.

This method of classification is optimal since no other classifier can achieve a better average accuracy using the same a priori knowledge.

The problem of maximum likelihood classification is to find a proper statistical process to describe the data objects of a class. Established methods for calculating $Pr[o|c_i]$ are:

- **Naive Bayes Classification**

  This is the most established way to calculate $Pr[o|c_i]$ for vector spaces. An object o is described by a vector $(o_1, \ldots, o_d)$. The classifier is called naive because it assumes that each dimension is independently distributed for each class. Thus,

  $$Pr[o|c_i] = \prod_{j=1}^{d} Pr[o_j|c_i]$$

  describes the probability for an object o and the class c_i. The distribution within each dimension can be chosen freely. Commonly used examples are the Gauss distribution or the multi-nomial distribution. Naive Bayes offers good accuracy for many application even, if the assumption of independent dimension does not hold.

- **Markov Processes**

  A Markov model is a probabilistic Moore automata that is often used to model sequential data. A first order Markov model is given by a triple (S, T, π). S is a set of states corresponding to discrete number elements of the modelled sequence. T is a $|S| \times |S|$ matrix containing the

transition probabilities for each pair of states. π is the start distribution indicating the probability that a sequence starts with state s_i for all states in S. Thus, a first order Markov model models the sequential character of an input sequence by considering the previous state for each transition. Markov models of order k consider the k last steps when calculating the probability of a new state. A Markov model with $k = 0$ models the probability of a multiset of states without any sequential character.

Other important methods for calculating $Pr[o|c_i]$ are Baysian belief networks [HK01] that model the dependency of certain dimensions in a vector space and general multi-dimensional Gauss distribution that are capable to consider any probability. Generally, any stochastic process generating valid data objects can be used to construct a Bayes classifier as long as its parameters can be determined with statistic methods.

The performance of a Bayes classifier is strongly dependent on the used statistical process. If a good model is found for all of the classes, Bayes classifiers offer accurate prediction. To explain the found knowledge a user can analyze the statistical model of each class. For example, the mean value and the variance of a multi-dimensional Gauss distribution might yield important insight into the structure of a class. The classification using Bayes classifiers is usually very fast because the calculation of $Pr[o|c_i]$ can be done very efficiently for most statistical processes.

k Nearest Neighbor (kNN) Classification

Nearest neighbor classifiers are based on the idea that an object should be predicted to belong to the same class as the objects in the training set with the biggest similarity. To use kNN classification, it is enough to have a suitable similarity search system in which the training data is stored. Classification is done by analyzing the results of a kNN query. The simplest way to determine a classification result of a kNN classifier is the majority rule. The

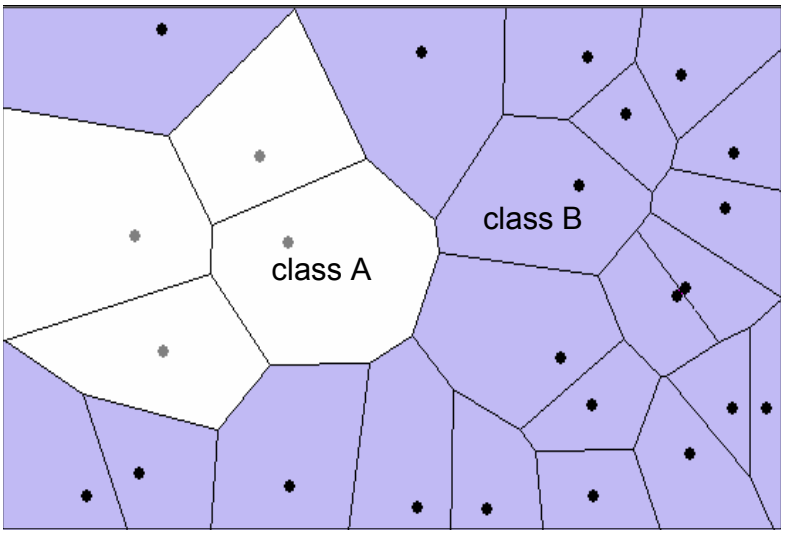


Figure 2.11: The Voronoi cells mark the class borders of an NN classifier in this two dimensional example.

objects in the query result are counted for each class and the class having the majority count is predicted to be the class of the object.

Another more sophisticated method is to consider the distances to the object to weight the impact of each neighboring object. Thus, a close object contributes more to the decision than an object having a large distance. This decision rule can be formulated as follows:

$$prediction(o) = argmax_{c \in C}(\sum_{q \in NN_k(o) \wedge G(q)=c} \frac{1}{d(o,q)^2})$$

kNN classifiers use the class borders given by the Voronoi cells of the objects within the training set and thus, do not need any training or model building. Figure 2.11 displays the Voronoi cells of a simply two dimensional example for a NN classifier. As a result, kNN classifiers cannot be used to gain explicit class knowledge to analyze the structure of the classes. kNN classification is also known as *lazy learning* or *case based reasoning*.

The parameter k is very important to the classification accuracy achieved by a kNN classifier. If k is chosen too small, classification tends to be very

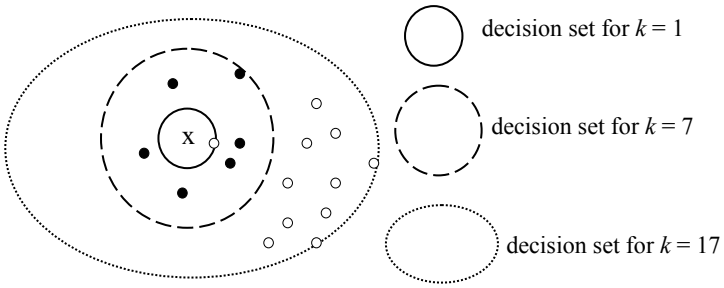


Figure 2.12: The figure illustrates the effect of 3 different values for k.

sensible to noise and outliers. On the other hand, a too large value for k might extend the result set of the k nearest neighbor by objects that are too far away too be similar to the classified object. Figure 2.12 illustrates the influence of the parameter k. For $k = 1$ the decision sphere is too small and for $k = 17$ the decision sphere is too big for a correct prediction. For $k = 7$ the object is classified correctly.

Since kNN classification works directly on the training data, the classification time is very dependent on the efficiency of the underlying similarity search system. Especially for the case of large training sets linear search becomes very inefficient. Using suitable index structures [CPZ97, LJF94, BKK96, BBJ+00] can offer a solution to this problem. However, for complex objects the usefulness of index structures is limited. Another approach to speed up kNN classification is to reduce the training set. This can be done by deleting unimportant objects as described in [BM02]. Another way to speed up classification is to build the centroid for the objects of each class and afterwards use only the centroids and NN classification. In [HK00] it was demonstrated that this rather simple approach still yields accurate classification for text data.

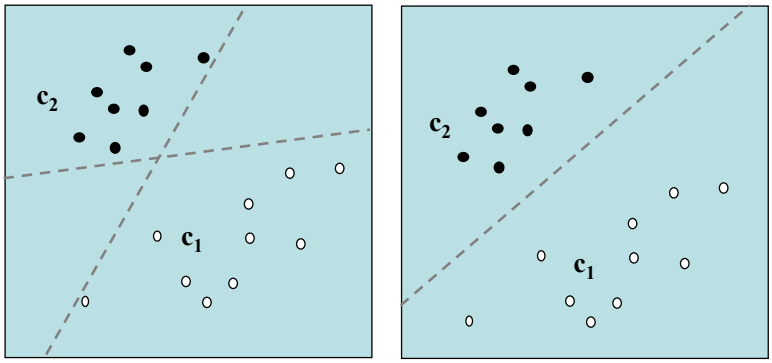


Figure 2.13: Arbitrary separating hyper planes(left). Maximum margin hyperplane (right).

Support Vector Machines

In [CV95] support vector machines (SVM) were introduced for the classification of feature vectors. Basic SVMs distinguish the objects of two classes by linear separation which is achieved by determining a separating hyperplane in the object space. The idea of SVMs is to find the hyperplane providing the maximum level of generalization and thus avoids overfitting as good as possible. In [CV95] it is shown that this most generalizing hyperplane is the hyperplane with a maximum margin between the classes. Figure 2.13 displays an illustration of a maximum margin hyperplane in a two dimensional example. The vectors in the training set having the minimal distance to the maximum margin hyperplane are called support vectors. The location of the maximum margin hyperplane does only depend on these support vectors and thus, the classifier was named support vector machine.

To determine the exact position of the maximum margin hyperplane and to find out the support vectors, a dual optimization problem is formulated which can be solved by algorithms like SMO [Pla98].

A major problem of linear separation is that there is not always a hyperplane that is able to separate all training instances. Therefore, two improvements for SVMs have been introduced that enable SVMs to separate almost

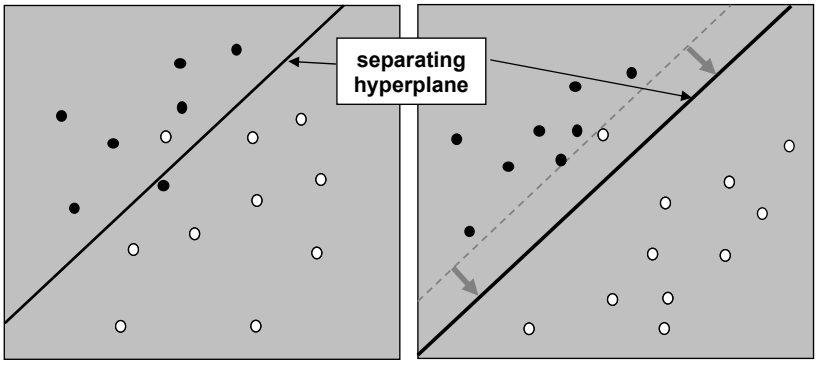
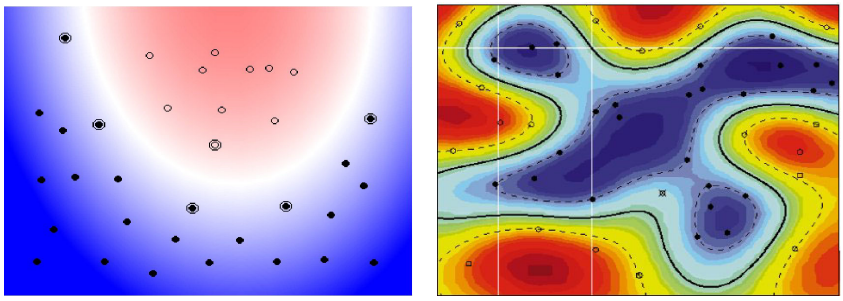


Figure 2.14: Example for a SVM using a soft margin for an inseparable data set(left).The dashed lined illustrates a strictly separating SVM(Right). The bold black line displays the more general SVM using soft margins.

any kind of data.

The first improvement is the introduction of soft margins. The idea of soft margins is to penalize, but not prohibit classification errors while finding the maximum margin hyperplane. Thus, the maximum margin hyperplane does not necessarily separate all training instances of both classes. If the margin can be significantly increased, the better generalization can outweigh the penalty for a classification error on the training set. Figure 2.14 illustrates that the use of soft margins enables the calculation of general separating hyperplanes (left side) and can increase the generalization of a classifier, even if the classes are linear separable (right side). To conclude, SVMs using soft margins are still able to find a general model even in noisy data containing many outliers.

The second improvement is the introduction of kernel functions. For many real-world applications, it is not possible to find a hyperplane that separates the objects of two classes with sufficient accuracy. To overcome this problem the feature vectors are mapped into a higher dimensional space by introducing additional features that are constructed out of the original ones. Since this mapping is not linear, hyperplanes in the so-called kernel spaces



(a) Quadratic kernel **(b)** Radial basis kernel

Figure 2.15: Visualization of the separation when using a quadratic kernel (left) and a radial basis kernel(right).

provide much more complicated separators in the original space. This way the data in the original space is separated non linear. An import characteristic of the use of kernel functions is that the calculation of a maximum margin hyperplane in the kernel space is not much more expensive than in the original space. The reason for this effect is that it is not necessary to calculate the feature vectors in the kernel space explicitly. Since the optimization problem calculating the maximum margin hyperplane does only use a scalar product in the feature space, it is enough to replace this scalar product with a so-called kernel function to calculate the maximum margin hyperplane in the kernel space. Figure 2.15 displays the effects of two established kernel functions, a quadratic kernel and a radial basis kernel, to the separation in the original feature space.

SVMs have been extended to multi-class problems as well [PCST00]. A good introduction to SVMs is found in [Bur98] and in [CST00]. The performance of SVMs has been tested for various application like text categorization [Joa98], function prediction for protein data [JH98] and image recognition [PV98]. In general, SVMs demonstrated superior classification accuracy compared to most other classification systems. For the case that the data objects are given in a non-feature vector form, additional kernel function were introduced. For example, in [GFKS02] a kernel for mult-instance objects is

proposed that is capable to the multi-instance learning problem (see chapter 3.2).

Additionally to their exceptional accuracy, SVMs provide fast classification. However, the training of SVMs tends to take large periods of time, especially for multi-class variants calculating many binary separators. Last but not least, the models built by SVMs do not provide any explicit knowledge that might help to understand the nature of the given classes.

Decision Trees

This direction of classification tries to find a set of rules distinguishing the classes. A decision tree is a tree with the following characteristics:

- Each inner node corresponds to one attribute.
- Each leaf is associated with one of the classes.
- An edge represents a test on the attribute of its father node.

For classification, the attribute values of a new object are tested beginning with the root. At each node the data object can pass only one of the tests that are associated to the departing edges. The tree is traversed along the path of successful tests until a leaf is reached.

To construct a decision tree there are multiple approaches like [Qui93, BFOS84, GRG98]. Most of these split the training set recursively by selecting an attribute. The training set is now split by the values of the selected attribute. To determine the attribute the most promising candidate with respect to a given quality criteria is determined. An example quality criteria is the information gain, we introduced in section 2.1.2. This split step is done recursively for all subsets until a breaking criteria is reached or the members of a subset strictly belong to a class. Finally, more sophisticated approaches prune the decision tree to avoid overfitting and find a smaller model. Note that this approach to decision tree construction does not necessarily create

the smallest decision tree possible. However, the problem of finding a minimal decision has an exponential time complexity and the introduced heuristic solutions yield good classification accuracy in many cases.

The advantages of decision trees are that they are very robust against attributes that are not correlated to the classes because those attributes will not be selected for a split. Another more important feature is the induction of rules. Each path from the root to a leaf provides a rule that can be easily interpreted by a human user. Thus, decision trees are often used to explain the characteristics of classes.

The drawback of decision trees is that they are usually not capable to consider complex correlations between attributes because they only consider one attribute at a time. Thus, decision trees often model correlated data by complex rules which tend to overfitting. An more detailed discussion of decision trees is found in [HK01].

There are several additional approaches that are used for classification. For example, Neural networks are a very powerful direction of machine learning trying to rebuild the learning mechanism that can be found in the human brain. Among several other tasks, neural networks are applicable to classification. There are multiple variants of neural networks. An overview of neural networks can be found in [Big96]. Another direction that can be applied to classificaiton is inductive logical programming (ILP). ILP is often used for data mining in multi-relational data and employs search algorithms that find logical clauses that are valid in a database of facts [MDR94]. To conclude there are multiple solutions to find classifiers each having its advantages and disadvantages.

Part II

Data Mining in Multi-Instance Objects

Chapter 3

Multi-Instance Objects

One important type of compound object representations are multi-instance objects. Multi-instance objects describe each data object as an arbitrary sized set of feature representations. The following chapter motivates the use of multi-instance objects and lists example applications. Furthermore, it describes multi-instance learning as a special case of data mining in multi-instance objects and explains differences to the introduced solutions.

3.1 Motivation

Modern KDD applications need to process more and more complex objects containing more and more detailed information. The most established way of representing complex objects are feature vectors. However, feature vectors are often not capable to comprise all relevant characteristics of complex objects. Therefore, the use of compound object representations for complex objects often yields advantages by offering a richer object model. Multi-instance objects are a basic type of compound objects that consist of a set of feature representations $\{r_1, \ldots, r_k\}$ where all instances $r_i \in F$ belong to the same feature space F. Examples for objects that can be transformed into a multi-instance representation are:

- **CAD parts**
  CAD parts are three dimensional objects that can be viewed as a composition of spatial primitives. Thus, representing CAD parts as sets of spatial primitives provides a meaningful object representation that allows effective similarity search and data mining. KDD in CAD parts has applications like automatically deriving geometric and functional hierarchies to allow the user to browse CAD databases. We will introduce a solution for this application in the next chapter (4).

- **Websites**
  A website is a set of HTML-documents that is published by the same group or institution and usually serves a common purpose. A good representation for a website is a set of feature vectors corresponding to its HTML-documents. Data mining for websites is useful for several purposes, e.g. searching the WWW for companies or institutions. In chapter 5 we are going to discuss this application extensively.

- **Webpages**
  Unlike common text documents, most webpages provide a rich structure given by HTML-tags. According to this structure, a webpage can

be split into several blocks. Each block is likely to contain information about the same topic. Thus, similar to websites, single webpages can be treated as sets of feature vectors by transforming each block into a text feature vector.

- **Images**
  There are several methods of transforming images into a more suitable form for data mining or content based image retrieval [VT00]. Some of these methods extract sets of shapes or partition an image into a set of regions in order to find a better description of the image content.

- **Protein Structures**
  Proteins can be described by various representations. One of the most meaningful description for a protein is the three dimensional structure. The structure of a protein is of special interest in many applications, since it is strongly connected to its function. Unfortunately, the three dimensional structure of a protein is not necessarily the same under all conditions. The structure of a protein may change significantly for different docking partners or cell environments. In most cases, a different structure changes the function as well. To consider this effect, it is possible to treat a protein as set of all possible three dimensional structures. Each structure represents the protein having another docking partner or being in a different environment. This problem led to the development of multi-instance learning which is an important special case of data mining in multi-instance objects.

The presented list of applications for multi-instance objects is not exhaustive and there are many other scenarios where employing multi-instance objects might be beneficial. Generally, modelling complex objects as set of instances preserves more information than using a single feature vector. On the other hand, the processing of multi-instance objects is not as computationally demanding as processing graphs or trees. Figure 3.1 illustrates the

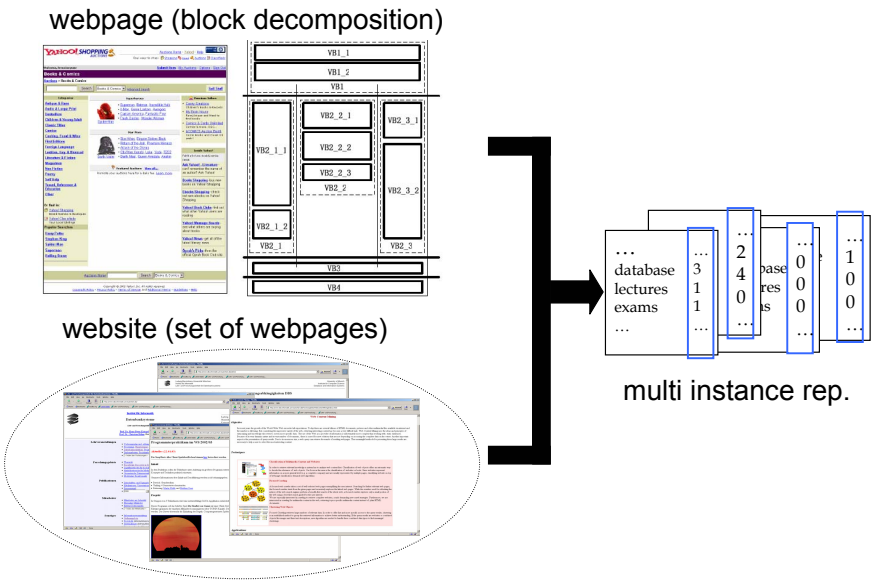


Figure 3.1: Webpages as well as websites can be transformed into multi-instance objects.

transformation of websites and webpages into a multi-instance representation.

3.2 Multi-Instance Learning

Multi-instance learning is an established direction of data mining in multi-instance objects. However, the problem that is solved by multi-instance learning is only a subproblem of classification of multi-instance objects. Multi-instance learning was introduced in [DLLP97] and tries to solve the following problem. Consider a set of objects O where each object o is represented by a set of feature vectors $\{v_1, \ldots, v_n\}$ with $v_i \in F$ and F is a feature space. Each object has a label $l = G(o)$ with $l \in \{relevant, other\}$. The class of an object o is determined by the existence of an instance $v_i \in o$ that is specific to the *relevant* class. If o contains at least one such instance, o is considered

to be relevant. If o does not contain any *relevant* instance, it is considered to belong to the *other* class. Thus, the classifier has to find out which of the instances in the training set of relevant multi-instance objects are specific to the relevant class.

To solve this problem, multiple solutions has been proposed. [DLLP97] proposes a solution, employing axis-parallel rectangles. [WZ00] introduced several methods of kNN classification based on the minimal Hausdorff distance. In [CZ00] a method is proposed to derive noise tolerant rules for multi-instance data. [GFKS02] introduces a specialized kernel function to enable support vector machines and other kernel based learning algorithms to learn from multi-instance examples.

Though multi-instance learning algorithms have been successfully used in special drug prediction examples, the assumption that a single instance can determine the class of a complete object does not hold in many applications. For example, in protein classification and drug prediction one structural description might not be enough to predict the class of a given molecule. So-called allosteric proteins change their shape depending on a particular binding partner and therefore need more than one instance to fit into a certain pattern in order to be properly described. Furthermore, this assumption does not hold for the applications that are treated in the following two chapters. The class of a website cannot be determined in a reliable way by examining one webpage and CAD parts are not necessarily similar if they share one out of several similar spatial primitives. To solve the problems in the next two chapters, more than one instance of a multi-instance object has to be considered to achieve best possible results.

Chapter 4

Clustering and Similarity Search in CAD Databases using Multi-Instance Representations

CAD Databases are an important application area of spatial data mining. Clustering and classification algorithms are used to organize the huge amounts of three dimensional objects or retrieve objects of similar shape. In this chapter, a new approach to represent three dimensional parts as multi-instance objects is introduced. To compare these multi-instance objects, the minimal matching distance is used, providing a metric distance measure for multi-instance objects that is computable in cubic time. Furthermore, a selective filter step is introduced that increases the efficiency of similarity queries that are used in distance based data mining algorithms like kNN classifiers or density based clustering algorithms. The evaluation uses OPTICS to demonstrate the superior quality of the new similarity measure compared to three established methods on two real-world CAD data sets. Furthermore, the efficiency of the filter step is compared to the performance of similarity search methods employing a one vector representation.

4.1 Motivation

In the last ten years, an increasing number of data mining techniques has emerged for which efficient and effective support of similarity queries is substantial. In general, the importance of similarity search grows in application areas such as multimedia, medical imaging, molecular biology, computer aided engineering, marketing and purchasing assistance, etc. [Jag91, AFS93, MG93, FBF+94, FRM94, ALSS95, BKK97, BK97, Kei99]. Particularly, the task of finding similar shapes in two dimensional and three dimensional spaces becomes more and more important. Examples for new applications that require the data mining of similar three dimensional objects include databases for molecular biology, medical imaging and computer aided design.

Especially, the development, design, manufacturing and maintenance of modern engineering products is a very expensive and complex task. Effective similarity models are required for two- and three-dimensional CAD applications to cope with rapidly growing amounts of data. Shorter product cycles and a greater diversity of models are becoming decisive competitive factors in the hard-fought automobile and aircraft market. These demands can only be met if the engineers have an overview of already existing CAD parts. In this chapter, we introduce an effective and flexible similarity model for complex three dimensional CAD data which can be used for distance based data mining algorithms such as density based clustering [EKSX96, ABKS99] and kNN classification. This model is particularly suitable for voxelized data that often occurs in CAD applications. It is based on the idea of representing a three dimensional part as multi-instance object and was published in [BKK+03].

The remainder of the chapter is organized as follows: In section 4.2 we shortly review already existing spatial similarity models and provide a categorization of the techniques into feature-based models and direct geometric models. Section 4.3 provides the basis for similarity models based on voxelized CAD objects. We address the issues of translation, rotation, reflection

and scaling invariances. Furthermore, we adapt three known similarity models to voxelized three dimensional data. Based on the most promising of these three models, we explain in section 4.4 our new approach based on multi-instance objects. In section 4.5, we analyze the different similarity models by means of hierarchical clustering. We show that our new similarity approach efficiently generates more significant results compared to the traditional approaches based on single feature vectors and is thus more suitable for data mining. The experiments are based on two real-world test data sets of our industrial partners, a German car manufacturer and an American aircraft producer.

4.2 Related Work

In recent years, considerable work on similarity search in database systems has been published. Many of the previous approaches, however, deal with one dimensional or two dimensional data, such as time series, digital images or polygonal data. Most of them do not support three dimensional objects or are not suitable for voxelized data. In this section, we shortly list different approaches to establish similarity measures. We provide a classification of the techniques into feature-based models and direct geometric models.

4.2.1 Feature-Based Similarity

A widely used class of similarity models is based on the paradigm of feature vectors as it is described in chapter 2.1.1. The paradigm of feature-based similarity has been successfully applied to the retrieval of similar spatial objects. Examples include structural features of TWO DIMENSIONAL contours [MG93], angular profiles of polygons [BMH92], rectangular covers of shapes [Jag91], algebraic moment invariants [FBF+94], two dimensional section coding [BK97, BKK97], and three dimensional shape histograms for biomolecular objects [AKKS99]. Non-geometric applications include similar-

ity search on time series [AFS93, FRM94, ALSS95] and color histograms in image databases [NBE+93, FBF+94] among several others.

4.2.2 Geometry-Based Similarity

A class of models that is to be distinguished from the feature-based techniques are the similarity models that are defined by directly using the geometry of the objects. Two objects are considered similar if they minimize a distance criterion that is purely defined by the geometry of the objects. Examples include the similarity retrieval of mechanical parts [SKSH89], the difference volume approach [KSF+96, Kei99], and the approximation-based similarity model for three dimensional surface segments [KSS97].

4.3 Similarity Models for Voxelized CAD Objects

In this section, we describe three established similarity models. The first two models (the *volume* and the *solid-angle* approach) are based on an paraxial, equi-sized space partitioning. Then, the voxel approximations of the objects are transformed into shape histograms. These histograms are used as intuitive feature vectors. In the third model (the *cover sequence* approach), we do not need this space partitioning but obtain our feature vectors directly from the rectangular covers which approximate our object by minimizing the symmetric volume difference. This third model forms the starting point for our new approach based on vector sets which is introduced in section 4.4.

4.3.1 Shape Histograms

Histograms are usually based on a complete partitioning of the data space into disjoint cells which correspond to the bins of the histograms.

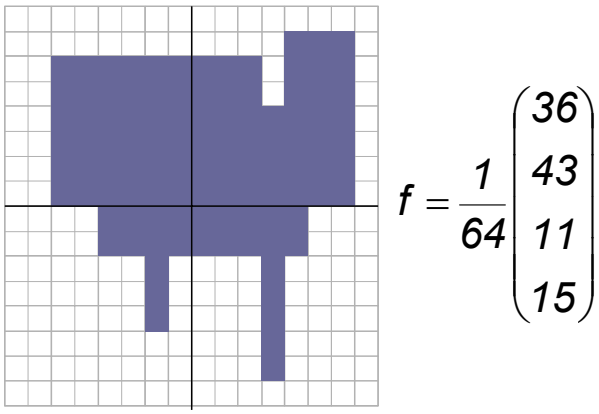


Figure 4.1: Space partitioning with 4 cells. The feature vector generated by the volume model is depicted on the right hand side.

We divide the data space into paraxial, equi-sized partitions (cf. Figure 4.1). This kind of space partitioning is especially suitable for voxelized data, as cells and voxels are of the same shape, i.e. cells can be regarded as coarse voxels.

Each of these partitions is assigned to one or several bins in a histogram, depending on the specific similarity model. By scaling the number of partitions, the number of dimensions of the feature vector can be regulated (cf. Figure 4.1). Obviously, the more partitions we use, the more smaller differences between the objects become decisive. The resulting feature vectors are compared by means of distance based similarity as was introduced in chapter 2.1.1. Throughout this chapter, we will use Euclidian distance to calculate the similarity of feature vectors.

4.3.2 Normalization

Similarity models for CAD data should recognize similar parts, independently of their spatial location. The four respectively five tires of a car are similar, although they are located differently. Furthermore, reflected parts, e.g. the

right and left front door of a car, should be recognized as similar as far as the design is concerned. If we look at the production, reflected parts are no longer similar and have to be treated differently. Likewise, the actual size of the parts may or may not have an influence on the similarity model. To sum up, a similarity model for CAD data should take translation and rotation invariances into account, whereas reflection and scaling invariances have to be adjustable.

CAD objects are usually designed and constructed in a standardized position in the center of the coordinate system. We store each object in this standard position and normalize it with respect to scaling. Furthermore, we store the scaling factors for each of the three dimensions. Thus, we can (de)activate the scaling invariance depending on the user's needs at runtime. In the case of CAD applications, not all possible rotations are considered, but only 90°-rotations. This yields 24 different possible positions for each object. For similarity search where we are not confined to 90°-rotations, we can apply principal axis transformation in order to achieve invariance with respect to rotation. Taking also reflection into account, we may obtain $24 \cdot 2 = 48$ varying positions. We could achieve 90°-rotation and reflection invariance by storing 48 different feature vectors for each object in the database or by carrying out 48 different permutations of the query object at runtime. As we want to decide at runtime whether we want to consider reflection invariance or not, we chose the second variant. Throughout our experiments, we considered invariance with respect to translation, reflection, scaling and 90°-rotation.

Taking all these transformations into account, we get the following extended similarity definition.

Definition 4.1 *(Extended Feature-Based Object Similarity)*
Let O be the domain of the objects, $F : O \to \mathbb{R}^d$ a mapping of the objects into the d-dimensional feature space, and $dist : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ a distance function between two d-dimensional feature vectors. Furthermore, let $\mathcal{T}$ be

a set of all user-dependent combinations of translation, scaling, rotation and reflection transformations. Then, simdist: $O \times O \rightarrow \mathbb{R}$ *is defined as follows:*

$$simdist(Obj_1, Obj_2) = \min_{T \in \mathcal{T}} \{dist(F(Obj_1), F(T(Obj_2)))\}.$$

4.3.3 Spatial Features

After partitioning the data space, we have to determine the spatial features of the objects for each grid cell depending on the chosen model. In order to do that, we first have to introduce some notations:

The data space is partitioned in each dimension into p grid cells. Thus, our histogram will consist of $k \cdot p^3$ bins where $k \in \mathbb{N}$ depends on the model which specifies the kind and number of features extracted from each cell. For a given object o, let $V^o = \{V_i^o \mid 1 \leq i \leq p^3\}$ be the set of voxels that represents o where V_i^o are the voxels covered by o in cell i. $\bar{V}^o \subseteq V^o$ denotes the set of voxels at the surface of the objects and $\dot{V}^o \subseteq V^o$ denotes the set of the voxels inside the object such that $\bar{V}^o \cup \dot{V}^o = V^o$ and $\bar{V}^o \cap \dot{V}^o = \emptyset$ holds.

Let f_o be the computed feature vector of an object o. The i-th value of the feature vector of object o is denoted by $f_o^{(i)}$.

Let r be the number of voxels of the dataspace in each dimension. In order to ensure a unique assignment of the voxels to a grid cell, we assume that $\frac{r}{p} \in \mathbb{N}$.

The Volume Model

A simple and established approach to compare two objects is based on the number of the object voxels $|V_i^o|$ in each cell i of the partitioning. In the following, this model is referred to as *volume model*. Each cell represents one dimension in the feature vector of the object. The i-th dimension of the feature vector ($1 \leq i \leq p^3$) of object o can be computed by the normalized number of voxels of o lying in cell i, formally:

$$f_o^{(i)} = \frac{|V_i^o|}{K} \qquad \text{where} \qquad K = (\frac{r}{p})^3$$

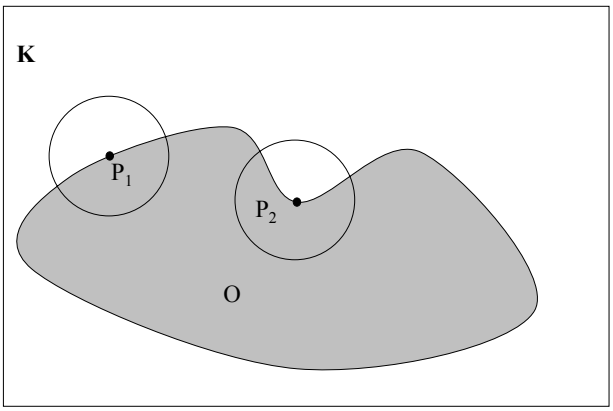


Figure 4.2: A sample object with different shapes at the surface-points p_1 and p_2.

Figure 4.1 illustrates the volume model for the TWO DIMENSIONAL case.

The Solid-Angle Model

The *solid-angle* method [Con86] measures the concavity and the convexity of geometric surfaces. Let K_c be a set of voxels that describes a three dimensional voxelized sphere with central voxel c. For each surface-voxel $\bar{v}$ of an object o, the so-called solid-angle value is computed as follows. The voxels of o which are inside $K_{\bar{v}}$ are counted and divided by the size of $K_{\bar{v}}$, i.e. the number of voxels of $K_{\bar{v}}$. The resulting measure is called the solid-angle value $\text{SA}(\bar{v})$ and can be computed as follows:

$$\text{SA}(\bar{v}) = \frac{|\, K_{\bar{v}} \cap V^o \,|}{|\, K_{\bar{v}} \,|} \quad \text{, where:}$$

$K_{\bar{v}} \cap V^o =$

$\{w \in K_{\bar{v}} \mid \exists v \in V^o : \; w.x = v.x \;\wedge\; w.y = v.y \;\wedge\; w.z = v.z\}$

A small solid-angle value $\text{SA}(\bar{v})$ indicates that an object is convex at voxel $\bar{v}$. Otherwise, a high value of $\text{SA}(\bar{v})$ denotes a concave shape of an object at

voxel $\bar{v}$. Figure 4.2 illustrates this behavior.

The solid-angle values of the cells are transferred into the according histogram bins as described in the following. We distinguish between three different types of cells:

1. Cell i contains surface-voxels of object o, i.e. $\bar{V}_i^o \neq \emptyset$. The mean of all SA-values of the surface-voxels is computed as the feature value of this cell:
$$f_o^{(i)} = \frac{1}{m} \sum_{j=1}^{m} \text{SA}(\bar{v}_{i_j})$$
where $\bar{V}_i^o = \{\bar{v}_{i_1}, \ldots, \bar{v}_{i_m}\}$.

2. Cell i contains only inside-voxels of object o, i.e. $\bar{V}_i^o = \emptyset$ and $V_i^o \neq \emptyset$. The feature value of this cell is set to 1, i.e. $f_o^{(i)} = 1$.

3. Cell i contains no voxels of object o,i.e. $V_i^o = \emptyset$. The value of the according bin of the histogram is 0, i.e. $f_o^{(i)} = 0$.

The Cover Sequence Model

The two models described above are based on a complete partitioning of the data space into disjoint cells. In this section, we adapt a known model [Jag91, JB91] to voxelized three dimensional data which is not restricted to this rigid space partitioning but rather uses a more flexible object-oriented partitioning approach. This model is in the following referred to as *cover sequence model*.

As depicted in Figure 4.3, each edge of an object can be extended infinitely in either direction to obtain a grid of lines. Each rectangle in this grid is called a *grid primitive*, and is located either entirely inside the object or entirely outside of the object. Furthermore, any pair of adjacent grid primitives must also form a rectangle, respectively a cuboid in the three dimensional data space. The basic idea of this model is to find large clusters of grid primitives, called *covers* which approximate the object as good as possible [JB91].

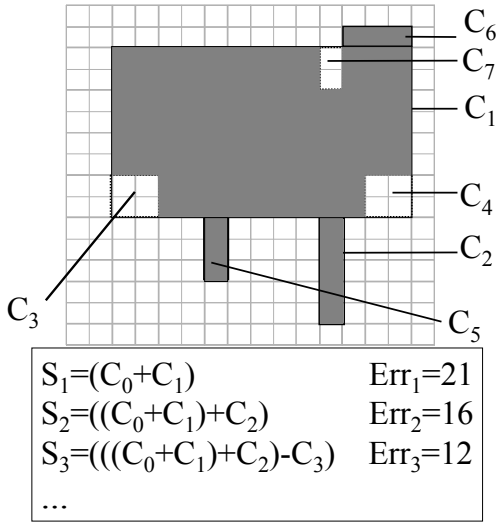


Figure 4.3: Cover sequence model.

The quality of such a cover sequence S_k is measured by the symmetric volume difference Err_k between the object O and the sequence S_k (cf. Figure 4.3). Formally, let the covers be drawn from the set $\mathcal{C}$ of all possible rectangular covers. Then each unit i of the cover sequence comprises a pair $(C_i \in \mathcal{C}, \sigma_i \in \{+, -\})$, where "+" represents set union and "−" represents set difference. The sequence after k units is:

$$S_k = (((C_0 \sigma_1 C_1)\sigma_2 C_2) \ldots \sigma_k C_k),$$

where C_0 is an initial empty cover at the zero point.

The symmetric volume difference after k units is:

$$Err_k = |O \; XOR \; S_k|$$

, where O is the approximated object.

Jagadish and Bruckstein [JB91] suggest two algorithms for the retrieval of S_k: a *branch and bound* algorithm with exponential runtime complexity, and a *greedy* algorithm with polynomial runtime complexity which tries to minimize Err_i in each step $i \leq k$. Throughout our experiments we used this second algorithm.

In [Jag91], Jagadish sketches how a three dimensional cover sequence $S_k = (((C_0\sigma_1 C_1)\sigma_2 C_2)\ldots\sigma_k C_k)$ of an object o, can be transformed into a $6 \cdot k$-dimensional feature vector. Thereby, each cover C_{i+1} with $0 \leq i \leq k-1$ is mapped onto 6 values in the feature vector f_o in the following way:

$$\begin{aligned} f_o^{6i+1} &= x\text{-position of } C_{i+1} \\ f_o^{6i+2} &= y\text{-position of } C_{i+1} \\ f_o^{6i+3} &= z\text{-position of } C_{i+1} \\ f_o^{6i+4} &= x\text{-extension of } C_{i+1} \\ f_o^{6i+5} &= y\text{-extension of } C_{i+1} \\ f_o^{6i+6} &= z\text{-extension of } C_{i+1} \end{aligned}$$

If an object O can be described by a sequence S_j with $j < k$ covers and $Err_j = 0$, we assign $((S_j\sigma_{j+1}C_0)\ldots\sigma_k C_0)$ to S_k. These dummy-covers C_0 do not distort our similarity notion (cf. Definition 4.1), but guarantee that all feature vectors are of the same dimensionality. Thus, we can use common spatial index-structures [BKK96, LJF94, BBJ+00] in order to accelerate similarity queries.

4.4 A Multi-Instance Representation for CAD-Parts

As proposed in [Jag91] a data object is now represented as a feature vector. For similarity queries, this method yields a major problem. Always comparing the two covers having the same ranking according to the symmetric volume difference does not make sense in all cases. Two objects can be considered very different because of the order of their covers, although they are very similar by intuition. The reason for this effect is that the order of the covers does not guarantee that the most similar covers due to size and position will be stored in the same dimensions. Especially for objects generating

two or more covers and having almost the same volume, the intuitive notion of similarity can be seriously disturbed. Thus, the possibility to match the covers of two compared objects with more degrees of freedom might offer a better similarity measure. Figure 4.4 displays a two dimensional example of a comparison between a query object and a very similar database object. The first sequence (cf. Figure 4.4(a)) represents the covers of the query object in the order given by the symmetric volume difference. Cover C_2, C_3 and C_4 in sequence (a) are not very similar to the corresponding covers of the database object, displayed in the second sequence and therefore the calculated similarity is relatively weak. By rearranging the order of these covers, the total distance between the query object and the database object is considerably decreasing, which is displayed in Figure 4.4(b). Thus, the new order preserves the similarity between the objects much better.

To overcome the problem, the author in [Jag91] proposes to generate several good representations of the query object and then process a query for each of the representations. Afterwards the union of the returned database objects is taken as a result. We can obtain different representations by permuting the order of the found covers and choose the most "promising" orderings to create the query vectors. Though, the method may offer reasonable results in many cases, there is no guarantee that the ordering providing the minimum distance is included within this selection. Thus, the whole similarity measure is dependent on the criteria used to select the most "promising" orderings. Since there is no well defined selection criterion known so far, the solution does not necessarily offer a precisely defined similarity measure.

Another solution for the problem is to consider all possible permutations. Since the distance between two objects can now be considered as the minimum distance over all possible orderings, the distance is defined as follows.

Definition 4.2

Let $exch : \mathbb{N} \times \mathbb{N} \times \mathbb{R}^{(d \cdot k)} \rightarrow \mathbb{R}^{(d \cdot k)}$ *be a function, where* $exch(i, j, \vec{x})$ *exchanges the* d *successive components beginning with dimension* $i \cdot d+1$ $(0 \leq i \leq$

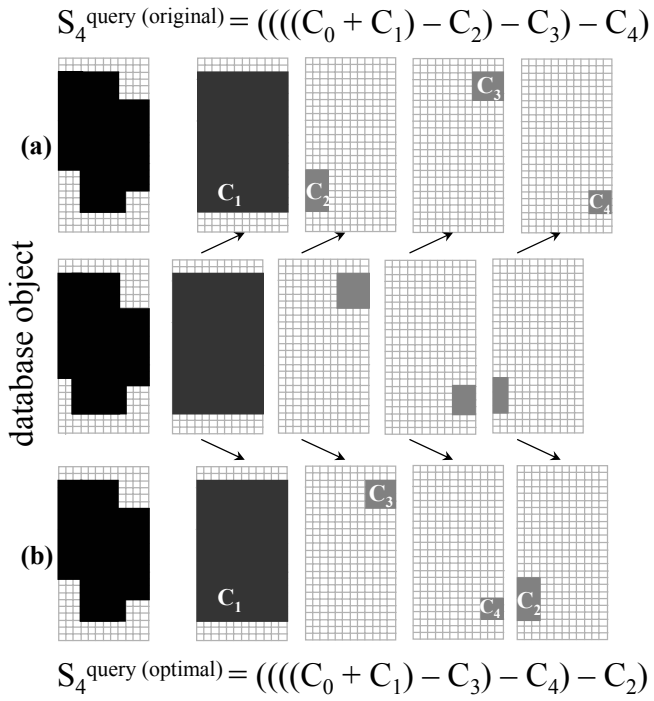


Figure 4.4: Examples demonstrating the advantage of free permutations.

$k-1)$ *with the* d *successive components beginning with dimension* $j \cdot d+1 (0 \leq j \leq k-1)$ *of a vector* $\vec{x} \in I\!R^{(k \cdot d)}$.

$Exch : I\!R^{(k \cdot d)} \rightarrow 2^{I\!R^{(k \cdot d)}}$ *is the function that generates the set of all vectors that can be generated by applying* $exch(i, j, \vec{x})$ *arbitrarily many times to a vector* $\vec{x}$ *using any combination for* i *and* j.

Definition 4.3

(minimum Euclidian distance under permutation)

Let O *be the domain of the objects, let* $F : O \rightarrow I\!R^{(k \cdot d)}$ *be a mapping of the objects into the* $k \cdot d$*-dimensional feature space, and let* $dist : I\!R^{(k \cdot d)} \times I\!R^{(k \cdot d)} \rightarrow I\!R$ *be a distance function between two* $k \cdot d$*-dimensional feature vectors. Then*

$dist_{\pi-eucl.}: O \times O \rightarrow \mathbb{R}$ is defined as follows:

$$dist_{\pi-eucl.}(Obj_1, Obj_2) = \min_{\vec{y} \in Exch(F(Obj_2))} \{dist(F(Obj_1), \vec{y})\}$$

With a growing number of describing covers k, the processing time of considering all possible permutations increases exponentially, since there are $k!$ many permutations. With computation cost rising this rapidly, it is obvious that the description length k has to be kept low which is not acceptable for all applications.

To guarantee that the permutation with the minimal distance is used, our approach does not work with one single feature vector, but with a set of feature vectors in lower dimensions. By treating the data objects as sets of d-dimensional feature vectors with a maximum cardinality of k, we introduce a new model for representing data objects in similarity search systems, the so called *multi-instance model*. In the following sections, we will discuss the concept of multi-instance representation in detail with the goal of defining a similarity search system that can be used as efficient foundation of distance based data mining algorithms.

4.4.1 Reasons for the Use of Multi-Instance Objects

The representation of extracted features as a multi-instance object is a generalization of the use of just one large feature vector. It is always possible to restrict the model to a feature space in which a data object will be completely represented by just one feature vector. In this applications, the use of multi- instance representations is able to avoid the problems that occur by storing a set of covers according to a strict order. Therefore, it is possible to compare two objects more intuitively, causing a relatively small increase of calculation costs compared to the distance calculation in conventional feature vector models. Another advantage of this approach is the better storage utilization. It is not necessary to force objects into a common size if they are

represented by sets of different cardinality. For our current application, there is no need for dummy covers to fill up the feature vectors. If the quality of the approximation is optimal with less than the maximum number of covers, only this smaller number of vectors has to be stored and loaded. In the case of a one-vector representation, avoiding dummies is not possible without further modifications of the employed search system. Furthermore, we are able to distinguish between the distance measure used on the feature vectors of a set and the way we combine the resulting distances between the single feature vectors. For example, this possibility might be useful when defining partial similarity where it is only necessary to compare the closest $i < k$ vectors of a set.

4.4.2 Distance Measures on Multi-Instance Objects

There are already several distance measures proposed on sets of objects. In [EM97] the authors survey the following three which are computable in polynomial time: the Hausdorff distance, the sum of minimum distances and the (fair-)surjection distance. Furthermore, they introduce the link distance which is computable in polynomial time, too. The Hausdorff distance does not seem to be suitable as a similarity measure, because it relies too much on the extreme positions of the elements of both sets. The last three distance measures are suitable for modelling similarity, but are not metric. This circumstance makes them unattractive, since there are only limited possibilities for processing similarity queries efficiently when using a non-metric distance function. In [EM97] the authors also introduce a method for expanding the distance measures into metrics, but as a side effect the complexity of distance calculation becomes exponential. Furthermore, the possibility to match several elements in one set to just one element in the compared set, is questionable when comparing sets of covers as in our application.

A distance measure on vector sets that demonstrates to be suitable for defining similarity in our application is based on the *minimum weight perfect*

matching of sets. This well-known graph problem can be applied here by building a complete bipartite graph $G = (S_1 \cup S_2, E)$ between the vector sets S_1 and S_2. The weight of each edge $(x, y) \in E$ with $x \in S_1$ and $y \in S_2$ in this graph G is defined by their distance $dist(x, y)$. A perfect matching is a subset $M \subseteq E$ that connects each $x \in S_1$ to exactly one $y \in S_2$ and vice versa. A minimum weight perfect matching is a matching with a minimum sum of weights of its edges. Since a perfect match can only be found for sets of equal cardinality, it is necessary to introduce weights for unmatched nodes when defining a distance measure.

Definition 4.4 *(enumeration of a set)*
Let S be any finite set of arbitrary elements. Then π is a mapping that assigns $s \in S$ to a unique number $i \in \{1, .., |S|\}$. This is written as $\pi(S) = (s_1, .., s_{|S|})$. The set of all possible enumerations of S is named $\Pi(S)$.

Definition 4.5 *(minimal matching distance)*
Let O be the domain of the objects and X be a set with $|X| \leq k$ and $X \subseteq 2^V$ with $V \subset \mathbb{R}^d$. Furthermore, let $F : O \to X$ be a mapping of the objects into X, and $dist : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ a distance function between two d-dimensional feature vectors. We assume w.l.o.g. $|F(Obj_1)| = m \geq n = |F(Obj_2)|$, and $F(Obj_1) = \{x_1, .., x_m\}$ and $F(Obj_2) = \{y_1, .., y_n\}$.

Then $dist_{mm}^{w,dist} : O \times O \to \mathbb{R}$ is defined as follows:

$$dist_{mm}^{w,dist}(Obj_1, Obj_2) =$$

$$\min_{\pi \in \Pi(F(Obj_1))} \left(\sum_{i=1}^{n} dist(x_{\pi(i)}, y_i) + \sum_{l=n+1}^{m} w(x_{\pi(l)}) \right)$$

where $w : \mathbb{R}^d \to \mathbb{R}^+$ is a weight function for the unmatched elements.

The weight function w provides the penalty given to every unassigned element of the set having a larger cardinality. Let us note that *minimum matching distance* is a specialization of *netflow distance* which is introduced in [RB01]. In [RB01] it is proven that netflow distance is a metric and that it

is computable in polynomial time. Therefore, we derive the following lemma without further proof.

Lemma 4.1 *The minimal matching distance is a metric if* $dist : \mathbb{R}^d \times \mathbb{R}^d \rightarrow \mathbb{R}$ *is a metric and* $w : \mathbb{R}^d \rightarrow \mathbb{R}^+$ *meets the following conditions:*

- $w(\vec{x}) > 0$, *for each* $\vec{x} \in V$
- *for* $\vec{x}, \vec{y}$, *with* $\vec{y}, \vec{x} \in V$ *the following inequality holds :* $w(\vec{x}) + w(\vec{y}) \geq dist(\vec{x}, \vec{y})$

In our application, the *minimum Euclidian distance under permutation* can be derived from the *minimum matching distance*. By selecting the squared Euclidian distance as distance measure on V and taking the squared Euclidian norm as weight function, the distance value calculated by the minimum matching distance is the same as the squared value of the minimum Euclidian distance under permutation. This follows from the definitions of both distance measures. Let us note that it is necessary to extract the square root from this distance value to preserve the metric character.

Though it was shown that the netflow distance can be calculated in polynomial time, it is not obvious how to achieve it. Since we are only interested in the minimum matching distance, it is enough to calculate a minimum weight perfect matching. Therefore, we apply the method proposed by Kuhn [Kuh55] and Munkres [Mun57]. The method is based on the successive augmentation of an alternating path between both sets. Since it is guaranteed that this path can be expanded by one further match within each step taking $O(k^2)$ time and there is a maximum of k steps, the all over complexity of a distance calculation using the method of Kuhn and Munkres is $O(k^3)$ in the worst case. Let us note that for larger numbers of k this is far better than the previously mentioned method on $k!$ many permutations.

4.4.3 Answering Similarity Queries on Vector Set Data Efficiently

Though we discussed the time for a single distance calculation, the problem of efficiently processing similarity queries in large databases is still unanswered. Since it is necessary here to locate the objects belonging to the result in comparably short time, the use of index structures that avoid comparing a query object to the complete database is mandatory. For one-vector-represented data objects there exists a wide variety of index structures that are suitable for answering similarity queries efficiently, e.g. the TV-Tree [LJF94], the X-Tree [BKK96] or the IQ-Tree [BBJ+00]. But unfortunately, those index structures cannot be used directly to retrieve multi-instance objects.

To accelerate similarity queries on multi-instance objects, the simplest approach is the use of more general access structures. Since the minimal matching distance is a metric for the right choice of distance and weight function, the use of index structures for metric objects like the M-Tree [CPZ97] offers a good possibility. Another approach is the use of the above mentioned high-dimensional index structures for querying sub tasks of the complete similarity query. In the following, we will introduce a filter step that is based on the relation between a set of d-dimensional vectors and its extended centroid.

Definition 4.6
Let $V \subset \mathbb{R}^d$ be a set of d-dimensional vectors. Then $w_{\vec{\omega}} : V \to \mathbb{R}$ denotes a set of weight functions having the following properties: $\vec{\omega} \in \mathbb{R}^d \backslash V$ and $w_{\vec{\omega}}(\vec{x}) = \|\vec{x} - \vec{\omega}\|_2$, where $\|\vec{x} - \vec{y}\|_2$ denotes the Euclidian distance between $\vec{x}, \vec{y} \in \mathbb{R}^d$.

Definition 4.7 *(extended centroid)*
Let $V \subset \mathbb{R}^d$ and $X \subset 2^V$ with $|X| \leq k$ be a set. Then the extended centroid $C_{k,\vec{\omega}}(X)$ is defined as follows:

$$C_{k,\vec{\omega}}(X) = \frac{\sum_{i=1}^{|X|} x_i + (k - |X|) \cdot \vec{\omega}}{k},$$

where $X = \{x_1, .., x_{|X|}\}$ *and* $\vec{\omega} \in I\!R^d \backslash V$.

Lemma 4.2 *Let* $V \subset I\!R^d$ *be a set and* $\vec{\omega} \in I\!R^d \backslash V$. *Furthermore, let* X,Y *be two vector sets with* $\vec{x}_i \in X$,$\vec{y}_i \in Y$, *let* $C_{k,\vec{\omega}}(X)$, $C_{k,\vec{\omega}}(Y)$ *be their extended centroids and let* $dist_{mm}^{dist_{Eucl.},w_{\vec{\omega}}}$ *be the minimal matching distance using* $w_{\vec{\omega}}$ *as weight function defined on* V. *Then the following inequality holds:*

$$k \cdot \|C_{k,\vec{\omega}}(X) - C_{k,\vec{\omega}}(Y)\|_2 \leq dist_{mm}^{dist_{Eucl.},w_{\vec{\omega}}}(X,Y).$$

Proof. *Let* π *be the enumeration of the indices of* X *that groups the* x_i *to* y_i *according to the minimum weight perfect matching. w.l.o.g. we assume* $|X| = n \geq m = |Y|$ *and* $n - m = \delta$.

$$\begin{aligned}
& k \cdot \|C_{k,\vec{\omega}}(X) - C_{k,\vec{\omega}}(Y)\|_2 = \\
& k \cdot \| \frac{\sum_{i=1}^{n} x_{\pi(i)} + \sum_{i=1}^{k-n} \vec{\omega}}{k} - \frac{\sum_{i=1}^{m} y_i + \sum_{i=1}^{k-m} \vec{\omega}}{k} \|_2 \\
&= \| \sum_{i=1}^{m+\delta} x_{\pi(i)} + \sum_{i=1}^{k-m-\delta} \vec{\omega} - \sum_{i=1}^{m} y_i - \sum_{i=1}^{k-m} \vec{\omega} \|_2 \\
&= \| \sum_{i=1}^{m} x_{\pi(i)} - \sum_{i=1}^{m} y_i + \sum_{i=m+1}^{m+\delta} x_{\pi(i)} - \sum_{i=m+1}^{m+\delta} \vec{\omega} \|_2 \\
&\overset{tri.\ ineq.}{\leq} \| \sum_{i=1}^{m} (x_{\pi(i)} - y_i) \|_2 + \| \sum_{i=m+1}^{m+\delta} (x_{\pi(i)} - \vec{\omega}) \|_2 \\
&\overset{tri.\ ineq.}{\leq} \sum_{i=1}^{m} \|x_{\pi(i)} - y_i\|_2 + \sum_{i=m+1}^{m+\delta} \|x_{\pi(i)} - \vec{\omega}\|_2 \\
&= \sum_{i=1}^{m} \|x_{\pi(i)} - y_i\|_2 + \sum_{i=m+1}^{m+\delta} w_{\vec{\omega}}(x_{\pi(i)}) \\
&= dist_{mm}^{dist_{Eucl.},w_{\vec{\omega}}}(X,Y)
\end{aligned}$$

◇

The lemma proves that the Euclidian distance between the extended centroids multiplied with the cardinality of the larger set is a lower bound for the minimal matching distance under the named preconditions. Therefore, when computing e.g. ε-range queries, we do not need to examine objects whose extended centroids have a distance to the query object q that is larger than ε/k. A good choice of $\vec{\omega}$ for our application is $\vec{0}$ because it has the

shortest average distance for the position and has no volume. Additionally, the conditions for the metric character of minimum matching distance are satisfied because there are no covers having no volume in any data object.

To implement the filter step, we stored the extended centroids in a 6-dimensional X-Tree [BKK96]. Since this index structure provides high performance for similarity queries, it offers an efficient way to determine the keys of the candidate multi-instance objects. Afterwards we loaded the multi-instance objects themself to determine the membership of the object within the result. Using established algorithms for ε-range [KSF+96] and kNN-queries [SK98] that employ filter steps, both kinds of queries can be answered efficiently.

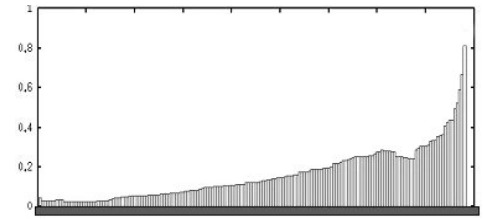
4.5 Evaluation

In this section, we present the results of our experimental evaluation. We apply the hierarchical clustering algorithm OPTICS [ABKS99] for a objective evaluation of similarity models to show that the multi-instance representation is a better foundation for clustering. The resulting reachability plot can be used to derive a meaningful geometric hierarchy that can be used to organize the clustered CAD parts. To demonstrate the efficiency of the introduced filter, we performed kNN queries.

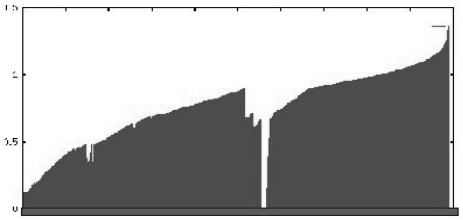
4.5.1 Data Sets

We evaluated the three proposed models on the basis of two real-world data sets. The first one, in the following referred to as *car data set*, contains approximately 200 CAD objects from a German car manufacturer. The car data set contains several groups of intuitively similar objects, e.g. a set of tires, doors, fenders, engine blocks and kinematic envelopes of seats.

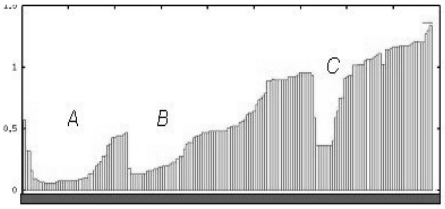
The second data set contains 5,000 CAD objects from an American aircraft producer and in the following is called *Aircraft Data set*. This data set



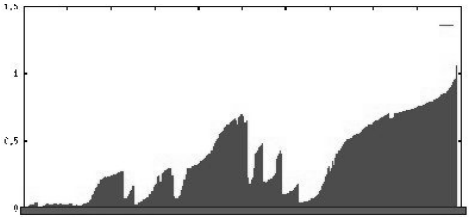
(a) car data set (volume model)



(b) aircraft data set (volume model)



(c) car data set (solid angle model)



(d) aircraft data set (solid angle model)

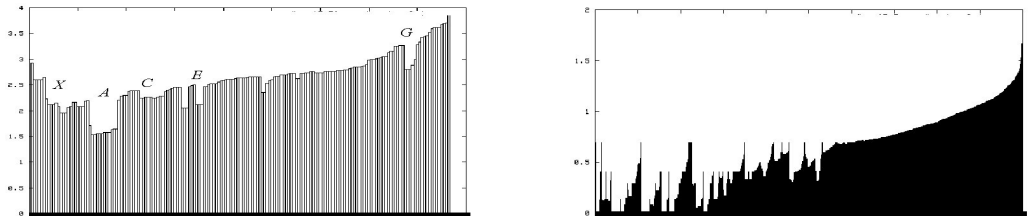
Figure 4.5: Reachability plots computed by OPTICS, using the volume (a,b) and solid angle (c,d) model [KKM+03].

contains many small objects, e.g. nuts, bolts, etc. and a few large ones, e.g. wings.

Using the cover sequence model and the multi-instance model, the data space of both data sets contains objects represented as voxel approximations using a raster resolution of $r = 15$. For the volume model and the solid-angle model, we used a raster resolution of $r = 30$. These values were optimized to the quality of the evaluation results.

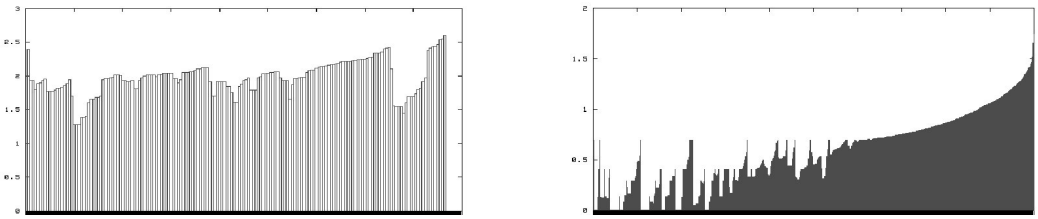
4.5.2 Clustering CAD-Parts

For the evaluation of the various similarity models, the density-based, hierarchical clustering algorithm OPTICS [ABKS99] (see. chapter 2.2.3) was used. OPTICS is well suited to derive a meaningful hierarchy of CAD parts, since it is insensitive to its parameters. Furthermore, it is likely that geometric shapes contain nested clusters. At last, the number of clusters is not known in advance and thus the ability of OPTICS to find an arbitrary number of



(a) car data set **(b)** aircraft data set

Figure 4.6: Reachability plots computed by OPTICS, using the cover sequence model with 7 covers.



(a) car data set **(b)** aircraft data set

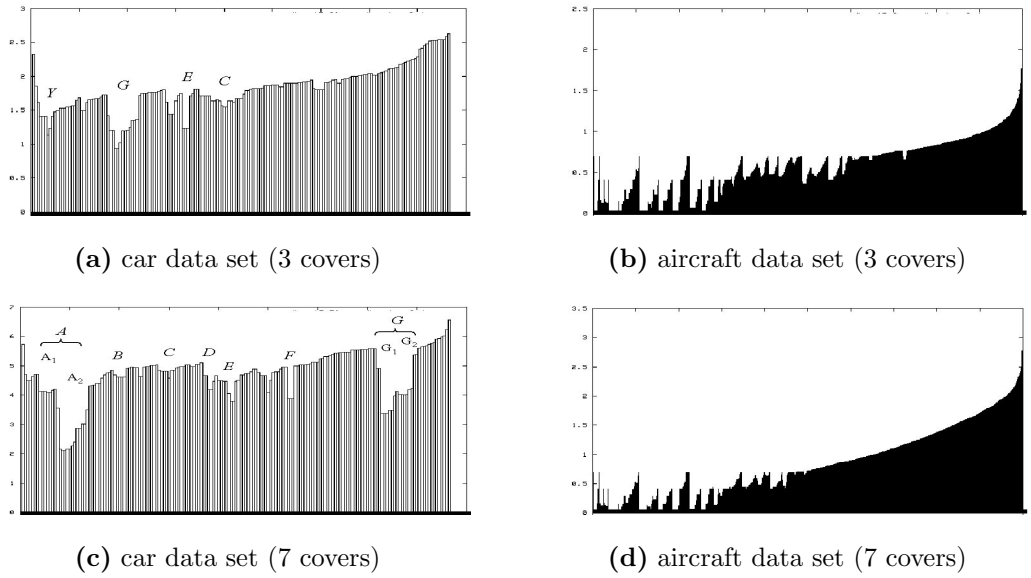
Figure 4.7: Reachability plots computed by OPTICS, using the cover sequence model with the minimum Euclidian distance under permutation with 7 covers.

clusters is very beneficial as well.

The reachability plots generated by OPTICS for all models are depicted in Figure 4.5, 4.6, 4.7 and 4.8.

Obviously, the volume model performs rather ineffective. The plots computed by OPTICS when applying the model on the car data set and the aircraft data set are depicted in Figure 4.5(a) and 4.5(b). Both plots show a minimum of structure, indicating that the volume model cannot satisfyingly represent the intuitive notion of similarity.

The solid-angle model performs slightly better. On the car data set, OPTICS found three clusters denoted as A, B, and C in Figure 4.5(c). We analyzed these clusters by picking out some samples of the objects grouped in each cluster. The result of this evaluation on the car data set is presented in



(a) car data set (3 covers)

(b) aircraft data set (3 covers)

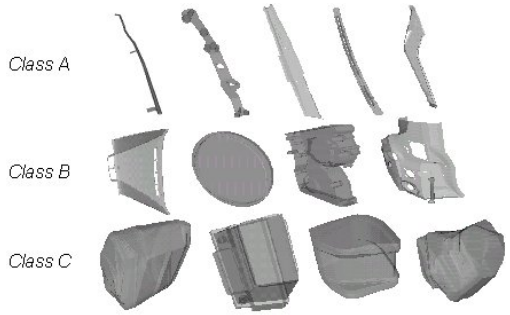
(c) car data set (7 covers)

(d) aircraft data set (7 covers)

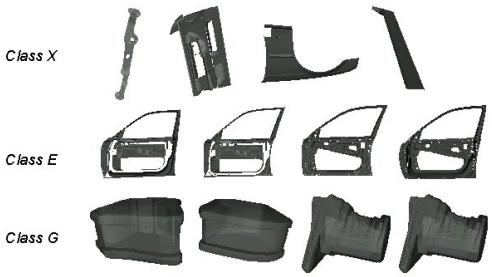
Figure 4.8: Reachability plots computed by OPTICS using the multi-instance model with 3 and 7 covers.

Figure 4.9(a). As it can be seen, the objects in clusters A and C are intuitively similar but the objects in B are not. Furthermore, there are clusters of intuitively similar objects, e.g. doors, which are not detected. Evaluating the solid-angle model using the aircraft data set, we made similar observations. The reachability plot computed by OPTICS (cf. Figure 4.5(d)) yields a clustering with a large number of hierarchical classes. But the analysis of the objects within each cluster displays that intuitively dissimilar objects are treated as similar. A further observation is the following: objects that are intuitively similar are clustered in different groups. This suggests the conclusion that the solid-angle model is also rather unsuitable as a similarity model for our real-world test data sets.

The plots computed by OPTICS for the cover sequence model, the cover sequence model using the minimum Euclidian distance under permutation and the multi-instance model (cf. Figure 4.6, 4.7 and 4.8) look considerably better. We will confirm this observation by evaluating the effectiveness of the different models in the following. We analyzed the cover sequence



(a) Classes found by OPTICS in the car data set using the solid-angle model (cf. Figure 4.5(c)) [KKM+03].



(b) Classes found by OPTICS in the car data set using the cover sequence model (cf. Figure 4.6(a)).

(c) Classes found by OPTICS in the car data set using the multi-instance model with 7 covers (cf. Figure 4.8(c)).

Figure 4.9: Evaluation of classes found by OPTICS in the Car Data set.

model without permutations as well as under full permutations, i.e. using the Euclidian distance under permutation. Note that the Euclidian distance under permutation is too time consuming for a straightforward calculation, since the runtime complexity increases with the faculty of the number of chosen covers. Therefore, we used the possibility of deriving this distance measure from the matching distance by employing the calculation via the Kuhn-Munkres algorithm, as described in section 4.4.2. Remember that this is achieved by using the squared Euclidian distance for comparing single feature vectors and drawing the square root from the result. The resulting plots (cf. Figure 4.7) look quite similar to the ones we derived from employing the minimal matching distance based on the normal Euclidian distance, i.e. using the multi-instance model (cf. Figure 4.8(c) and 4.8(d)). A careful investigation of the parts contained in the clusters showed that the cover sequence model using the minimum Euclidian distance under permutation and the multi-instance model lead to basically equivalent results. Due to this observation and the better possibilities for speeding up k-nn queries, we concentrated on the evaluation of the multi-instance model. We first compared the multi-instance model to the cover sequence model without permutations (cf. Figure 4.6). Furthermore, we used different numbers of covers for the multi-instance model (cf. Figure 4.8) in order to show the benefits of a relatively high number of covers for complex CAD objects.

Comparing the multi-instance model with the cover sequence model on the car data set (cf. Figure 4.6(a), 4.8(a), and 4.8(c)) we conclude that the multi-instance model is superior. All plots look similar on the first glance. When evaluating the clusters (cf. Figure 4.9(b) and 4.9(c)), it turned out that there are clusters which are detected by both approaches and thus appear in both plots, e.g. classes E in Figure 4.9(b) and 4.9(c). Nevertheless, we observed the following three shortcomings of the cover sequence model:

1. Meaningful hierarchies of clusters detected by the vector set model, e.g. G_1 and G_2 in Figure 4.8(c) which are visualized in Figure 4.9(c), are

lost in the plot of the cover sequence model (Class G in Figure 4.6(a) evaluated in Figure 4.9(b)).

2. Some clusters found by the multi-instance model are not found when using the cover sequence model, e.g. cluster F in Figure 4.9(c).

3. Using the cover sequence model, objects that are not intuitively similar are clustered together in one class, e.g. class X in Figure 4.6(a) which is evaluated in Figure 4.9(b). This is not the case when using the multi-instance model.

A reason for the superior effectiveness of the multi-instance model compared to the cover sequence model is the role of permutations of the covers. This is supported by the observations which are depicted in Table 4.1. In most of all distance calculations carried out during an OPTICS run, there was at least one permutation necessary to compute the minimal matching distance.

The plots in Figure 4.8(a) and 4.8(c) compare the influence of the number of covers used to generate the multi-instance representations on the quality of the similarity model. An evaluation of the clusters yields the observation that 7 covers are necessary to model real-world CAD objects accurately. Using only 3 covers, we observed basically the same three problems which we had already noticed when employing the cover sequence model for 7 covers.

All the results of the evaluation on the car data set can also be observed when evaluating the models on the Aircraft data set. As a consequence, the evaluation shows that the multi-instance model outperforms the other models with respect to effectiveness. Furthermore, we see that we need about 7 covers to model similarity most accurately.

4.5.3 Evaluation of the Efficiency

The most effective results on our test data sets were generated with $k = 7$ covers, entailing an average permutation rate of 99.0% (cf. Table 4.1). This observation leads to the conclusion that the cover sequence model can

No. of covers	Permutations
3	68.2%
5	95.1%
7	99.0%
9	99.4%

Table 4.1: Percentage of proper permutations.

only compete with the multi-instance model with respect to quality if all permutations are taken into account. Obviously, the multi-instance model using the minimal matching distance approach is much more efficient than the cover sequence model (one-vector model) using the minimum Euclidian distance under permutation.

To analyze the performance of the filter step that was introduced in section 4.4, we evaluated k-NN queries. Since the car data set consists of only some 200 objects, it is not suitable for efficiency evaluation. Thus, we ran our efficiency experiments on the aircraft data set only. We took 100 random query objects from the database and examined 10-NN queries. Our test machine was equipped with an INTEL XEON 1.7 GHZ processor and 2 GByte main memory. Since data and access structures fitted easily into the main memory, we calculated the I/O cost. One page access was counted as 8 ms and for the cost of reading one byte we counted 200 ns. The results are shown in Table 4.2.

It turns out that the filter step yields a speed-up of factor 10 on the CPU time, but suffers from a higher I/O-time. Nevertheless it provides a speed up factor of about 2 for total time. Furthermore, Table 4.2 demonstrates that the runtime using the multi-instance model with filter step is in the same order of magnitude as the one-vector model even without permutation. In our experiments, the multi-instance approach even outperformed the one-vector model in both CPU time and I/O time. Let us note that in our experiments we based the implementation of the one-vector model on the

Model	CPU time	I/O time	total time
1-Vect.	142.82	2632.06	2774.88
Vect. Set w. filter	105.88	932.80	1038.68
Vect. Set seq. scan	1025.32	806.40	1831.72

Table 4.2: Runtimes for sample 10-nn queries in s.

X-Tree [BKK96] which is penalized by the simulation of I/O time. Since it does not take the idea of page caches into account, an implementation of the one-vector model using the sequential scan exhibited a slightly better performance for some combinations of dimensionality and data set size, but the performance was still in the same order of magnitude.

4.6 Summary

In this chapter, we surveyed three feature transformations that are suitable for data mining on voxelized CAD data: the volume model, the solid angle model and the cover sequence model. The cover sequence model generates a set of covers of a three dimensional object that can be stored in a feature vector. In comparison to the other two models, it offers a better notion of similarity. A major problem of the cover sequence model is the order in which the covers are stored within the feature vector. For calculating the similarity of two objects, the order realizing minimum distance offers a better similarity measure, but is prohibitive in calculation cost. To represent an object as a set of feature vectors avoids this problem. Furthermore, it offers a more general approach for applications working with multi-instance objects. We described a metric distance measure on multi-instance objects, called minimal matching distance. Minimal matching distance is computable in $O(k^3)$. Furthermore, we introduced a highly selective filter step that is able to speed up similarity queries by the use of spatial index structures. To evaluate our system, we used two CAD data sets. To demonstrate the good notion of

similarity provided by the combination of the cover sequence model and the multi-instance representation, we applied hierarchical clustering to examine similarity measures. We evaluated the efficiency of the filter step using 100 sample 10-NN queries. It turned out that our new approach yields more meaningful results without sacrificing efficiency and thus a good foundation for distance based data mining on CAD-parts.

Chapter 5

Website Mining

The world wide web (WWW) is currently the largest source of information that is available to a broad public. Web content mining is concerned with applying the principles and solutions of KDD and data mining in order to extract specific knowledge from the WWW. Most established approaches of web content mining are concerned with the efficient retrieval of specific HTML-documents. In this chapter, we will introduce Website Mining as a new direction of web content mining that aims at the effective recognition and efficient retrieval of specific websites. A website is a set of HTML-documents that is published by the same person, group or organization and usually serves a common purpose. The chapter starts with providing a motivation and pointing out the advantages of treating websites as objects of interest. To distinguish relevant from irrelevant websites several methods of website classification are introduced. After describing two naive solutions, two more sophisticated directions of website classification are introduced. Finally, a novel focused crawler is introduced that is capable to efficiently retrieve relevant websites from the WWW.

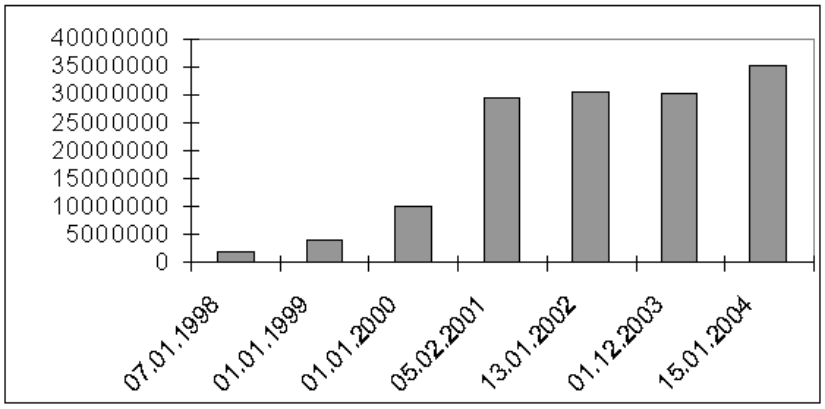


Figure 5.1: Numbers of registered international top level domains (.com, .net, .org, .biz, .info, .edu)[Pro] .

5.1 Motivation

In recent years the world wide web (WWW) has turned into one of the most important distribution channels for private, scientific and business information. One reason for this development is the relatively low cost of publishing a website. Compared to other ways like brochures or advertisements in newspapers and magazines, the web offers a cheaper and more up-to-date view on a business for millions of users. Thus, large numbers of companies, private persons and other organizations publish information via the WWW. As a result the WWW has been growing tremendously for the last five years. The search engine Google [Cen] recently reported that it is currently indexing over 4 billion text documents. Another statistic, demonstrating the enormous expansion of the WWW, is displayed in figure 5.1. According to [Pro], the number of registered international top level domains has increased more than 7 times over the last 5 years.

To find specific information in this vast amounts of information, there are several established solutions to retrieve interesting content from the WWW. Search engines like Google[Goo] download a large amount of webpages and index them with respect to the words occurring within them. To retrieve

webpages treating a specific content, a user provides one or several key terms that have to be contained in the wanted document. The search engine returns links to all documents containing these key terms. Since the number of webpages containing the key terms can exceed several thousands, modern search engines employ sophisticated ranking algorithms that try to ensure that the list of results starts with the most relevant pages.

A more novel approach to retrieve relevant webpages from the WWW are focused web crawlers [CvdBD99b]. A focused web crawler explores only a small portion of the web, using a best-first search guided by the user interest. Compared to web search engines, focused crawlers obtain a much higher precision and return new pages which are not yet indexed. We will give a more detailed introduction to focused crawling in section 5.4.

Another approach to answer user queries on the web are so-called web directories like Yahoo [Yah] or DMOZ [DMO]. The idea of a web directory is to organize important topics into a class hierarchy or taxonomy. For each of these topics, the web directory contains a number of links leading to relevant web content. The web content linked by a web directory might be a single HTML-document containing specific information, but more often web directories link to complete websites.

A **website** *is a linked set of webpages that is published by the same person, group or institution and usually serves a common purpose, e.g. to present a whole organization or company.*

Focusing on websites offers a more abstract view on the web and is useful for many applications. Companies are represented by websites and not by single webpages. Thus, companies that are looking for new costumers, suppliers or competitors, screen the WWW for interesting websites instead of single HTML-documents. For example, in the IT-business where products and services can change quickly, a system that spots special kinds of websites and offers the opportunity to search them will turn out to be very useful. Other reasons for focusing on whole sites instead of single pages are: There are much less sites than single pages on the WWW, reducing the search space

dramatically. The mining for whole websites offers a filter step when searching for detailed information. For example, when looking for the price of a new computer, it is very helpful to search only the websites of computer retailers instead of searching the whole WWW. One final reason is the higher stability of websites. Sites appear, change and disappear less often then single pages which might be updated daily. Thus, the retrieved or indexed knowledge is up-to-date for a longer time period.

The problem of spotting new websites of special interest to a user is not handled adequately yet. Though directory services like Yahoo [Yah] and DMOZ [DMO] are useful to find relevant websites for a listed topic, they have major drawbacks. Web directories offer in most cases only a very small portion of the websites that are relevant to a given topic. Furthermore, the categorization of the web directory might totally lack the topic a user is interested in. Last but not least, web directories might not be up-to-date due to manual maintenance.

To solve these problems, the area of website mining aims at applying KDD techniques to the retrieval and analysis of relevant websites. Therefore, the following chapter introduces solutions for two important applications of website mining. The first is the classification of websites which can be employed to maintain web directories automatically, increasing the recall of this established method for searching the web. The second application proceeds a step further and combines the technique of focused crawling and website classification to efficiently retrieve relevant websites with high accuracy.

The rest of the chapter is organized as follows. The next section provides general definitions. Section 5.3 will introduce the problem of website classification and surveys corresponding related work. Afterwards several approaches to website classification will be discussed and evaluated. Section 5.4 concludes the chapter by introducing a focused website crawler. First, this section will introduce focused crawling and discuss related work. Afterwards our new solution to retrieve relevant websites employing focused crawling is discussed in detail. At last, the introduced focused crawler is compared to

established methods of focused crawling that are applied to the retrieval of websites. The chapter concludes with a summary of the introduced techniques of website mining.

5.2 A Graph-Oriented View of the WWW

We identify a webpage p by its URL. Then, $content(p) \rightarrow \sigma \in \Sigma^*$ denotes the string we receive when trying to download p. Furthermore, we assume a feature transformation $FT : \Sigma^* \rightarrow T \subseteq\cong I\!N^d$ which transforms a string, for example, the contents of a webpage, into a d-dimensional feature vector. Let $\Lambda(p)$ be the set of all links $(p, q_1), (p, q_2), \ldots, (p, q_n)$ from p to $q_i \neq p$. The link (p, q) points from the source page p to the destination page q. Links within the same webpage are ignored. We define the webpage graph as a directed graph $G = (V, E)$ with V being the set of all existing webpages, extended by a special element which is needed to represent broken links, and E being the union of $\Lambda(p)$ for all $p \in V$.

A website is a linked set of webpages that is published by the same person, group or institution and usually serves a common purpose, e.g. to present a whole organization or company. Unfortunately, this intuitive definition is not well suited for automatic retrieval. Since there is no reliable way to find out who really published a webpage and to what purpose, there is no exact method to determine the webpages belonging to a certain site. Nonetheless, no one would deny the existence of websites. In order to recognize and retrieve relevant sites it is necessary to find a pragmatic definition that is suitable for the majority of cases. For our solutions, we benefit from the characteristic that a very large percentage of all websites is published under one dedicated domain or subdomain. In case a website is spread over several domains/subdomains, we do not lose any results, but may have some duplicates in the result set. However, if large websites are classified more than once, their chance of being part of the result increases as well. The other problem of our definition is the case that one domain hosts several websites.

Thus, websites without a domain of their own are not treated as separated websites. However, websites without a domain of their own are important in rare cases only and there is no search system on the WWW that can claim to achieve 100% recall.

Definition 5.1 (Website)
For each page p $host(p)$ returns the domain or subdomain of p, i.e. the substring of the URL of p between the protocol and the file section. We define a website W as a subgraph $W = (V', E')$ of the webpage graph with the following properties:

$$\forall u, v \in V' : host(u) = host(v)$$
$$\forall u \in V', v \notin V' : host(u) \neq host(v)$$
$$\forall (p, q) \in E' : p, q \in V'$$

We define the **homepage** as the webpage that is referenced by the URL, consisting of the domain name only, e.g. "http://www.cs.sfu.ca". Thus, each website has a unique homepage that can be accessed by knowing the website name. Compared to the webpage graph, the website graph which is the conceptual view of website mining onto the WWW has several important differences: We distinguish two different types of nodes at different levels of abstraction, page nodes and site nodes. We distinguish two different types of edges, representing inter-site links and intra-site links. Edges for intra-site links point to page nodes, but edges representing inter-site links point to site nodes. For a more formal definition, let $G = (V, E)$ be the webpage graph. We distinguish between intrinsic links (p, q) with $host(p) = host(q)$ and transversal links with $host(p) \neq host(q)$. Let U denote the union of V and let W be the set of all existing websites.

Definition 5.2 (Website Graph)
We define the website graph as a directed graph $WG = (U, D)$ with the set of edges D given as follows:

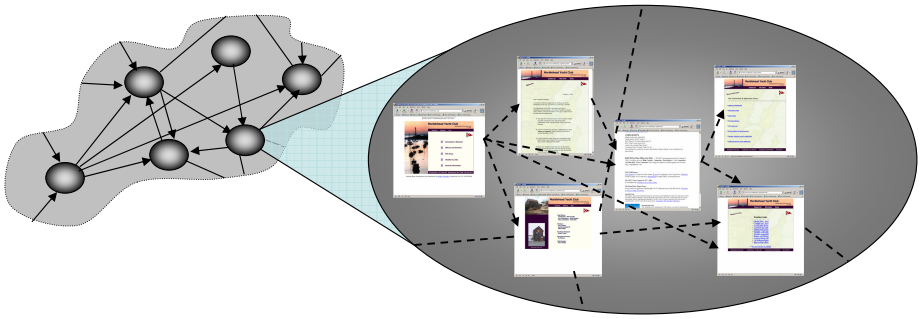


Figure 5.2: Sample portion of the website graph.

$$\forall (p,q) \in E' : host(p) = host(q) \Rightarrow (p,q) \in D$$
$$\forall (p,q) \in E' : host(p) \neq host(q) \Rightarrow (p, host(q)) \in D$$

Figure 5.2 depicts a small sample portion of the website graph consisting of three websites. Intrinsic links are represented by dashed arrows, transversal links by solid arrows.

5.3 Classification of Websites

Web directories like DMOZ [DMO] or YAHOO [Yah] provide thousands of classes organized in a hierarchy or taxonomy. To manually map new websites to the classes they belong to is a very time consuming task. However, most established web directories categorize their entries manually or semi-automatically. A classifier that automatically maps new websites into the class hierarchy would speed up the insertioninto a web directory. Principally, website classification is one of the key tasks of website mining, since it is necessary to distinguish relevant and irrelevant sites in order to retrieve relevant sites. Parts of this work were published in [EKS02] and [KS04].

5.3.1 Related Work on Website Classification

In this subsection, we briefly review related work on text classification, and on the classification of sequential data which is related to the classification of website trees. Text classification has been an active area of research for many years. The common approach to transform text documents to feature vectors is surveyed in chapter 2.1.2. After the feature transformation is applied, documents can be classified by any classification method. However, not all classifiers can cope with the sparse nature of text feature vectors very well. Classifiers that are reported to be well suited for text classification are naive Bayes [MCN98, YL99], support vector machines [Joa98] and centroid based kNN classification [HK00].

While most of the above methods have been applied to pure text documents, an increasing number of publications especially deals with the classification of webpages. Several authors have proposed methods to exploit the hyperlinks to improve the classification accuracy, e.g. [CDI98] and [CDF+99]. [CDF+99] introduces several methods of relational learning, considering the existence of links to webpages of specific classes. [CDI98] presents techniques for using the class labels and the text of neighboring, i.e. linked, webpages. However, all these methods aim at classifying single webpages not complete websites.

In one of the approaches to website classification, we will represent websites as so-called website trees and use the paths within these trees for classification. Therefore, we briefly survey methods for classification of sequence data. [DK02] discusses and evaluates several methods for the classification of biological sequence data, e.g. the kNN classifier, Markov classifiers and support vector machines. Whereas biological sequences tend to be very long, paths in a website tree are relatively short. Furthermore, in biological sequence classification the data are given and labelled a priori, whereas in website mining loading and labelling the data is an expensive procedure. Classification algorithms are difficult to apply to sequential data because of

the extremely large number of potentially useful features. [LZO99] proposes a sequence mining technique to act as a preprocessor to select features for standard classification algorithms such as naive Bayes. Several techniques have been introduced in the literature for efficiently determining the frequent sequences within some database, e.g. [Zak01]. However, these techniques only find the frequent patterns but do not build a classifier.

5.3.2 General Aspects of Website Classification

The classification of complete websites is in many aspects different from the classification of single webpages. Sites may strongly vary in size, structure and techniques. Another aspect is the used language. Many professional sites, especially in the non-English-speaking regions, are at least bilingual to provide international usability. Most page classification projects use only text documents in a single language which may prove insufficient when trying to handle whole sites.

To download a site from the web, the following algorithm can be applied. First, examine the homepage, since it is the only page that can be directly derived from the domain name . After reading it, we can use a HTML-parser to determine the links to the other pages within a site. Note that considering FRAME- and EMBED-tags as links is necessary to get a picture of a site that is as complete as possible. After link extraction, we follow every intrinsic link and explore the corresponding webpages in the same way as the homepage. It is necessary to mark the pages already visited, since a webpage might be reachable by following more than one link.

The most common way to classify single HTML-documents is to use naive Bayes classifiers [YL99] or support vector machines [Joa98] on a feature space of terms. Here the quality of the results depends highly on the right choice of terms. Thus, we employ the techniques introduced in chapter 2.1.2 to find a good selection. Another interesting possibility is to expand the feature space to include structural components. The number of words and images,

the occurrence of forms or frames or the number of links from a page can offer vital information depending on the specified classes.

Website classification is task to map a complete website W to the element c_i of a determined set of website classes C which describes the purpose of W in a best possible way.

5.3.3 Naive Approaches to Website Classification

The simplest way to classify websites is to apply established techniques of webpage classification to the homepage of a website. This approach is simple and efficient, but it depends on the assumption that the homepage contains the information to identify the purpose of the complete website. Unfortunately, this assumption does not hold for many real-world websites. A homepage might only consist of structural information, e.g. frame tags, or provide not much text. Furthermore, the homepage of many websites provides only an introduction, but does not describe the purpose of a website. Last but not least, the purpose of a website may not be given by the contents of a single webpage within this site. If no single document contains information about the purpose of the website, it has to be discovered by examining several pages. Thus, to make reliable class predictions for real-world websites, it is necessary to employ more than one webpage of the website.

Another way of classifying a website is to apply the methods used for page classification to our definition of websites. We just generate a single feature vector, counting the frequency of terms over all webpages of the whole site, i.e. we represent a website as a single "superpage". Therefore, we call this simple approach "classification of superpages". The advantage of this approach is that it is not much more complex than the classification of single pages. The user just have to walk through the nodes of the site and count terms. Afterwards the vector can be classified by any standard data mining package, e.g. the weka-package [WF99] we used in our experiments. However, the superpage classifier has several conceptual drawbacks. The

approach is very sensitive to the right selection of key terms. As mentioned before, sites can contain documents in several languages. Structural features like the occurrence of frame tags lose most of their significance. Another very important problem is the loss of local context. Keywords appearing anywhere within the site are aggregated to build up a bag-of-words view of the whole website. As shown in the evaluation (section 5.3.7), this simple classifier achieved insufficient accuracy in most experiments.

5.3.4 Classification using Page Classes

The main reason why the superpage approach does not perform well is the fact that it makes no difference between an appearance within the same sentence, the same page or the same site. However the context plays an important role because sites can spread over several thousand single HTML-documents, containing information about various topics. For example, the meaning of the terms "network administration" and "services" on the same page implies that this company offers network administration as one of its services. But without the constraint that both terms appear on the same page, the implication is much weaker. Any company offering any service and looking for a network administrator will provide those terms, too.

To overcome these problems, we need more natural and more expressive representations of websites. In this section, we introduce two kinds of such representations. Then, we present two advanced methods of website classification based on these representations. For the rest of this section, we use the discovery of potential customers, competitors or suppliers as our running application. However, all the proposed methods for website classification are not restricted to corporate websites and have a much broader range of applications.

Representations of Websites

Compared to the superpage approach, we change the focus of site classification from single key terms to complete HTML-documents. In order to summarize the content of a single webpage, we map the webpage to a so-called page class. A page class represents a certain type of webpage that is likely to appear in a certain type of website. Since the terms only influence the page class of a webpage, the local context is preserved. Actually, the preprocessing step can use all mentioned techniques for the selection of terms introduced for webpage classification without any restriction. To conclude, we introduce page classes besides the website classes and label each webpage within a website with the most likely page class.

The choice of page classes for our experimental evaluation was based upon the observations that we made during the examination of many business sites in several trades. Although their trades varied widely, the following categories of pages are to be found in most classes of business-sites: company, company philosophy, online contact, places and opening hours, products and services, references and partners, employees, directory, vacancies and other. The "other"-category stands for any topic not specified more precisely. Note that we used these page classes only for the purpose of illustration, but our method is generally applicable for any website class as well as any set of webpage classes. Since the features representing a page class will vary from trade to trade, every category except "other" has to be specialized for each trade we want to investigate. For example, we distinguished between the products and services of a florist and an IT-service provider (our examples in the evaluation).

To determine the page class of a given webpage, we use text-classification on terms, using naive Bayes classification. Since there is always a classification error for each page, the probability that the complete graph is correctly labelled is rather low. But the average number of correctly labelled nodes is about the mean classification accuracy of the page classification. This can be

shown when treating the problem as a Bernoulli chain. We will soon discuss the impact of this effect on our main classification problem. Based on the labelled pages of a website, we propose the following representations of a website:

- *Feature vector of topic frequencies*
  Each considered page class defines a dimension of the feature space. For each page class or topic, the feature values represent the number of pages within the site. This representation does not exploit the link structure of the site, but it considers a website as a set of labelled webpages. In other words, we treat a website as a multi-instance object and use a webpage classifier to condense this set into a single feature vector.

- *Website trees*
  To capture the essence of the link structure within a site, we represent it as a labelled tree. The idea is that the structure of most sites is more hierarchic than network-like. Sites begin with a unique root node provided by the homepage and commonly have directory-pages that offer an overview of the topics and the links leading to them. Furthermore, in most sites the information in the area around the homepage is very general and gets more and more specific with increasing distance. For example, we observed that pages regarding the company itself are found more often only a few links away from the homepage than ones about specific product features.

For building website trees, we use the minimum number of links as a measure of distance between two pages of a site. To construct a website tree the minimal paths from the homepage to every page in the website are joined. Therefore, we perform a breadth-first search through the graph of a website and ignore the links to pages we already visited. Note that in the case of two paths of equal length leading to the same webpage, the path occurring

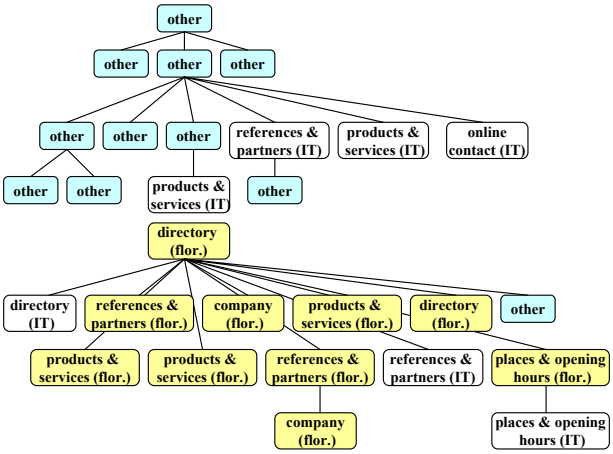


Figure 5.3: Example of website trees. A typical small IT-service provider (above) and a typical small florist site (below).

first is chosen. Though there is no way to tell which path preserves more information, this definition was made to make tree derivation deterministic. The trees in figure 5.3 are generated by this method.

Classification of Topic Frequency Vectors

After the transformation of websites into topic frequency vectors, most general classifiers such as Bayes classifiers and decision tree classifiers are applicable. Especially tree classifiers like C4.5 [Qui93] showed an enormous improvement of classification accuracy compared to the simple superpage approach. Let us note that the dimensionality of the topic frequency vectors is much smaller than the dimensionality of the term frequency vectors which are used in the superpage approach.

Classification of Website Trees

In this paragraph, we present a method of website classification based on the website tree representation, i.e. exploiting the links within a site. Our method is based upon the idea of Markov chains in combination with Bayesian

classification. Therefore, we first follow the Bayesian decision rule:

$$c^* = maxarg_{c_i \in C} Pr[c_i|S] = maxarg_{c_i \in C}(Pr[c_i] \cdot Pr[S|c_i])$$

Here the predicted class c^* of the site S is the class c_i that explains the occurrence of the given site S best. Due to the Bayesian rule, the probability $Pr[c_i|S]$ is the product of the a priori probability $Pr[c_i]$ and the probability $Pr[S|c_i]$ that the class model for c_i constructed the website tree of S. Therefore, we estimate $Pr[c_i]$ as the relative frequency of websites in the class c_i. The approximation of $Pr[S|c_i]$ depends on the chosen model.

The concept of k-order Markov chains is applied to website trees using the following procedure. Beginning with the probability for the label of our root node we multiply the probabilities of the transition between the k last nodes and their successor. Note that these transition probabilities only use the static link structure of the webpages. They do not use any dynamic click-through probabilities. In the simple case of 1-order Markov chains, the transition probability for the page classes pc_i and pc_j with respect to site class c_l is the probability that within a website of class c_l a link from a page belonging topc_i is directed at a page belonging to pc_j. Since there can be more than one successor in the tree, we multiply the transition probabilities for every child node, traversing along every path to a leaf node. Let us note that the probability for reaching each node is accounted only once. This Markov tree model is very similar to the concept of branching Markov chains [MV97], but does not take branching probabilities into account.

The statistical process to calculate the probability $P[S|c_i]$ is the following: Let P be the set of page classes extended by the element "none". The "none"-class acts as a fill-in for paths shorter than k and is used to simplify the calculation. Furthermore, let pc be the label of a node t and let c_i be the i-th site class. The function pre (with $pre(k,t) = pc$) returns the page class pc of the k-th predecessor of the node t with respect to the website containing t. If there is no such predecessor, it returns "none". Note that the predecessor is uniquely defined because we have a tree structure. Then the conditional

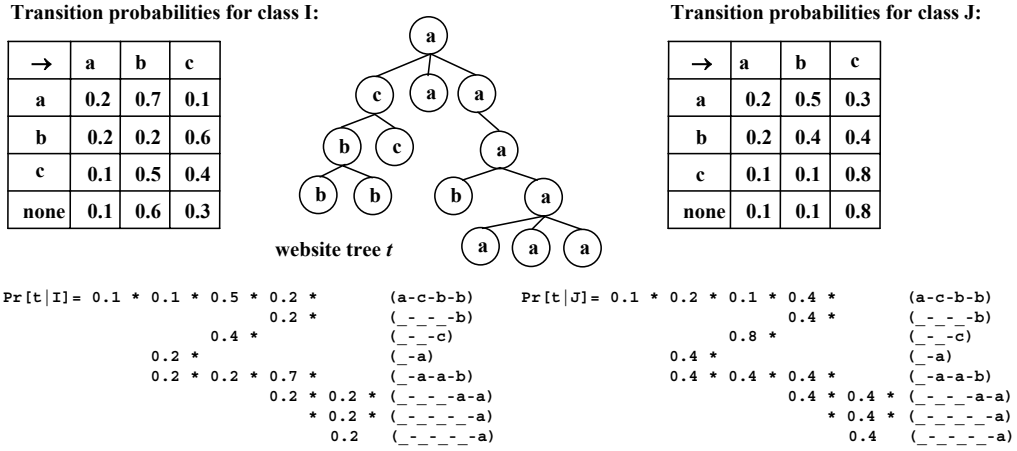


Figure 5.4: The calculation of the model probability $Pr[t|c_i]$ for two site classes (I and J) and three page classes (a, b, c).

probability of the website tree S can be calculated as:

$$Pr[S|c_i] = \prod_{t \in S} Pr[pc_t|pre(k-1,t), \ldots, pre(1,t)]$$

Thus, for every node t in the site tree S, the probability that its label pc_t occurs after the occurrence of the labels of its k predecessors is multiplied. Figure 5.4 visualizes the calculation of $Pr[S|c_i]$ for two site classes.

This method does not use the possible correlations between siblings and thus, the context taken into account is limited to the path from the homepage. To estimate the transition probabilities, we calculate the mean distribution of the occurring transition of length k. Note that this is a difference to the relative occurrence of the transitions in class c_i. Due to the classification error in the preprocessing step (when assigning page classes to the webpages), the probability for "phantom transitions" that are generated accidentally is rather high. Especially in site classes where the average number of pages is rather high, the absolute number of certain transitions can accidentally match the number in site classes trained on sites having fewer pages. Thus, a problem appears when the matched transition is highly specific for the class consisting of smaller sites. In this case, the significance is distorted.

To smooth this effect, the mean distribution uses the size of a site as a measure for the number of transition occurrences and normalizes the number of transition occurrences within a site to one. Therefore, the appearance in a site class for which the total number of transitions is higher is given less importance. Note that the information about the number of pages within a site is not taken into account. The use of the mean distribution proves to be superior in every approach based upon the preprocessing step in all experiments.

For the choice of the degree k of the Markov chains, we tested the values zero, one and two. According to our results and a consideration discussed in the evaluation, a higher degree was not reasonable. For every choice of k, this model yields comparably good results to the standard classifiers applied to the distribution vectors. For $k = 0$ it even outperformed the other approaches.

Since the concept of the 0-order Markov tree shows similarities to the naive Bayes classifier applied to topic frequency vectors, we examined their differences more closely. For a better understanding of the following comparison, we will present the calculation of the single probabilities in a 0-order Markov tree.

$$Pr[S|c_i] = \prod_{t \in S} p^{c_i}_{pageclass(t)} = (p_1^{c_i})^{r_1} \cdot \ldots \cdot (p_k^{c_i})^{r_k}$$

given $\sum_{j \in L_i} = |S|$

Here, S is the site to be classified, c_i a site class and $p^{c_i}_{pageclass(t)}$ is the probability of the occurrence of the page class for page t in class c_i. Furthermore, let r_j be the number of occurrences of the topic $j \in PC$ in the site S and let PC be the set of page classes. Thus, the probability is calculated by taking the r_j-th power of every page class probability and then multiplying those factors for every topic. This is equivalent to a multinomial process except for the difference that the multinomial coefficient can be neglected due to its equal occurrence in every class c_i.

To explain the different results compared to naive Bayes, the following differences can be pointed out. Naive Bayes considers the occurrence of a topic to be independent from the occurrences of the other topics. But since naive Bayes calculates the probability of the total number of occurrences within the site, the probability of just one occurrence is not independent from other occurrences of the same topic. Depending on the used distribution model for one dimension, a further occurrence of a topic that is very typical for a certain site class will even decrease the probability for that class if the number of occurrences differs strongly from the estimated mean value. On the other hand, the 0-order Markov tree always increases the conditional probability $Pr[S|c_i]$ if a further page class specific to the site class c_i occurs in the site S. A further important difference is the consideration of the number of total pages of a site. Since a large number of occurrences will automatically decrease the number of occurrences in all other page classes, the 0-order Markov tree uses additional knowledge compared to naive Bayes.

A further interesting property of 0-order Markov trees is the possibility to calculate the probabilities incrementally. For every disjunctive segmentation $(s_1, \ldots, s_m)$ of our site S, the following equation holds:

$$\prod_{s_j \in S} Pr[s_j|c_i] = Pr[S|c_i]$$

In other words, if the probability $Pr[s_j|c_i]$ for the class c_i is higher than for any other class, the subset s_j will increase the probability that $Pr[S|c_i]$ is also higher than for any other class. This property will be useful in section 5.3.6.

5.3.5 Classification without Page Classes

Though transforming a website into a labelled website tree enables us to classify websites with high accuracy, it demands expensive preprocessing. To train the page classifier, it is necessary to determine a set of page classes characterizing each website class. Afterwards additional effort has to be

spent to label a sufficient set of training documents for each page class to train the page classifier. For most practical applications, a less expensive solution for training a website classifier has to be found in order to make website classification applicable.

Turning away from the concept of page classes leaves us without an appropriate feature transformation for websites. Thus, there is no feature space that most of the well-established classification methods like Bayes classifiers or SVMs require. Therefore, we adopt the paradigm kNN classification that only assumes a pairwise distance function. For the classification of an unknown object, a basic kNN classifier performs a kNN query on the training database and predicts the most frequent class in the result set. The key to the effectiveness of kNN classification is an intuitive distance function. Since the content of each single page $p \in S$ can be represented by a feature vector of term frequencies, a whole website is represented by a multi-instance object. Several distance measures for sets of vectors in a metric space have been introduced in the literature [EM97, RB01]. From these distance measures, the Sum of Minimum Distances (SMD) [EM97] most adequately reflects the intuitive notion of similarity between two websites. In the context of website classification, it can be defined in the following way.

Definition 5.3 (Sum of Minimal distances (SMD))
Let S_1 and S_2 be two websites and let $FT : P \rightarrow \mathbb{R}^d$ be a feature transformation that returns the feature vector of a page $p \in P$, where P is the set of all webpages. Furthermore, let $d(x, y)$ be a distance measure on feature vectors. The SMD of S_1 and S_2 is given by:

$$SMD(S_1, S_2) = \frac{\sum_{v_i \in S_1} \min_{w_j \in S_2} d(FT(v_i), FT(w_j)) + \sum_{w_j \in S_2} \min_{v_i \in S_1} d(FT(v_i), FT(w_j))}{|S_1| + |S_2|}$$

The idea of SMD is to map every element of both sets to the closest element in the other set. This means that several pages belonging to the

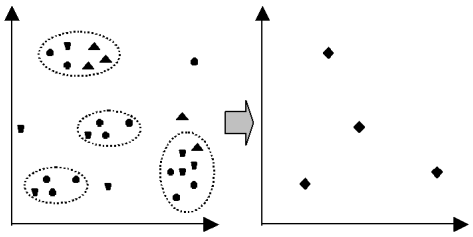


Figure 5.5: Centroid set of a sample website class.

website S_1 might be mapped to the same page in the other site S_2 or vice versa. This is quite adequate for websites because the number of different webpages describing the same information may vary among different websites belonging to the same class. Furthermore, sites of very related content but varying size will become very similar w.r.t. SMD since the cardinality of the set is not considered. The SMD is reflexive and symmetric, but does not fulfill the triangle inequality, i.e. it is not metric. The SMD distance calculation for a pair of websites S_1 and S_2 has a quadratic runtime complexity $O(w^2)$, where w denotes the maximum of the numbers of webpages of S_1 and S_2. As distance measure between the single feature vectors we use the cosine coefficient (compare chapter 2.1.2) which is well-established for text data.

Improving Efficiency Using Centroid Sets

Though the basic kNN classifier is very accurate, the computational cost for classifying a website is very high. The standard approach of speeding-up kNN queries by using a multi-dimensional index structure such as the M-tree [CPZ97] is infeasible because SMD is not metric. An alternative to speed up classification is to reduce the size of the training database to one representative per website class.

Therefore, we introduce centroid sets to summarize and represent a website class. The idea is that each website class provides several groups of webpages that are somehow related and can be summarized by one common representative. Let us note that these groups of pages are somewhat similar to

the manually assigned page classes to the TFV and Markov tree approaches, but refer to one site class only and are derivable without manual interaction. In [HK00], the authors show that the centroid of several text documents is a useful representative for a complete class in terms of kNN- classification. Furthermore, the paper mentions clustering to treat multi-modal classes in its conclusion chapter. However, to our knowledge the authors did not follow this direction any further. To speed up website classification and find meaningful descriptions of websites, we take up this idea.

Given some groups of related elements, we calculate one mean vector for the training pages of each group and define the centroid set for a website class as the set of all such mean vectors.

Definition 5.4 (Centroid Set)
Let S be a set of sets s_i with vectors v_{j,s_i} and let $\pi_l(s_i) = \{v|g(v) = l\ \forall v \in s_i\}$ be the restriction of s_i to group l, where g is a mapping from a vector v to a group $l \in G$, the set of all groups. Then the centroid set CS of S is defined as:

$$CS(S) = \left[c_j \mid \forall j \in G, c_j = \frac{1}{|\bigcup_{\forall i} \pi_l(s_i)|} \cdot \sum_{x \in \bigcup_{\forall i} \pi_l(s_i)} x \right]$$

Figure 5.5 illustrates the centroid set for a sample website class using a two-dimensional feature space for the webpages. The remaining problem is now to determine the grouping within the training pages of a website class. Fortunately, the task of identifying similar groups of instances within a database of feature vectors is known as clustering. Though there are many established clustering algorithms, the choice of a suitable algorithm for our problem is limited by two requirements. First, the number of clusters should be determined by the clustering algorithm. Since there is no a priori knowledge about the groups within a site class, we are unable to input the number of clusters. Second, the cluster algorithm should be able to deal with noise.

In our context, noise represents webpages that are uncommon for the class of websites they occur in. To provide relevant generalization, noise should not be considered within the constructed centroid set. We choose the density based clustering algorithm GDBSCAN [SEKX98] to group the training pages within each website class, because of its ability to find an arbitrary number of clusters and to filter out noise. The parameters of GDBSCAN are used to adjust the number of centroids per class and to control how much noise is eliminated. Thus, the centroid set for a website class c_i is derived as follows:

1. Join the (feature vectors of the) webpages found in the training websites of class c_i into one set.
2. Determine clusters in this set of feature vectors using GDBSCAN.
3. For each cluster, calculate the centroid and insert it into the centroid set of class c_i.

Incremental Distance Calculation

When using this compact representation of website classes, the website classifier has to calculate the SMD between a test website and all website classes, each represented by a centroid set. Now the following problem occurs especially when classifying objects belonging to the obligatory "other" class that is trained on a random mix of different types of websites that are not further distinguished. Consider a website, only consisting of a few webpages that can even be contained in the training set. The portion of the SMD that sums up the distance of the webpages of this website to the centroid set will be rather small because the website is part of the training set. However, the second sum consisting of the distances of all centroids to the few specialized pages in the website will be very large, since we might have many other topics in the representative object that are not obligatory for websites belonging to the site class. This contradicts our intuition because an instance belonging to a class should be very similar to the class representative. To avoid this effect,

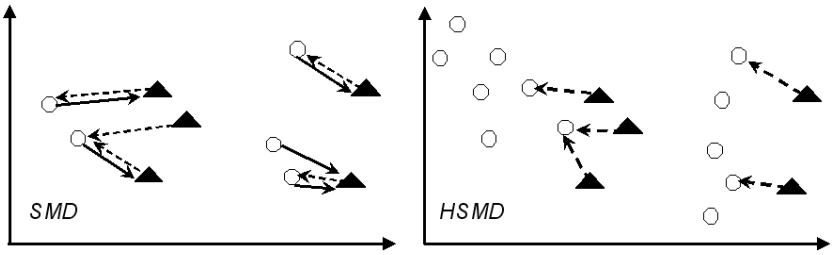


Figure 5.6: Illustration of SMD (left) and HSMD (right).

we replace the SMD by the half-SMD (HSMD) as a more adequate distance measure for calculating the distance of test websites to centroid sets.

Definition 5.5 (Half-Sum of minimal Distances (HSMD))
Let S be a website, let CS be a centroid set and let $FT : P \rightarrow \mathbb{R}^d$ be a transformation that returns the feature vector of $p \in P$ where P is the set of all pages and centroids respectively. Furthermore, let $d(x, y)$ be a distance measure on feature vectors. The HSMD of S to CS is given by:

$$HSMD(S, CS) = \frac{\sum_{v_i \in S} \min_{w_j \in CS} d(f(v_i), f(w_j))}{|S|}$$

Figure 5.6 illustrates the calculation of SMD and HSMD. In the following, we will call the NN-classifier using centroid sets and HSMD the *centroid set classifier*.

A further advantage of HSMD is a faster calculation, especially for incremental classification. Classifying a website incrementally using the introduced methods of NN-classification is basically possible for all variants mentioned above. Since the view of a website as a set of pages allows us to treat the already retrieved part as a website, the distances can be calculated for each subset, too. However, the variant using centroid sets and HSMD is suited best for incremental classification, due to the following reasons. By limiting the training set to just one instance per site class, we can store the HSMD values of the subset retrieved so far with each centroid set. Since

the HSMD only considers the distance from the page to its nearest neighbor in the representative object, the distance can be summed up during the traversal. Thus, the effort for extending the classification to an additional webpage is limited to one NN-query for each class. This ability will be very useful in the next section where we will turn to the reduction of the number of webpages examined for website classification.

5.3.6 Pruning the Irrelevant Area of a Website

The efficiency of website classification crucially depends on the number of downloaded webpages, since the download of a remote webpage is orders of magnitude more expensive than in-memory operations. Therefore, we introduce a classification method, downloading only a small part of a website, which still achieves high classification accuracy. This method performs incremental classification and stops downloading additional pages when an area around the homepage is visited that is likely to be a good representation of the purpose of a website. The existence of such an area is very likely due to the hierarchical design of most sites. The challenge is to detect a reasonable border of this area.

For the following reasons, the naive approach of reading the first n pages of a website does not yield a good accuracy. First, the topology of a website is a matter of individual design and therefore tends to be very heterogeneous. Many sites contain large amounts of pages providing only structure but no content. For example, animated introductions or frames are instruments of making a site more usable or attractive, but in most cases they do not contain any content recognized by a page classifier. Another important aspect is how much content is provided on a single page. The same amount of information could be spread over several pages or be contained in one large webpage. Consequently, the total number of pages already read is not a good indicator for pruning a website.

The homepage is always the first page to be read when traversing a web-

site. Additional pages are found by following the links on the homepage. Therefore, the question is, how to traverse the website and where to stop. A breadth-first traversal (used already to build the website trees) seems to be a promising approach. Since this traversal strategy orders the pages with respect to their distance to the homepage, the more general and therefore more important pages are visited first. Thus, the traversal offers a reasonable guidance for the classification process. The key to efficient classification is to prune certain subgraphs or subtrees in the graph of the website. Note that the site tree is derived from the graph of a website during classification and that its topology depends on such a pruning criteria. Thus, a node can only be ignored when every path leading to it is pruned. Therefore, the trees derived by the breadth-first traversal in combination with pruning can vary from those derived by a pure breadth-first traversal.

Our pruning method is based on the following two propositions about the paths from the homepage to the following pages:

- **Case I:** The membership of a complete path in some site class strongly depends on the pages closest to the homepage. As mentioned before, general information about the class of a website is most likely placed within a few links from the homepage. If a new topic follows, it appears in the context of the former topic.

- **Case II:** There are cases where a whole subtree and the path leading to it does not show any clear class membership at all. Though it is obvious to a human user that its impact on classification is rather low. Recognizing this kind of subtree is a difficult problem. A tree could always become highly specific after the next node. But after a reasonable length of the path, the probability that the meaning of the subtree is of general nature is significantly decreasing. Therefore, the strength of the class information has to be measured by the length of the path.

To exploit these propositions, it is necessary to measure the degree of

class membership for a path and its impact on site classification. Here the ability of a website classifier to incrementally calculate the class membership is very useful. The 0-order Markov tree and the centroid set classifier can calculate the probability for the occurrence of a path for each class although they might be trained on complete sites. To derive class probabilities from the centroid set classifier, we used the inverse distance to the centroid set of each class $1 - HSMD(S, CS_i)$ as class probability $Pr[s|c_i]$. Let us note that pruning can be applied to the other classifiers as well. However, for some classifiers the already examined part of a website has to be reevaluated after each step, which might be very time consuming. Therefore, we applied pruning to the methods of both cases that performed best. The 0-order Markov tree when using page classes and the centroid set classifier for the case without page classes.

The conditional probabilities $Pr[s|c_i]$ yield the information about the degree that a path s supports a site class c_i. Since the focus lies on the importance of a path for site classification, the actual class or classes it supports are not relevant. To quantify the importance for the complete site, we use the variance of the conditional probabilities over the set of all website classes. Since the variance is a measure for the heterogeneity of the given values, it mirrors the ability of a path to distinguish between the different site classes. A high variance of the probabilities of the website classes indicates a high distinctive power of that particular path. Let s be any path in the site tree S and let $Pr[s|c_i]$ be the probability that s will be generated by the model for class c_i, then

$$weight(s) = variance_{c_i \in C}(Pr[s|c_i]^{\frac{1}{lenght(s)}})$$

which is a measure for the importance of the path s for site classification. Let $length(s)$ be the number of nodes in path s. The $(1/length(s))$th-power is taken to normalize $weight(s)$ with respect to $length(s)$. This is necessary for comparing weights of paths of varying length.

To determine $weight(s)$ according to the above propositions (cases I and

II), we have to show that we can recognize a change in the class membership and recognize the occurrence of unimportant paths. The last requirement is obvious since a low variance means that the path is treated similar by the model of any class. The first requirement is not as easy to fulfill, but is provided with high probability after a certain length of the path is reached.

With increasing $length(s)$, $Pr[s|c_i]^{\frac{1}{length(s)}}$ becomes less sensitive to the multiplication of a further factor within in the calculation of $Pr[S|c_i]$. The chance of a single node changing the predicted class and keeping up the variance at the same time therefore decreases with increasing $length(s)$. An additional webpage that is more likely to be found in a site class different from the currently predicted class of s, will most likely decrease the $weight(s)$. Thus, after a few nodes on a path s a decreasing value of $weight(s)$ indicates a changing topic. Now our first proposition can be applied, i.e. the path can be pruned.

Due to the important role $length(s)$ plays for estimating the importance of the observed path s, it is an essential ingredient for the pruning criterion. Let s_1 and s_2 be paths within the site tree S where s_2 is an extension of s_1 by exactly one node. Then we stop the traversal at the last node of s_2 iff:

$$weight(s_2) < weight(s_1) \cdot \frac{length(s_2)}{\omega}$$

where $\omega \in \mathbb{R}^+$.

For suitable values of ω ($\omega \geq 3$), the criterion will very likely allow the extension of shorter paths which should not be pruned for the following reasons. Due to the small number of nodes, the membership can strongly be influenced by the classification error of the preprocessing step. Furthermore, our weight function cannot recognize a change in the membership in very short paths. In addition, the question of the importance of those paths for site classification cannot be decided in such an early state. Thus, applying the pruning rule makes no sense until some nodes are descended along every path. Figure 5.7 illustrates the proposed pruning method on a small sample website tree and a 0-order Markov tree classifier. In particular, it shows

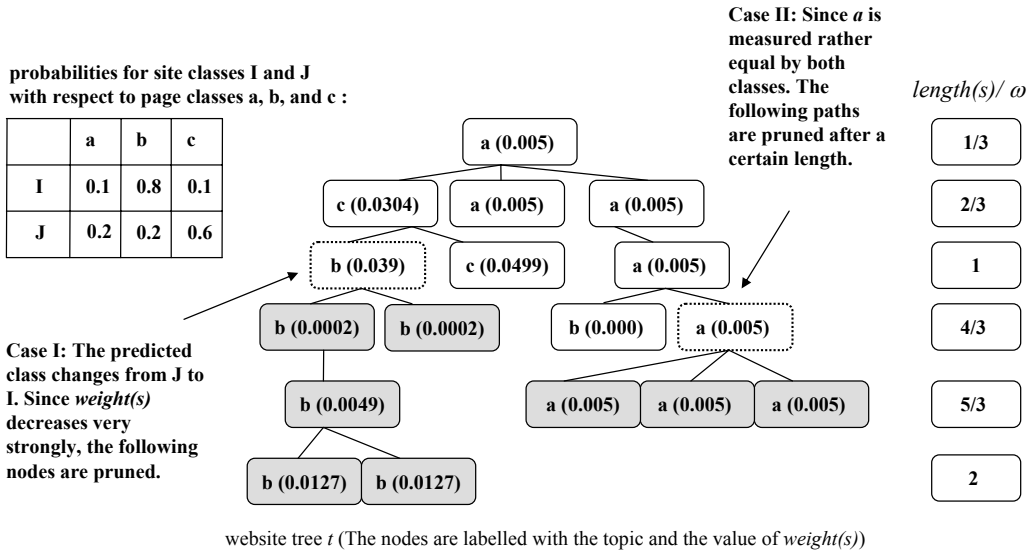


Figure 5.7: The effects of the pruning method on the 0-order Markov tree classifier with $\omega = 3$. The dashed nodes are to be pruned.

the $weight(s)$ for each path s. For a path s with $length(s)$ smaller than ω, this rule will stop the traversal only if a relevant decrease in variance is observed. As mentioned above, this is interpreted as a change of the site class and we can prune any following nodes due to our first proposition. For $length(s) \geq \omega$, the criterion is likely to prohibit the extension of a path unless a topic occurs that can significantly raise the variance. With increasing $length(s)$ it is more and more unlikely that an additional factor can increase the variance strongly enough. Due to the required growth of variance and the decreasing influence of the additional factor, most paths are cut off after a certain length. This corresponds to the requirement made by our second proposition that a path will not provide general information about the class of the website after a certain length. Hence, we avoid reading large subtrees without any impact on site classification.

The parameter ω is used to adjust the trade-off between classification accuracy and the number of downloaded webpages. Since the tolerance for the change of $weight(s)$ depends on the ratio $\frac{length(s)}{\omega}$, ω is the length from

which an extension of the path is only permitted if the variance increases. Thus, a good estimate for ω is the distance from the homepage in which the relevant information is assumed. Our experiments will show that the interval of reasonable values for ω is relatively wide.

Pruning does not only increase the efficiency of website classification, but it can also improve the classification accuracy. When classifying a complete website, all introduced methods (with the exception of Markov trees with $k \geq 1$) consider all pages equally important and independently from their position within the site. Thus, unspecific subtrees can drive the classification process into the wrong direction. By providing an effective heuristic to disregard areas that are unlikely to contain the relevant information, the classifier gets a better description of the website and therefore it will offer better accuracy. To conclude, the introduced pruning rule tries to cut off misleading areas from the website tree and thus can reduce the processing time and also increase the classification accuracy.

5.3.7 Evaluation of Website Classifiers

This section presents the results of our experimental evaluation of the proposed methods of website classification. Our classifiers were tested on two scenarios.

First, we will focus on the case that page classes and corresponding training pages are available. We compared the accuracy of the introduced classifiers and examined the performance of the introduced pruning method and its parameter ω. In the second part of our evaluation, we will turn to the case that no page classes are specified. In this scenario, we also compared the classification accuracy and examined the classification time of kNN-based classifiers, since this is a general weakness of this direction of classification algorithms. The classifiers were implemented in Java 1.3 and were tested on a workstation equipped with 2 Pentium 4 processors (2,4 GHZ) and 4 GB main memory.

classifier	accuracy	other		IT serv.prov.		florists.	
		pre.	rec.	pre.	rec.	pre.	rec.
naive Bayes (superpage)	55.6 %	0.80	0.32	0.48	0.89	0.57	0.62
naive Bayes (homepage)	63.0 %	0.70	0.53	0.65	0.68	0.54	0.73
2-ord. Mark. tr.	76.7 %	0.73	0.92	0.85	0.62	0.83	0.48
centroid set Cl.	77.1 %	0.76	0.87	0.80	0.70	0.75	0.45
naive Bayes (TVF)	78.7 %	0.74	0.95	0.88	0.61	0.92	0.57
1-ord. Mark. tr.	81.7 %	0.79	0.95	0.85	0.56	1.00	0.92
C4.5 (TVF)	82.6 %	0.80	0.90	0.83	0.73	1.00	0.76
0-ord. Mark. tr.	86.0 %	0.83	0.94	0.89	0.76	1.00	0.81
0-ord. Mark. tr. (pruned)	87.0 %	0.84	0.94	0.96	0.77	1.00	0.86

Table 5.1: Accuracy for the first testbed using 10-fold cross-validation.

Experiments using Webpage classes

Our first testbed provides page classes and consists of 82,842 single HTML-documents representing 207 websites. For the considered trades, we chose florists and IT-service providers to have a significant distinction in the business. The distribution of the website classes was: 112 "other", 21 "florist" and 74 "IT-service provider". The websites for the other class were taken randomly from various categories in Yahoo [Yah]. To make the experiments reproducable, the downloaded information was stored locally. To classify the pages into the page classes listed in section 5.3.4, we labelled about 2% of the pages in the testbed and obtained a classification accuracy of about 72% using 10-fold cross-validation with naive Bayes on the manually labelled pages. As implementation for this and the rest of the standard algorithms, we used the well-implemented weka-package [WF99]. The remaining 98% of the pages were labelled by the naive Bayes classifier based upon this training set.

Table 5.1 shows the overall classification accuracy as well as precision and

recall for the single site classes. Since the superpage approach provided only an accuracy of about 55%, it seems not to be well-suited for website classification. Webpage classification of the homepage using naive Bayes (homepage naive B.) performed similarly bad by achieving only a classification accuracy of 63%, which underlines the assumption that more pages than the homepage are necessary for accurate website classification.

All approaches based on the preprocessing step (introducing page class labels, etc.) obtained reasonable results. The best method using the complete website turned out to be the 0-order Markov tree which yielded 3.4% more classification accuracy than the C4.5 [Qui93] decision tree classifier on TFVs. It also clearly outperformed the 1-order Markov tree by 4.3%. As a comparison the 0-order Markov tree, applying the introduced pruning method, increased the accuracy by one percent to 87% by reading only 57% of the data. To compare the methods using page classes with those ones that do not, we additionally applied the centroid set classifier which is the best performing type of this direction to this testbed. Though the centroid set classifier offered reasonable results as well, it was outperformed by the 0-order Markov tree by about 10%. Thus, employing page classes increases the effort spent on preprocessing, but is likely to increase the classification accuracy as well.

Our experimental evaluation demonstrates that using higher values than 0 for the order k did not improve the results when applying a Markov tree classifier. This is due to the following reasons. First of all, the above mentioned problem of "phantom paths" increases with the length of the considered context (represented by the order k), depending on the error rate of page classification. We already noted that the overall error rate p of wrongly recognized pages in the site is about the same as the classification error for the single pages. But the probability of a correctly observed transition is only $(1-p)^2$, since it takes two correctly classified pages to recognize a transition. This problem gets worse with increasing order k. Thus, a distribution based upon observed transitions will model reality only poorly. A further

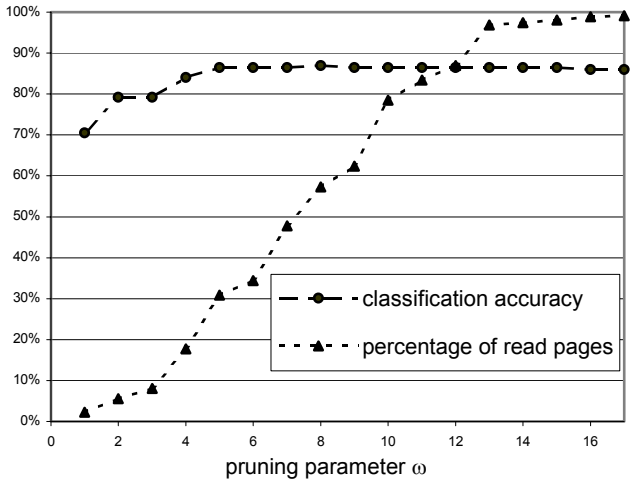


Figure 5.8: Effect of the parameter ω on the classification accuracy and the percentage of downloaded webpages.

reason is caused by the specific characteristics of the application. The question of the class membership of a site is mostly decided in the area around the homepage. Since the nature of the information specifying the business is rather general, most designers are placing the purpose of a website near to the homepage. Hence, the area relevant for site classification is not characterized by long paths and the use of considering them is questionable. To conclude, the most effective classification method is based on the representation of a website as a multi-instance object.

The second set of experiments demonstrates the effects of the pruning method when applied to the most promising approach in this scenario, the 0-order Markov tree. Figure 5.8 shows the percentage of the downloaded webpages for varying values of ω . Additionally, the corresponding classification accuracy for each ω is depicted. Note that for values of $5 \leq \omega \leq 15$ the achieved accuracy exceeds 86%, which is the accuracy when reading all webpages. The accuracy for these values is around 86.4%. For $\omega = 8$, it even reaches 87.0%. Thus, pruning a website tree has the advantage of improving the accuracy of the 0-order Markov tree classifier. The efficiency of

the method can obviously be improved too. When reading only 30% of the pages ($\omega = 5$), the classifier already provides the accuracy observed on the complete data or even exceeds it. Thus, reading more pages is not necessary. Even when choosing $\omega = 4$, i.e. reading only 17% of the pages, this classification method still outperforms the second best of the introduced approaches (84.1% against 82.4% for C4.5). Note that reading only 17% of the pages implies a speed-up factor of more than 5, since loading the webpages is the major cost of website classification. Determining a reasonable choice for ω is not very difficult. Since the accuracy did not react very sensitive to varying values for ω after a reasonable value (about 4) was reached, it is relatively easy to make a choice that favors accuracy and/or efficiency. Therefore, the 0-order Markov tree classifier employing the introduced pruning rule is able to offer superior classification accuracy, only using a minor part of the I/O operations for reading complete websites. Let us note that we apply the introduced pruning method on centroid set classifier as well in the next paragraph, but will not discuss the influence of ω again.

Evaluation without explicit Page Classes

To provide classes and corresponding training sites for the second scenario, we employed the Yahoo [Yah] hierarchy. In our testbed, we chose 6 different website classes and built an additional "other"-class from a randomly chosen mixture of other Yahoo [Yah] classes. Our training database consisted of 86 websites for the category "other" and between 12 and 47 example sites for the 6 classes. The total number of sites was 234, comprising a total of about 18,000 single webpages. In this testbed, no page classes were provided to label single webpages within a website.

The first set of experiments tested precision and recall for each of the 6 website classes only for the two-class case. The comparison partners included a 0-order Markov tree classifier and a basic 5-NN classifier using SMD (5-NN for short). Furthermore, we tested an incremental centroid set classifier.

	0-ord-Mark. T.		5-NN		Cent.S.Cl.	
class	prec.	rec.	prec.	rec.	prec.	rec.
busin.sch.	0.74	0.98	0.75	0.89	0.87	0.96
horse deal.	0.80	0.86	0.95	0.78	0.95	0.78
game retail.	0.75	0.75	0.92	0.60	0.77	0.85
ghosts	0.50	0.92	0.60	0.75	0.90	0.75
astron.	0.63	0.92	0.79	0.88	0.88	0.88
snowboard	0.61	0.75	0.86	0.60	0.93	0.70
Acc. 7-Cl.	0.65		0.76		0.81	

Table 5.2: Comparison of precision and recall. Last line: Accuracy for the 7-class problem.

Without having appropriate page classes and training pages, we used the site classes also as page classes for the 0-order Markov tree. The topics of the single webpages were determined by another naive Bayes classifier. Note that only the 0-order Markov tree and the centroid set classifier employed incremental classification, using only a reduced portion of the website as shown in the last section. A second set of experiments investigated the ability of the above three classification methods to handle more than one class by giving the complete training set to the classifier as a 7-class problem. Both experiments used 10-fold cross-validation. The results displayed in Table 5.2 document the ability of the basic 5-NN classifier to provide good precision and recall without using page classes. The 0-order Markov tree classifier using the provisional page classes still shows acceptable results, but the 5-NN-classifier achieves a better trade-off between precision and recall in most of the cases. The incremental centroid set classifier provided very good accuracy and outperformed the other two classifiers. Let us note that the results of the incremental centroid set classifier, displayed in table 5.2, do not belong to the parameter setting offering the best accuracy, but to the setting with the best trade-off between classification time and accuracy. The accuracies achieved for the 7-class problem, listed in the last line of Table 5.2, follow the trend observed in the 2-class problem and underline the capability of the centroid

website class	5-NN	Cent.S.Cl.	0-ord.-Mark.T.
business school	39.16	0.37	0.12
horse dealer	22.40	0.28	0.02
game retailer	22.27	0.34	0.09
ghosts	28.67	0.38	0.03
astronomy	36.99	0.42	0.24
snowboarding	31.59	0.36	0.04

Table 5.3: Classification time in seconds per website for the two class problems.

set classifier to handle larger classification problems. The accuracy is used to measure this experiment because it the most common quality measure for classification problems distinguishing more than two classes.

In Table 5.3, we display the average time spent on the classification of one website for the 2-class problems. The results clearly show that the basic 5-NN classifier takes a considerable amount of time for classification. On the other hand, the incremental centroid set classifier performed pretty well compared to the extremely fast 0-order Markov tree and offered a speed-up of about 100 compared to the basic 5-NN approach. This enormous speed up is due to the small average number of centroids (about 180 per centroid set) and the use of incremental classification considering only few pages of a website (about 20) for very accurate classification. Summarizing the centroid set classifier offered a remarkable classification accuracy in an efficient time.

A third experiment investigated the effects of the parameter setting of GDBSCAN [SEKX98] which is the clustering algorithm used to derive the centroid sets. For the astronomy example, Figure 5.9 shows the dependency of accuracy and classification time on the number k of neighbors needed to define a core point and the radius ϵ. The shape of the graph indicates that the influence of the radius is very stable within the interval from 0 to 0.5 which is half of the possible target interval of the cosine coefficient. On the other

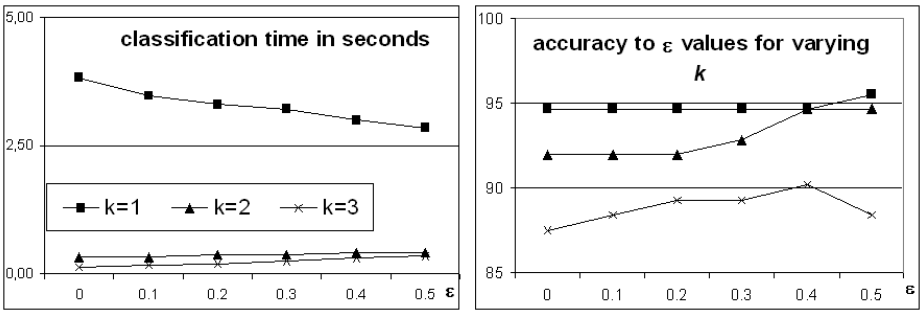


Figure 5.9: Accuracy and classification time depending on the parameter setting for GBDBSCAN for the Astronomy example.

hand, the influence of the number of neighbors k shows an obvious decrease of accuracy for $k = 3$ and no significant efficiency gain for $k < 2$. Therefore, setting $k = 2$ and $\epsilon = 0.4$ offered a good trade-off between classification time and accuracy.

Conclusions on Website Classification

To conclude, the simple methods of website classification like the homepage and the superpage approach were not suitable to achieve reliable website classification. For the scenario that page classes and corresponding training pages can be provided, the 0-order Markov tree performed best. Since this approach does not employ the link structure like other Markov trees, it treats websites as multi-instance objects, i.e. sets of feature vectors. For the scenario without page classes, the centroid set classifier outperformed all other classifiers and demonstrated classification times that are suitable for real-world applications. Last but not least, the introduced pruning method is capable to reduce the classification time and to increase the accuracy.

5.4 Focused Crawling for Relevant Websites

5.4.1 Motivation

Focused web crawlers have recently emerged as an alternative to the established web search engines like Google [Goo]. A focused web crawler [CvdBD99b] takes a set of well selected webpages, exemplifying the user interest. Searching for further relevant webpages, the focused crawler starts from a set of given pages and recursively explores the linked webpages. While the crawlers used for refreshing the indices of web search engines perform a breadth-first search of the whole web, a focused crawler explores only a small portion of the web using a best-first search guided by the user interest. Compared to web search engines, focused crawlers obtain a much higher precision and return new pages which are not indexed yet. Recently, focused web crawlers have received a lot of attention in the research areas of database systems, information retrieval and data mining [CvdBD99a, CvdBD99b, CPS02, Cha03, CGMP98, RM99].

As mentioned before, using a web directory for the search of relevant websites has several drawbacks. Web directories offer in most cases only a very small portion of the websites that are relevant to a given topic. The given categorization might totally lack the topic a user is interested in. Last but not least, web directories might not be up-to-date due to manual maintenance. In this section, we therefore extend focused crawling to the search for relevant websites, offering a method to significantly increase the recall of existing web directories. Additionally, such a crawler can act as an alternative approach for searching the web for topics not listed yet in any web directory.

To adopt focused crawling for website retrieval, the simplest way is to use one of the well-established methods for focused webpage crawling and, in a step of post-processing, analyse the resulting webpages in order to find relevant sites. This analysis can be done by looking for relevant homepages or by applying a website classifier to all pages retrieved from a given web-

site. However, this approach is severely limited by the fact that there is no guarantee that the crawled webpages are representatives of their corresponding websites. We argue that in order to achieve efficient and accurate website crawling, the concept of websites has to be integrated into the focused crawler. Therefore, we introduce a novel focused crawler that directly searches for relevant websites instead of single pages. The proposed focused website crawler is based on the website graph, introduced in section 5.2. The website graph is a two-level graph abstraction of the WWW, representing both webpages and websites together with their links. The crawling task is divided into two major subtasks corresponding, to the two different levels of abstraction:

- An internal crawler views the webpages of a single given website and performs focused (page) crawling within that website.
- The external crawler has a more abstract view of the web as a graph of linked websites. Its task is to select the websites to be examined next and to invoke internal crawlers on the selected sites.

The proposed two-level architecture allows the crawler to control the number of pages to be downloaded from each website and enables it to find a good trade-off between accurate classification and efficient crawling. Our experimental evaluation demonstrates again that website classification based on the homepages is considerably less accurate than classification methods employing more than on webpage. Furthermore, we compare our prototype of a focused website crawler to a focused webpage crawler with website post-processing and show that the introduced methods of focused website crawling clearly increase the efficiency as well as the accuracy of retrieving relevant websites from the WWW. The solutions in this section were published in [EKS04]. The outline of the section is as follows. After this introduction, we briefly survey related work on focused crawling. Afterwards, we define the task of focused website crawling and a basic solution. Section 5.4.5 presents

our novel approach to focused website crawling and section 5.4.6 reports the results of our experimental evaluation.

5.4.2 Related Work on Focused Crawling

One of the first focused web crawlers was presented by [CGMP98] which introduced a best-first search strategy based on simple criteria such as keyword occurrences and anchor texts. Later, several papers such as [CvdBD99a] and [CvdBD99b] suggested to exploit measures for the importance of a webpage (such as authority and hub ranks) based on the link structure of the WWW to order the crawl frontier. These measures, which are very successfully used to rank result lists of web search engines, also proved to be very effective in focusing a crawler on the topic of interest of a user. Recently, more sophisticated focused crawlers such as [CPS02], [DCL+00] and [RM99] incorporate more knowledge gained during the process of focused crawling. [DCL+00] introduced the concept of context graphs to represent typical paths leading to relevant webpages. These context graphs are used to predict the link distance to a relevant page and, consequently, are applied to order the crawl frontier. [RM99] explored a reinforcement learning approach, considering the successful paths observed, to weight the links at the crawl frontier based on the expected number of relevant and reachable webpages. [CPS02] extends the architecture of a focused crawler by a so-called apprentice which learns from the crawler's successes and failures and is later consulted by the crawler to improve the ratio of relevant pages that are visited. Like a human user, the apprentice analyses the HTML structure of a webpage to judge the relevance of the outlinks of this page. To the best of our knowledge, all focused crawlers presented in the literature search for individual webpages and not for whole websites. The only site-oriented features of established page crawlers are the measures to prevent so-called spider traps and the prevention of host-to-host reinforcement proposed by Bharat and Henzinger [BH98]. A spider trap is an infinite loop within a website that dynamically produces new pages trap-

ping a web crawler within this website. Therefore, the common approach to prevent spider traps limits the maximum number of pages to be downloaded from a given website in order to escape the trapping situation [Cha03]. However, these crawlers do not have any means to control the search within a website.

5.4.3 Retrieving Websites with Focused Crawling

A focused webpage crawler [CvdBD99b] takes a set of well selected webpages, exemplifying the user interest. Searching for further relevant webpages, the focused crawler starts from the given pages and recursively explores the linked webpages. The conceptual view of the WWW of a focused page crawler is the webpage graph (compare section 5.2). The crawl frontier consists of all hyperlinks (or the referenced webpages) from downloaded pages pointing to not yet visited pages. The performance of the crawler strongly depends on the crawling strategy, i.e. the way the frontier is ordered. There are several ways of post-processing the results of focused page crawlers to adapt them for the task of retrieving relevant websites. The simplest way is to select all homepages of websites found within the relevant pages of a crawl and to conclude that all corresponding websites are relevant. However, the classification of websites based on the homepage alone is not as accurate as more sophisticated methods of website classification. This was demonstrated in the former section.As a consequence, this approach to extract relevant websites from the results of a focused webpage crawl suffers from inaccurate results. Furthermore, since the webpage crawler does not prefer homepages over other webpages, the rate of newly discovered websites tends to be rather low. Another approach of post-processing is to group the resulting webpages by their website, i.e. domain, and apply a more sophisticated website classifier. Though this approach promises better classification accuracy, it still has drawbacks. Since the set of downloaded webpages for each site is controlled by the page crawler which is not conscious of websites at all, this selection

of pages might not be well suited for representing the website. Thus, the crawler does not guarantee that enough webpages per site are downloaded. In our experiments, it turned out that usually more than 50% of the websites that were classified as relevant by this method were represented by only one webpage. On the other hand, the efficiency suffers from the effect that very relevant websites might be scanned completely due to the high relevance scores of most of their pages. In addition to the number of examined page, the selection of webpages of a conventional focused crawler causes a problem as well. Since a focused crawler prefers relevant pages, a website might be represented by the pages closest to the relevant topic. But this selection is not a good representation for websites that are irrelevant. Thus, websites containing some pages with relevant information, belonging to the "other" class are misclassified. For example, a university might be classified as relevant for skiing because there are some student pages referring to this topic. We argue that in order to achieve high classification accuracy and to control the number of pages to be downloaded, a focused website crawler requires an explicit concept of websites and corresponding crawl strategies.

5.4.4 Preliminaries to Focused Website Crawling

Website crawling can be considered as the process of successively transforming a subgraph G_0 of the website graph WG with $V_0 = W1, \ldots, Wn$, $n \geq 1$, where W_i is a website, $1 \leq i \leq n$, into a sequence of subgraphs $G_1, \ldots, G_m$ such that in each step exactly one website node from WG is added to G_i to obtain G_{i+1}. V_0 is called the set of start websites. In the context of focused website crawling, we assume two classes of websites. A class of relevant sites (the target class) and a class of irrelevant sites (the "other"-class) with respect to some user interest. The set of start sites V_0 should (mainly) consist of relevant sites. To distinguish between relevant and irrelevant websites, a website classifier is required which predicts the class of a website (V', E') based on the feature vectors $FT(p)$ of the pages $p \in V'$. In the context of

focused website crawling a website classifier is a function that takes a website from W and a website class from the set of classes C and returns a numerical confidence value for this website w.r.t. the given class.

$$confidence : W \times C \rightarrow [0, \ldots, 1]$$

A website is called relevant if its confidence for the target class c_{target} exceeds its confidence for the "other" class.

$$relevance : W \rightarrow \{true, false\}$$

The website classifier is trained using the start websites that can be provided either explicitly by the user (if available) or implicitly by selecting some subtrees (and the corresponding websites listed in these subtrees) of a directory service like [Goo, DMO, Yah]. Based on website classification and the notion of relevant websites, we introduce the following performance measure for focused website crawlers.

Definition 5.6 (PPRS-Rate)
The pages per relevant site rate (pprs-rate) of the website crawler after step s is defined by the ratio of the number of downloaded webpages to the number of relevant websites found, i.e.

$$pprs(G_s) = \frac{\sum_{W \in (G \setminus G_{s-w})} |pages(W)|}{|\{W \in (G \setminus G_{s-w}) | relevance(W) = true\}|}$$

where $pages(W) = \{p \in W \cap (G_s \setminus G_{s-w})\}$ denotes the set of pages in W that were visited so far and w is the beginning of the time interval that is observed.

The pprs-rate measures the average effort to retrieve one additional relevant website. It depends on two factors: (1) the number of pages that have to be downloaded within a relevant website (to be controlled by an internal crawler) and (2) the number of pages downloaded from irrelevant websites that were examined before finding the relevant website (to be controlled by

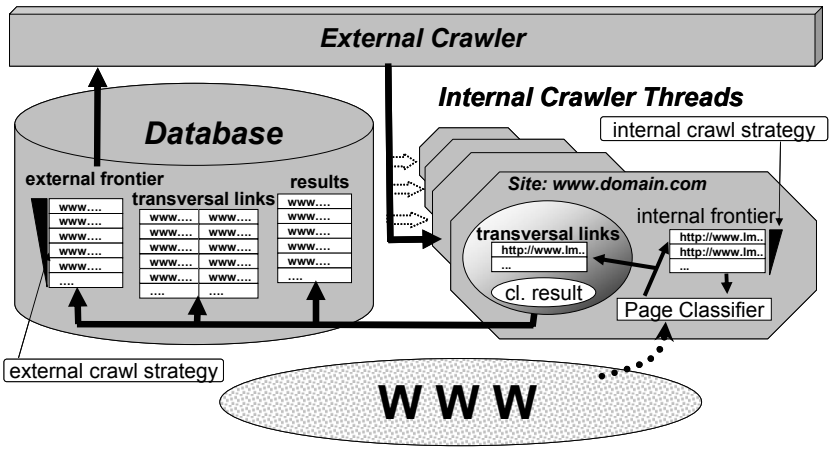


Figure 5.10: Architecture of the focused website crawler.

the external crawler). The task of a focused website crawler is to find as many relevant sites as possible while downloading as few webpages as possible. A website crawl terminates if the wanted number of relevant sites is found or the pprs-rate decreases significantly. In the next section, we will introduce our architecture of a focused website crawler

5.4.5 A Focused Website Crawler

The Architecture

Focused website crawling is performed on two levels. The external or website level traverses the first level of the website graph. The external crawl orders the (hyperlinks to) yet unknown websites and invokes internal crawls on the top-ranked ones. Since there are much less domains than webpages, the external crawl frontier is rather small compared to the crawl frontier of an ordinary focused crawler. Thus, even for large crawls ranking can be done on-the-fly and sophisticated ranking algorithms can be applied. The second level is the internal or webpage level. It examines the current website to identify its purpose and extracts links to other websites while downloading as few webpages as possible. Since the webpages within the internal crawl frontier

are only needed for a limited time and their number is usually small, it can be stored in the main memory. Thus, expensive I/O operations are avoided and the crawl frontier can be accessed and updated very fast. Let us note that several internal crawlers examine different websites simultaneously. Thus, it is guaranteed that the data is drawn from several remote hosts at the same time which ensures a high overall download rate. Furthermore, controlling the number of visited pages from each website helps to keep the additional load at each website as low as possible, helping to increase the acceptance of the focused crawler within the webmaster community. Figure 5.10 shows our architecture for a focused website crawler. The external crawler stores the external frontier consisting of websites only. To decide which website has to be examined next, it ranks the external frontier. To expand the frontier and to decide if a chosen site is relevant, the external crawler invokes an internal crawler. The internal crawler traverses the website, building an internal crawl frontier that is restricted to the pages of this site. During this traversal, it examines the webpages to determine the site class. Furthermore, it collects all transversal links to other unexplored websites together with the confidence w.r.t. the target class of their source pages. As a result, the internal crawler returns information about the website class and the set of transversal links from the domain to new unexplored domains. Note that these transversal links are not real hyperlinks, but an aggregation of all hyperlinks that are found within the website directing to pages located within another website. Thus, the number of transversal links from one site to another website is limited to one.

The External Crawler

The task of the external crawler is to order the external crawl frontier (consisting of links to not yet visited websites) and to decide which site has to be examined next by an internal crawler. The external crawler starts its traversal of the website level from the user-specified start websites and

expands the graph by incorporating the newly found websites. Since the task of the external crawler is similar to the task of a focused crawler for webpages, most of the methods mentioned in section 5.4.2 are applicable to order the external frontier. The major difference is that distillation takes place at another more abstract level. Thus, the relevance scores attached to nodes and edges may be determined in a different way in order to achieve good results. During a crawl, we distinguish two different sets of nodes of the website graph: Nodes corresponding to already examined websites are elements of V_{ex} and so-called border nodes that have not yet been examined are elements of V_{bd}. The task of the crawler is to rank the elements of V_{bd} with respect to the information, gained while examining the elements of V_{ex}. Each website $W \in V_{bd}$ is reachable by at least one link contained in some website $V_i \in V_{ex}$. The (external) crawling strategy employed in this paper is simple but effective and is very similar to the basic crawler proposed in [CPS02]. Note that most of the established crawling strategies [CvdBD99a, CvdBD99b, CPS02, Cha03, CGMP98, RM99] are applicable as well. For every node $W \in V_{bd}$, a ranking score is calculated as follows:

$$rank(W) = \frac{\sum_{V_i \in L_{ex}(W)} weight(V_i, W)}{|L_{ex}(W)|}$$

where $L_{ex}(W) = \{V | V \in V_{ex} \wedge \exists edge(V, W)\}$ and $edge(V_i, W)$ denotes that there is at least one link from node V_i to node W. Furthermore, $weight(V, W)$ is a function that determines the confidence for each edge that its destination is relevant to the topic. In other words, an unknown website is judged by the average weight of the known edges referencing it. Thus, the website W with the highest $rank(W)$ should be crawled first. The edges do not directly correspond to the hyperlinks, but represent an aggregate of all hyperlinks, leading from one website to another. Let us note that this method solves the same problem as the host-to-host cleaning improvement suggested in [BH98], i.e. it avoids that strongly connected domains are overemphasized. The remaining task is how to determine the weights for the edges. To answer this question, we investigated the following three approaches :

- Global edge weights: Each edge is weighted by the confidence w.r.t. the target class of the website the edge is contained in:

$$weight_{global}(W, V) = confidence(W, c_{target})$$

where $confidence(W, c_{target})$ denotes the confidence value for website W w.r.t. the target class c_{target}.

- Local edge weights: Each edge is weighted by the average confidence w.r.t. the target class of the webpages containing links pointing to the given website:

$$weight_{local}(W, V) = \frac{\sum_{p \in \{p \in W | \exists (q \in V \wedge (p,q))\}} Pr[target|p]}{|\{p \in W | \exists (q \in V \wedge (p, q))\}|}$$

where $Pr[target|p]$ is the confidence of page p being contained in a target class website. These confidences for single webpages are also collected within the website classifier, but do not correspond to the complete set of webpages downloaded for classification.

- Combined edge weights: This is a combination of both methods integrating both scores to combine local and global aspects by taking the average weight of both methods:

$$weight_{combined}(W, V) = \frac{weight_{local}(W, V) + weight_{global}(W, V)}{2}$$

The advantage of local edge weights is that they distinguish the transversal links according to the relevance of the source pages of a link. Thus, transversal links found on irrelevant pages are weighted less than those found on highly relevant webpages. On the other hand, local edge weights might consider the links from source pages containing only sparse text as irrelevant since the page itself can be classified only poorly. This shows the strength of global edge weights. Since global edge weights consider the relevance of the complete site, they transfer relevance from other relevant pages to the link pages which do not provide enough content for proper classification.

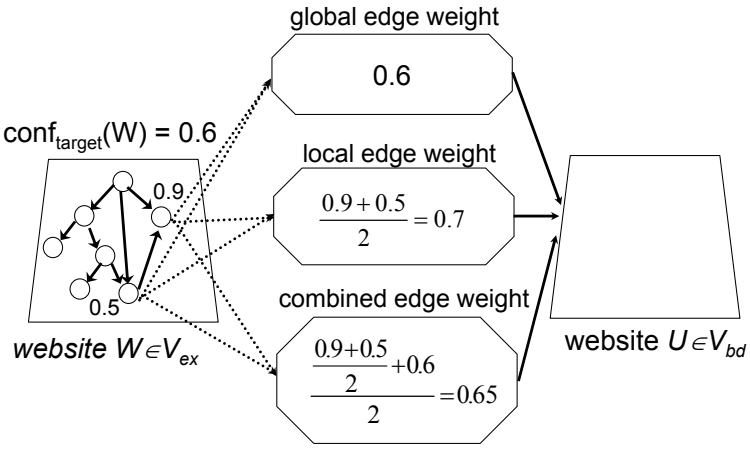


Figure 5.11: The three variants of edge weights for two sample websites W and U.

Combined edge weights incorporate both aspects. The links found in pages containing not enough text for reliable classification are at least judged by the relevance of the website and relevant pages transfer more importance to the links than irrelevant ones. Figure 5.11 displays an example for all three methods of edge weighting. The confidence of W w.r.t. the target class is 0.6. There are two pages in W referencing pages in U, one page with confidence (w.r.t. the target class) 0.9 and the other with confidence 0.5.

The performance of the external crawler influences one important aspect of the pprs-rate: the number of relevant sites that are examined compared to all websites that are crawled by an internal crawler. We will refer to this ratio as the *website harvest rate*. However, this aspect is not the only influence on the pprs-rate. Even an optimal external crawler will achieve very bad pprs-rates, if the internal crawler explores large numbers of webpages per site.

The Internal Crawler

The internal crawler is responsible for the main advantage of a dedicated website crawler namely that the results are more reliable due to better clas-

sification accuracy. On the other hand, the efficiency strongly depends on the ability of the internal crawler to restrict the number of downloaded webpages per site to as few pages as possible. Furthermore, additional aims have to be achieved like the avoidance of spider traps and the retrieval of new promising transversal links. The main task of the internal crawler is to select a representative sample set of webpages from a website W and determine for each page p_i the likelihood (called confidence in this context) of p_i appearing in website class c_k. To determine this probability $Pr[w_i|w_i \in W \wedge W \in c_k]$, we employ a text classifier. To choose the sample set, we employ focused crawling using a so-called internal crawl strategy. To determine the class of an entire website W, we calculate the probability that W was generated by the process corresponding to class c_k for each class c_k. Additionally, there are several other side goals of the internal crawler like collecting new transversal links and avoiding spider traps.

The Webpage Classifier

The task of the webpage classifier is to decide how likely it is that a certain webpage p_i appears in a website W of Class c_k. The task of this classifier is slightly different from the task of the classifier in an ordinary focused crawler. A webpage that is likely to appear in a typical website does not necessarily have to be relevant for the user interest. The page classifier should be capable to handle multi-modal classes, i.e. classes that are strongly fractioned into an unknown number of subclasses. This feature is important because the webpages found in websites of a common class provide several page classes, e.g. contact-pages, directory pages, etc.. For our crawler, we employed a centroid based kNN classifier as described in [HK00]. This variant of kNN classification constructs the centroid of the training word vectors for each class. The class is now determined by choosing the class that belongs to the closest centroid. In order to achieve multi-modality, we adopted an idea mentioned in the summary of [HK00]. We clustered each training set, using the k-means algorithm and represented a class as the set of centroids of the resulting clusters. Let us note that we started our prototype by using naive

Bayes classification, but changed to this classifier due to better accuracy. Formally, each class c_k of our classifier is represented by a set of centroids CS_k. Let $d_{min}(p, CS_k)$ denote the distance of the word vector p of a given webpage to the closest element of CS_k. Then, we estimate the confidence value for p belonging to c_k as follows:

$$Pr[p|c_k] = \frac{\ln\left(d_{min}(p, c_k)\right)}{\sum_{c_j \in CS_k} \ln\left(d_{min}(p, c_j)\right)}$$

In other words, we use the logarithm of the distance to the closest centroid in CS_k and normalize over all classes. Let us note that we use the logarithm to weight close distances higher than far distances. Therefore, if a page has a large distance to the centroids of all classes, the confidence values are very similar for all classes. The closer the distance to a centroid is the more sensible the distance is measured. The resulting confidences are used by the local and combined edge weights for determining the weights of the transversal links. To train the classifier, we first select a set of relevant websites. The websites in our experiments, for example, were taken from common directory services [Yah, Goo, DMO]. To represent the "other"-class, we chose several websites belonging to a variety of other non-relevant topics. Since we need to learn which types of webpages might occur in a relevant site and which not, we have to draw a representative sample of webpages from each training website. The pages downloaded during the process of classification of a website are limited to a small set around the homepage, since these pages are most likely connected to the purpose of the site. Thus, we should use these pages for training as well. We restrict the training pages to the first k pages when traversing the website using breadth-first search. This simple method worked out well in our experiments.

The Internal Crawl Strategy

The internal crawl strategy determines the sample of pages downloaded from the website to be classified. Each internal crawl is started at the homepage. As mentioned before, the information about the purpose of a website is usually located around the homepage since most publishers want to tell the user

what a website is about, before providing more specific information. Analogously to a focused page crawler, the internal crawler traverses the web using a best-first search strategy. However, the internal crawl is restricted to the webpages of the examined site. The goal is to find a set of webpages reflecting the site's purpose in a best possible way. This is a major difference to focused page crawlers which try to find as many relevant pages as possible. However, looking for relevant pages is only appropriate for site classification if the examined website belongs to the target class. If the given website belongs to the other class, the crawler should prefer pages that typically occur in non-relevant websites in order to find a good representation. Thus, the internal crawler should rank the pages by their confidences for any class compared to the average confidence over all classes. To solve this problem, our internal crawling strategy works as follows. Like in the external crawler, we again use a crawling strategy similar to the basic crawler in [CPS02]. The ranking score of a webpage p is defined as the average weight of the links referencing p:

$$rank(p) = \frac{\sum_{q_i \in L_{in}(p)} weight((q_i, p)}{|L_{in}(p)|}$$

where L_{in} is the set of pages read so far that link to p. To represent the contribution of a page for the decision in favor of either class, we determine the weight of link (q_i, p) as:

$$weight(q_i, p) = variance_{c_j \in C}(Pr[q_i|c_j])$$

where $Pr[q_i|c_j]$ is the confidence of q_i w.r.t. to class c_j obtained by the page classifier. The internal frontier is sorted in decreasing order of these confidence values.

The Website Classifier

The combination of the page classifier and the internal crawl strategy produces a sequence of webpages downloaded from the site. Furthermore, each webpage is classified and is associated with a confidence w.r.t. the target class. The following statistical model incrementally, i.e. after each down-

load of a new page aggregates these page confidences to calculate an overall confidence w.r.t. the target class for the entire website. In our model, each website class defines a statistical process that can generate any webpage with a certain probability. A website W belonging to class c_k is a set of webpages generated by drawing pages from the corresponding probability distribution. In the following, we present a maximum-likelihood classifier that assigns a website to the class with the highest probability of having generated the observed website W. Let W_t denote the sample of site W that the internal crawler has retrieved by time t. The probability that the class c_k has generated W_t is given by:

$$Pr[W_t|c_k] = \prod_{p_i \in W} Pr[p_i|c_k].$$

Applying the Bayes theorem, the desired probability is:

$$Pr[c_k|W_t] = \frac{Pr[c_k] \cdot Pr[W_t|c_k]}{\sum_{c_i \in C} Pr[c_i] \cdot Pr[W_t|c_i]}$$

Unfortunately, this formalization suffers from two practical limitations:

- The a priori probabilities $Pr[c_k]$ are unknown for the WWW. However, the application of focused crawling enables us to make a suitable estimation. Since the focused website crawler focuses relevant sites, the probability distribution within the whole web is expected to be very different from the probability distribution within relevant sites close to the frontier. Thus, we can use the rate of relevant sites found so far as an estimate for $Pr[c_k]$.

- Since there is no classifier guaranteeing 100 % accurate class predictions, the confidence values are not always realistic as well. The class prediction values generated by the classifier always suffer from a certain classification error. Thus, the combination of these results should consider this inaccuracy.

To incorporate the possibility of classification errors, we extend our model by integrating the classification error observed on the training data into the

model. Thus, we obtain an error corrected probability for the occurrence of page p in a website of class c_k and the classification error p_{err}:

$$Pr[p|c_k \wedge p_err] = Pr[p|c_k] \cdot (1 - p_{err}) + Pr[p|c_other] \cdot p_{err}$$

The idea is that the probability for a correct prediction is calculated by multiplying the confidence value with the probability that the webpage classifier made no mistake. Additionally, we have to consider the case that the classifier made a wrong prediction. Thus, we have to add the confidence value of the "other"-class multiplied with the error probability p_{err}. To estimate p_{err}, we calculate the accuracy of the page classifier on the set of webpages in the training websites using 10-fold cross-validation. Using the error corrected probabilities avoids the effect that the influence of a single page is overestimated during classification. Even if the classifier outputs are 1.0 and 0.0, our process does not automatically overestimate the impact of a single page. Thus, the calculated value for $Pr[W_t|c_k]$ will usually produce meaningful values after some pages have been considered.

To stop classification, we define a certain confidence threshold $p_{threshold}$ and the internal crawl stops classification as soon as this confidence level is reached. By choosing $p_{threshold}$, the internal classifier can be adjusted to find an appropriate trade-off between accuracy and efficiency. However, if its value is chosen too high, the crawler will require too many pages with respect to a website's purpose. This is a problem, because the performance suffers significantly and the reservoir of characteristic pages within one website is limited. To conclude, after the confidence for W_t reaches $p_{threshold}$, we assume that the class of W is identical to the class of W_t and we denote:

$$confidence(W, c_{target}) = Pr[c_{target}|W_t]$$

and

$$relevance(W) = (Pr[c_{target}|W_t] > Pr[c_other|W_t])$$

Figure 5.12 illustrates the complete process of website classification. The displayed example describes the common case that a website starts with a

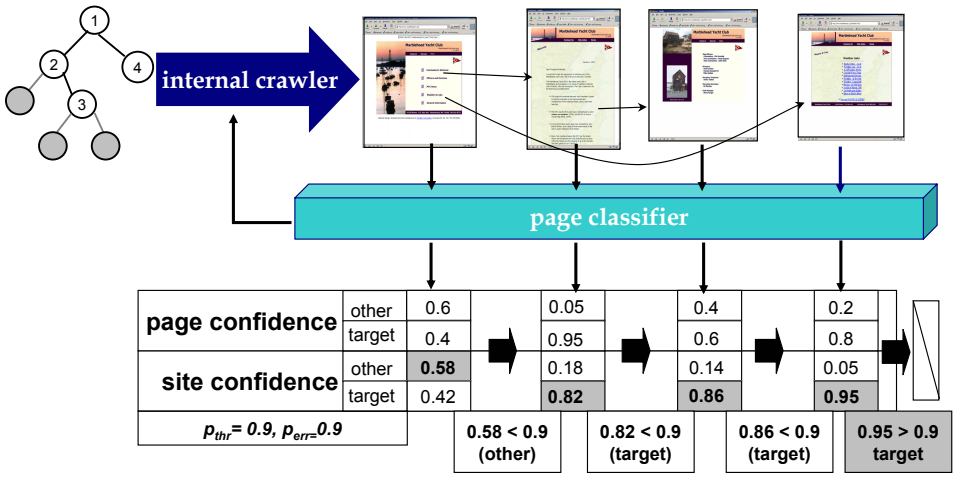


Figure 5.12: Illustration of website classification during an internal crawl.

frame page and, thus, the prediction of the class based only on the homepage would be wrong.

Retrieving Transversal Links and Terminating the Internal Crawler

Besides the primary goal to achieve accurate classification of the examined website, the internal crawler has another purpose of retrieving enough transversal links for extending the external crawl frontier. Therefore, the internal crawler collects all transversal links, i.e. the links leading to new unexplored websites. Additionally, the crawler stores the confidence values $Pr[p|c_{target}]$ of the source pages of the link. These values are used to calculate local and combined edge weights. Since, according to the above stop condition, classification might be finished after a few pages, it is possible that the internal crawler has not yet found enough interesting transversal links. In such cases, we want to continue the crawl until a reasonable number of transversal links has been extracted. To decide if enough links have been found within a website, we define the *linkWeight* as a measure for the contribution of page p to the set of relevant transversal links found within the site:

$$linkWeight(p) = (Pr[p|c_{target}] \cdot |LT_p| + c)$$

where LT_p is the set of transversal links found in p and $c \geq 1$ is a constant. Furthermore, we define the $LinkRank$ for the set of webpages W_t as:

$$LinkRank(W_t) = \sum_{p \in W_t} linkWeight(p) \cdot \frac{1}{Pr[c_target|W_t]}$$

To employ the $LinkRank$ for ensuring that enough relevant links are found, we continue the internal crawl even after classification has finished until it reaches a certain level $l_{threshold}$. The idea of this heuristic is that each webpage contributes its $linkWeight$ to the $LinkRank$ of the website. The more links are contained in p and the more relevant p is, the more will p contribute to the $LinkRank$. The constant c is added to ensure that the $linkWeight$ has at least some value and thus the $LinkRank$ grows constantly until $l_{threshold}$ is reached. The $LinkRank$ increases slower for relevant websites and faster for irrelevant ones. Thus, an internal crawl of a relevant website will encompass more webpages than an internal crawl of an irrelevant site which usually terminates after classification. This way relevant websites add more new links to the external frontier than irrelevant ones. Let us note that we continue the crawl to reach $l_{threshold}$ by employing the mentioned internal crawling strategy. We argue that if a website is relevant, the crawling strategy is targeted to find new relevant pages which are most likely to contain relevant links. For websites classified to the other class, $l_{threshold}$ is reached rather fast and switching the crawl strategy is not necessary. An additional benefit of the internal crawler is that it makes the website crawler robust against spider traps. Since the number of webpages retrieved from one website is explicitly controlled, the crawler might run into a spider trap only in those rare cases where a site consists mostly of pages without any meaning to the classifier. To ensure termination in such cases, it is sufficient to restrict the number of pages downloaded from one domain. Unlike in page crawlers, no additional database table is needed to store websites containing a spider trap. This is

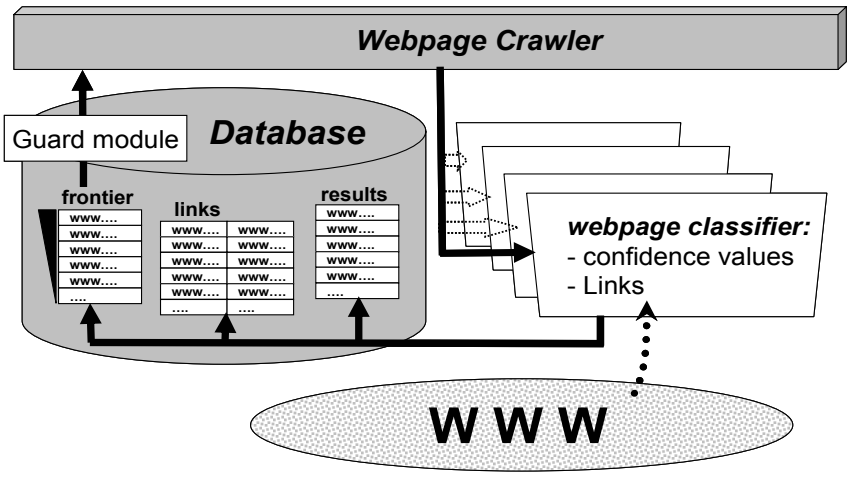


Figure 5.13: Architecture of our focused webpage crawler.

not necessary within the focused website crawler, since the crawler will not visit a website more than once.

5.4.6 Experimental Evaluation for Focused Website Crawling

The Test Environment

We performed our experiments for the topics listed in Table 5.4. For each topic, we first acquired a sample set of relevant websites taken from a category in a web directory. Additionally, we selected a random mixture of websites to represent all other topics on the web. For each category, Table 5.4 provides the number of training websites, and the web directory service the websites were taken from. We stored the websites in a training database to have a stable test environment consisting of 20,793 HTML-documents from 335 websites and we implemented 2 focused crawlers. The first is our prototype of a focused website crawler. The second is a focused webpage crawler that crawls the internet by using only one frontier of webpages. To provide a fair comparison, both crawlers are based on the same algorithm for page classifi-

cation and ranking. The design of our focused webpage crawler is illustrated in Figure 5.13. The system starts its crawl on a defined set of webpages, in our case the homepages of the websites found in a directory service. Each new unexplored webpage is stored in the crawling frontier. The page classifier generates confidence values for each webpage that is explored. Within the frontier, each unexplored webpage is measured by the average confidence value for the target class of the webpages linking it. The webpage providing the highest average confidence value within the frontier is examined next. In order to prevent spider traps and to keep the load for each website at an acceptable level, we implemented a guard module as described in [Cha03]. This guard module prevents the page crawler from accessing webpages in websites that already contributed an extraordinarily high number of webpages to the already explored part of the web graph. To test the crawlers, we performed various crawls on the WWW. This testbed seemed to be suited best, although it is not guaranteed that the web stays the same between two crawls. However, due to the more stable character of the website graph, we argue that the influence to the results is negligible. Let us note that we performed some of the experiments again after several weeks and achieved almost identical results. On the other hand, downloading a representative section of the website graph to provide a static test environment is difficult. Since the part of the WWW visited by a website crawler tends to be spread over several thousands hosts, it is difficult to find a closed section that allows a realistic behavior of the tested crawlers. Our experiments were run on a workstation that is equipped with two 2.8 GHZ Xeon processors and 4 GB main memory. As a database system we used an ORACLE 9i database server hosted on the same machine. Both crawlers were implemented in Java 1.4 with the exception of the ranking algorithms and the guard module which were partly implemented in PL/SQL to improve the runtime performance.

topic	number of websites	websites provided by
horses	32	YAHOO
astronomy	39	YAHOO
sailing	39	Google
mountain biking	34	DMOZ
skate boarding	35	DMOZ
boxing	33	DMOZ
other	132	all

Table 5.4: Overview over the training database.

Accuracy of the Website Crawler

Our first experiment demonstrates the higher accuracy that can be achieved for website classification by using the internal crawler compared to a homepage classifier. The homepage classifier uses the same centroid based kNN classifier as the internal crawler, but is trained and tested on homepages only. The internal crawler used in these experiments terminates its crawl after a confidence threshold of $p_{threshold}$ is reached and does not continue the crawl to find interesting links. Since this test needs labelled test data, we performed 10-fold cross-validation on the topics stored in the training database (Table 5.4). Table 5.5 displays the precision, recall and f-measure (as trade-off between precision and recall) for the tested topics when employing the website classifier and the homepage classifier. Additionally, the table reports the classification error p_{err} and the average number of webpages that the website classifier downloaded per website. For the training of the page classifier of the internal crawler, we used the first 25 webpages of each training website when applying a breadth-first traversal. For all of the tested topics, the internal crawler obtained significantly higher f-measures than the homepage classifier. For the topic horses, it even increased the f-measure from 0.63 to 0.9, i.e. by 0.27. Thus, by classifying the websites by more than one page, the classification accuracy was substantially increased. Let us note that a

topic	p_{err}	$p_{thres.}$	pages	internal crawler.			homepage classifier		
			per site	prec.	rec.	f-mea.	prec.	rec.	f-mea.
horses	0.85	0.9	6.9	0.84	0.97	0.90	0.49	0.88	0.63
astronomy	0.90	0.9	6.3	1.00	0.90	0.95	0.86	0.79	0.83
sailing	0.88	0.9	6.3	0.90	0.97	0.94	0.77	0.92	0.84
mountain biking	0.86	0.8	6.3	0.81	0.97	0.88	0.76	0.83	0.79
skate boarding	0.88	0.8	3.2	0.76	1.00	0.86	0.74	0.89	0.81
boxing	0.88	0.9	7.4	0.79	0.79	0.79	0.74	0.72	0.73

Table 5.5: Classification results using 10-fold cross validation within the training database for the internal crawler and the homepage classifier.

manual analysis of the crawled websites confirmed the hypothesis that especially commercial websites often do not provide a meaningful homepage. The average number of pages used for classification was between 3.2 and 7.4, indicating that website classification does not require large numbers of webpages per site for making more accurate predictions.

Evaluation of the Crawling Performance

To demonstrate the performance of the complete focused website crawler, we performed numerous crawls. Since we retrieved a total number of approximately 50,000 potentially relevant websites, we could manually verify only samples from each crawl. Table 5.4.6 displays a sample of relevant websites retrieved for the topic horses. The first five domains were retrieved after approximately 250 websites were visited, the last five at the end of the crawl after about 2,500 relevant websites were retrieved. This example illustrates that the crawler started to discover relevant websites early and kept his good accuracy until the end of the crawl. Our first crawling experiment compares the three different weightings introduced in section 4.2 for ranking the external frontier. Therefore, we started each crawler using the parameters achieving maximum accuracy for the internal crawler and stopped the crawler after approximately 2,500 relevant websites were found. To compare the effect of each of the weightings, we compared the website harvest rate, i.e. the ratio of relevant websites to all websites that were screened. Figure

website	visited pages	confidence
www.tbart.net	4	0.65
www.socalequine.com	6	0.59
www.thehalterhorse.com	4	0.65
www.thejudgeschoice.com	4	0.75
www.thehorsesource.com	3	0.68
...		
www.laceysarabians.com	5	0.71
www.baroquehorses.com	5	0.60
www.knightmagicfarms.com	4	0.67
www.pccha.com	7	0.64
www.danddhorsetransport.com	4	0.70

Table 5.6: Example websites returned for the topic horses.

5.14 displays the average website harvest rate aggregated over the last 1,000 pages. For the topics horses and astronomy, all three weightings performed very similar, although the global edge weights achieved a small advantage, especially at the beginning of the crawl. However, for the topic sailing, the combined edge weights were able to compensate some of the weaknesses of both underlying methods. The experiments for the topic mountain biking displayed a strong advantage for the local edge weights. However, the combined edge weights were still able to compensate some of the weaknesses of the global edge weights. Though our experiments did not reveal that one of the mentioned weightings showed superior results, we advise to employ the combined edge weights function, since it was always at least the second best and sometimes outperformed the other methods.

The next series of experiments was conducted to back up our claim that common focused (webpage) crawlers are unsuitable for retrieving websites and that the proposed focused website crawler overcomes the problems of page crawlers, providing a more efficient and accurate retrieval of relevant

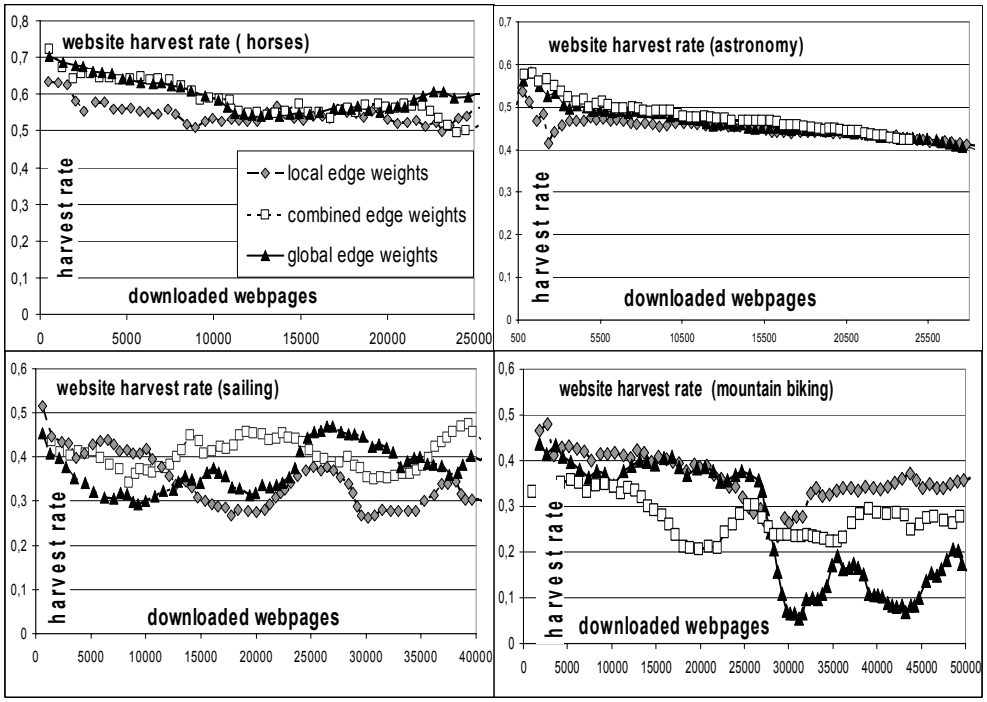


Figure 5.14: Website harvest rates (average of the last 1,000 pages) for each topic each weighting.

websites. In our first experiment, we have already demonstrated that the accuracy of the internal crawler is superior to the accuracy achieved by the homepage classifier. Thus, the post-processing counting relevant homepages is unlikely to produce the same quality of results either. To show that applying a website classifier is not sufficient for providing comparable accuracy, we determined the percentage of websites that were classified by one single webpage. For all four examples approximately 50% of the resulting websites where classified by using only one page. Thus, in half of the cases applying a more sophisticated website classifier to the websites being aggregated from the results of a page crawl cannot perform any better than the homepage classifier. This behavior of the page crawler can be explained as follows. Most transversal links referencing a new site are directed at one special en-

try page (usually the homepage) and most other webpages found within this website are linked only via internal links. A page crawler examining a website visits this entry page first and classifies it. The ranking score of the other webpages within the website now strongly depend on the confidence value of the entry page. If the confidence w.r.t. the target class is rather high, then additional pages are examined also. If the classification result is rather uncertain, however, the ranking scores tends to be rather low and it is likely that the additional pages will not be visited during the crawl. For the task of website retrieval this behavior is unsuitable. If the relevance of the entry page is hard to decide, it would make sense to examine additional pages from the site in order to achieve more reliable classification. On the other hand, if the relevance of the entry page is very certain, it is wasteful to proceed crawling to discover the obvious. Our proposed website crawler handles candidate sites that cannot be reliably classified, based on the entry page more carefully than those where a certain classification can immediately be obtained.

To demonstrate this difference, we ran the focused webpage crawler for each of the first 4 topics listed in Table 5.4 and applied a website classifier to the results. Additionally, we performed two different website crawls to demonstrate the capability of the website crawler to find a suitable trade-off between accuracy and efficiency by adjusting the confidence threshold. The first one uses again the parameter setting providing maximum accuracy ($p_{threshold}$ 90%). Thus, we can judge the overhead for the additional accuracy. The second crawl used a confidence value of 70 %. Due to this rather soft breaking condition, the second crawl usually visited very few pages per website, but provided less reliable results. Figure 5.15 displays the average pprs-rate over the last 5,000 webpages for the first four topics displayed in Table 5.4. Recall that the pprs-rate measures the average number of additional webpages that are downloaded until a new relevant website is discovered. Let us note that the crawls vary in length, since we terminated crawling after reaching at least 2,500 relevant websites regardless how many webpages

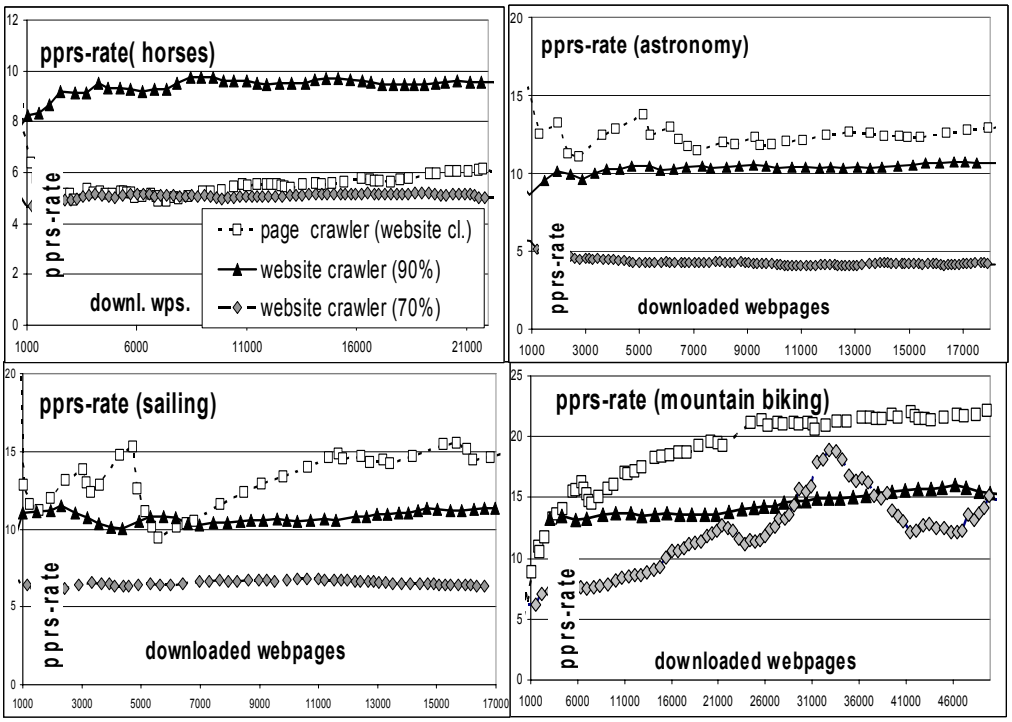


Figure 5.15: Pprs-rates (average of the last 5,000 pages) for each topic and each crawler.

where downloaded. For three out of four topics even the website crawler aiming at more accurate results ($p_{threshold}$ 90%) achieved a lower pprs-rate than the page crawler. For the topic mountain biking, it needed approximately 7 pages less than the page crawler to find an additional relevant domain at the end of the crawl. Thus, even when returning more reliable results, in most cases the website crawler gained an efficiency advantage compared to the page crawler. For all topics, the website crawler with a 70% confidence threshold clearly outperformed the two comparison partners with respect to efficiency. For the topic astronomy, it visited only about five additional webpages until it retrieved another relevant site. Due to the large number of results, we could not verify the entire result set, but a manual analysis of a sample supported our claim of more reliable results even for the 70% web-

site crawler. To conclude, our experimental evaluation demonstrates that a focused website crawler is, for similar accuracy requirements, clearly more efficient for retrieving relevant websites than a focused webpage crawler with website post-processing. In an alternative scenario, when achieving a comparable pprs-rate, the focused website crawler returns more accurate results.

5.5 Summary

In this chapter, we introduced a new direction of web content mining called website mining. While most directions of web content mining are concerned with the retrieval, classification and grouping of single webpages, website mining is aimed at websites. A website is a linked set of webpages that is published by the same group, person or organization and usually serves a common purpose. The retrieval of websites answers queries one a more abstract level than webpages. For example, the search for companies of a certain business is answered best by returning the websites of the companies instead of every single HTML document these companies publish. Other applications of website mining are the automatic extension of directory services and the restriction of the search space for the search of specific webpages.

To retrieve websites that are relevant to a certain topic, it is necessary to distinguish relevant from irrelevant websites. Therefore, the classification of websites is a key task of website mining. For example, a website classifier could be used to map new websites to the classes in a web directory. To solve this problem, we proposed several new solutions. The classification of the homepage turned out to be not sufficient for reliable classification results. The reason for the bad performance of this approach is that the homepage is not necessarily a good representation of the purpose of a website. Thus, to improve the results, it is advisable to employ more than one webpage of each website to determine the correct class. The next approach is called superpage classification. This approach condenses all webpages of the website into one word vector. However, the classification of this superpage does not yield

good classification accuracy either.

After introducing these naive solutions, we introduced two directions of website classification that provide better results. The first uses a preprocessing step by assigning page classes to each webpage of the website to be classified. To achieve this labelling for each website class, a set of specific page classes is specified. Afterwards a text classifier is used to map the webpages to the specified page classes. A website can now be represented as a so-called website tree or as a topic frequency vector (TVF). TVFs can be classified by established classification methods like naive Bayes classifiers. To classify website trees, we applied Markov classifiers of varying order to have the possibility to consider the tree structure for the classification result.

The second direction of website classification does not need the use of page classes and is therefore much easier to apply. It is based on kNN classification and treats a website directly as multi-instance object or set of feature vectors. The proposed kNN classifier uses a distance function for sets of vectors that is called "sum of minimum distances"(SMD). To increase the performance of this approach the training set for each class is condensed into a so-called centroid set. A website is compared to a centroid set by a modification of SMD called half-SMD.

After introducing classification methods, we introduced a pruning method that restricts the number of webpages that is used for website classification. This method is applied to incremental website classifiers and is capable to reduce the classification time and to increase the accuracy. The idea is that a website is descended, beginning with the homepage. After each additional webpage the path from the homepage to this page is measured with respect to its use to make a class decision. Thus, pathes that do not provide useful information are pruned and therefore not followed any further.

To compare the introduced methods, we evaluated our classifiers on two testbeds. One provided page classes and the other did not. It turned out that the method using page classes performed best if the effort is spent to specify page classes and label enough training documents to achieve suitable

webpage labels. On the second testbed, the kNN classification was compared to the methods using page labels by taking the website classes as page classes as well. In these experiments, it turned out that this simple solution is not sufficient to label the webpages well enough and that in the case that no page classes are provided the kNN approach using centroid sets, called centroid set classifier, is the better choice.

After we treated the problem of recognizing relevant websites, we turned to the task of actively searching the WWW for relevant sites. For that purpose, we introduced a focused crawler searching for relevant websites instead of webpages. The proposed two-level architecture allows us to control the number of pages to be downloaded from each website and to find a good trade-off between accurate classification and efficient crawling. The external crawler views the web as a graph of linked websites, selects the websites to be examined next and invokes internal crawlers. An internal crawler views the webpages of a single given website and performs focused page crawling within that website. In our experimental evaluation, we demonstrated that reliable website classification requires to visit more than one but less than all pages of a given site. Furthermore, we compared our proposed crawler to a focused webpage crawler that handles the concept of websites in a corresponding step of post-processing. For the same efficiency (measured by the number of pages downloaded per relevant site), the website crawler achieved significantly higher classification accuracy than its comparison partner. For comparable accuracy, the website crawler needed a considerably smaller rate of visited pages per relevant site. These results support our claim that in ordear to achieve high classification accuracy and efficiency of crawling, a focused website crawler requires a two-level architecture and corresponding crawl strategies with an explicit concept of websites.

Chapter 6

Conclusions about Multi-Instance Data Mining

This chapter concludes the part of the thesis that deals with data mining in multi-instance objects. The solutions described in the former two chapters solve practical applications and thus also contain methods solving problems that are not directly related to multi-instance data mining. This chapter sums up the introduced methods from the multi-instance point of view. Furthermore, we draw general conclusions about multi-instance data mining by analyzing the developed solutions.

6.1 Summary of the Introduced Multi-Instance Data Mining Techniques

As mentioned in the introduction, the usefulness of compound object representations for data mining strongly depends on the given application. Thus, the former two chapters provided advanced solutions for two real-world applications. Our experimental results underline the benefits of the introduced data mining methods. However, real-world problems demand solutions for several problems which are specific for the given application. In the following, we will sum up the multi-instance aspects of our solutions and draw general conclusions about multi-instance data mining.

In this part, we introduced solutions for clustering and classification of multi-instance objects. In chapter 4, we introduced a new similarity search system for multi-instance objects. This system is based on the minimal matching distance and uses multi-step query processing to speed up similarity queries. The evaluation of our approach used the density-based clustering algorithms OPTICS [ABKS99] and demonstrated that employing multi-instance representations is capable to provide a more intuitive notion of similarity.

In chapter 5, we employed multi-instance objects to represent websites. To classify multi-instance objects, we pursued two different strategies, aggregation and kNN classification. Aggregation-based classifiers like the topic frequency vector approach and the 0-order Markov tree, do not directly employ multi-instance objects but aggregate each multi-instance object into a single feature vector. Afterwards the multi-instance objects are represented as one single feature vector that can be used as input for established classification methods. In our solution, this aggregation was achieved by defining groups of instances for each website class which are called page classes. Then, another classifier is used to map each instance to the page class it most likely belongs to. To derive a feature vector, we built the histogram with respect

to these groups. We employed varying classifiers to process this group. The 0-order Markov tree does not exactly match into this pattern because it does not explicitly build up a histogram, but incrementally derives a class prediction at each step. However, the method also employs a page classifier that maps each object to a page class.

The second approach used kNN classification directly on the multi-instance objects. Therefore, we employed a distance function called sum of minimum distances. To speed up this classification approach, we applied the idea of centroid based kNN classification to multi-instance objects. Since it is not possible to directly derive a centroid from several multi-instance objects, we introduced the centroid set to represent a set of multi-instance objects. The centroid set is built by clustering the union of the instances of a set of multi-instance objects and then calculating the set of cluster centroids. Therefore, it contains a representative for each important type of instance occurring in the set of multi-instance objects. Based on the centroid set, classification was improved with respect to accuracy and efficiency compared to plain kNN classification.

In the last section of this chapter, a focused website crawler is described. A core component of this crawler is the so-called internal crawler that classifies websites while crawling them. From the multi-instance data mining point of view, the internal crawler offers another approach to multi-instance classification. The idea is to treat a class of multi-instance objects as a statistical process that generates different instances with varying probabilities. A multi-instance object is now the result of employing this process several times. This model is based on the assumption that the average number of instances is approximately the same for each class. Furthermore, we assume that the instances are independently generated, i.e. the occurrence of one instance does not influence the occurrence probability of any other instance within the same object. This approach can be used to define a Bayes classifier for multi-instance objects. Furthermore, the approach is useful to point out in which cases multi-instance classification yields benefits and how it does

relate to classical multi-instance learning.

Assuming a two class problem of multi-instance objects where the number of instances within an object is approximately equal for both classes and the instances are generated independently. If the probability that the processes of both classes generate the same instance is rather low, the benefit from having more than one instance is rather low as well. In this setting a single instance is usually sufficient to make an accurate class prediction. However, if the probability that both processes generate the same instance with a high probability, it becomes more unlikely that a single instance might contain the information that is necessary for a correct class prediction. In these cases, employing additional instances increases the probability that some of the treated objects are specific for one of the classes.

In this model multi-instance learning is a special case in which the statistical processes provide the following characteristics. The process modelling the irrelevant class cannot generate relevant objects or generates them with a zero probability. All other instances i can be generated by both processes with some probability $Pr[i|c] > 0$ where $Pr[i|relevant] \leq Pr[i|irrelevant]$. If a relevant instance occurs within a classified object the probability of the irrelevant class drops to zero. In all other cases, the irrelevant class is predicted because the probability for irrelevant instances tends to be higher for the irrelevant class.

6.2 Conclusion about Multi-Instance Data Mining

The above contributions lead us to some general conclusions about multi-instance data mining. Several distance measures on multi-instance objects have been introduced that are suitable for different applications. The mininal Hausdorff distance is known to provide a suitable kNN classifier for classical multi-instance learning[WZ00]. In chapter 4, we use minimal matching dis-

tance for clustering and in chapter 5 we use the sum of minimum distances for kNN classification. Thus, distance based data mining proves to be a valuable approach for multi-instance data mining. However, this approach still suffers from the following drawbacks:

- Though for a some of the mentioned problems a suitable distance measures was found, it is not clear which of the distance measures is suited best for a new application. This drawback is a general problem of distance based data mining. However, for multi-instance problems it is very critical because the ideas of the distance measures vary very strongly. For example, the minimal Hausdorff distance defines a distance value of two objects as the distance of the closest pair of instances while minimal matching distance compares disjoint pairs of all instances.

- Another problem of distance based data mining is efficiency. Many data mining algorithms need large numbers of distance calculations and most distance measures for multi-instance objects are very expensive, i.e. one distance calculation has quadratic or even cubic time complexity with respect to the maximum number of instances in the compared objects. Furthermore, often similarity queries are difficult to speed up because the used similarity measures might not even be metric. To avoid this problem, we introduced a filter step and kNN classification based on centroid sets. However, a general approach for speeding up distance based multi-instance data mining is unlikely to exist due to the strongly varying notion of multi-instance distance functions.

Besides distance based data mining, we applied an aggregation-based approach to handle multi-instance classification. Aggregation can be considered as an additional preprocessing step that transforms a multi-instance object to a feature vector. The resulting feature vector is used to represent the multi-instance object when applying standard data mining algorithms. Though we

found a solution for the given application, aggregation is not generally applicable. Finding a suitable aggregation function is also strongly application dependent like the selection of a suitable distance measure. Furthermore, not all relationships between sets of instances are expressible by a single feature vector.

The internal crawler used a statical process to model a class of multi-instance objects. This approach is the most general of the introduced techniques for multi-instance data mining because it can handle different kinds of relationships between the instances of two object. Depending on the used distribution function for each process this approach is capable to decide the specificity of a single instance for a class. However, this approach is suitable for classification only and is based on the assumption of independence between the instances of one object. Another important problem is the selection of the distribution function that is used to generate the instances of a class. Last but not least, the number of instances of an object is treated as equally distributed for each class which might not be realistic.

To conclude, for classification and clustering of multi-instance objects a careful examination of the given application is necessary. Depending on the relationships between the instances of two compared objects and the degree the instances within an object are correlated, varying methods or similarity measures are applicable. Thus, finding a proper solution depends on the given application to a very high degree.

Part III

Data Mining in Multi-Represented Objects

Chapter 7

Multi-Represented Objects

Multi-represented objects are the second basic type of compound objects besides multi-instance objects. A multi-represented object consists of a tuple of feature representations. Each feature representation belongs to a different feature space and represents another view on the object. In this chapter, we give a brief introduction to multi-represented objects and survey important applications.

7.1 Multi-Represented Objects

Many important areas of KDD are concerned with finding useful patterns in large collections of complex objects. Images, biomolecules or CAD parts are only some examples of complex objects that are in the center of interest of many researchers. However, the more complex a type of object is the more feature transformations exist that try to extract relevant features and construct a meaningful object representation. For example, [VT00] surveys a variety of systems for content based image retrieval and their various feature transformations. Other examples are the feature transformations of CAD-parts described in chapter 4.3. All of these feature transformations are well suited for different applications and treat a data object from another point of view. For example, shape descriptors are well suited for spotting certain objects on images, whereas color histograms are better suited to compare complete sceneries.

For data mining, the existence of multiple feature transformations is often problematic because it is not clear which of the representations contains the features that are needed to achieve the desired results. Thus, the selection of a feature transformation is often a difficult and crucial decision that strongly influences the resulting patterns. Clearly, incorporating all available feature transformations offers a more complete view of a data object and minimizes the hazard that the information that is necessary to derive meaningful patterns are not contained in the object representation. On the other hand, considering too many aspects of an object is often problematic as well. The found patterns are often very complicated and lose generality. Furthermore, the efficiency of the data mining algorithms suffers strongly since much more features have to be processed. Thus, integrating multiple representation yields chances as well as drawbacks.

In this chapter, we introduce data mining techniques that allow data mining for multi-represented objects. The idea of multi-represented data mining is to use compound objects, i.e. the tuples of all available object repre-

sentations as input for the data mining algorithms. The use of this object representation yields solutions that can draw advantages out of additional information, as we will see in the next two chapters.

Formally, we define a multi-represented object o as a tuple $(r_1, \ldots, r_k) \in R_1 \times \ldots \times R_k$. The representation space $R_i = F_i \cup \{" - "\}$ consists of a feature space F_i for a given representation and a symbol " − " to symbolize missing representation vectors. The consideration of missing objects is an important case that must be considered to solve real-world problems. For example, in image databases new images often lack a text annotation, or in protein databases the three dimensional structure of an already known protein is not explored yet.

7.2 Applications of Multi-Represented Objects

As mentioned above, one reason for the occurrence of multi-represented objects is the existence of several useful feature transformations that model different, important aspects of the same data objects, e.g. shape and color of an image. Another reason for the occurrence of multi-represented objects is the existence of different measuring techniques for an object. A satellite might offer several pictures of the same area for varying color spectra like infrared or ultraviolet. One final reason for the occurrence of multi-instance objects is that several databases store the same data object independently. If these databases are integrated into a global data collection, the global view contains a representations from each of the source database. For example, the efforts to build up integrated databases for terror prevention provide a picture of various facets of a person after data linkage is done. Each data source represents another type of information about a potential terrorist. Thus, this highly delicate application uses multi-represented data as well.

Important applications providing multi-represented objects are:

- **Biomolecules**

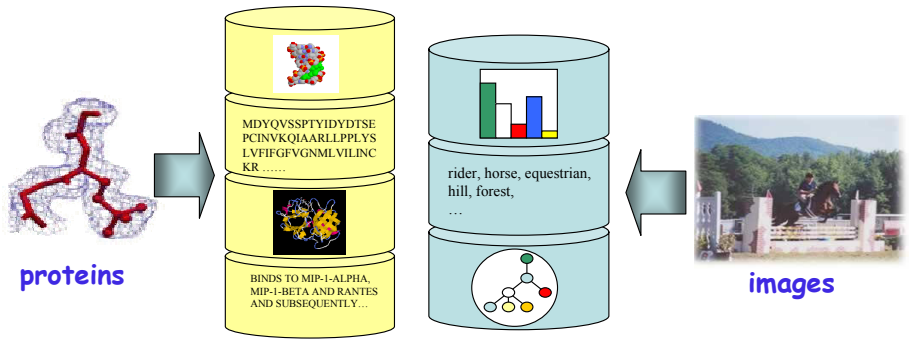


Figure 7.1: Proteins and images can be described by multi-represented objects.

Biomolecules are described by various representations like text annotations, sequential data, e.g. genes or amino acid sequences, or structural features,e.g. the secondary structure or the three dimensional structure.

- **General Images**
  As mentioned before, for images there exists a large variety of possible feature transformations that try to express different kinds of content of an image like shapes, textures and colors.

- **CAD-Parts**
  CAD-parts can be transformed into feature representations by a variety of transformations. Furthermore, CAD parts are often described by text containing structural, functional and commercial information.

- **Biometry**
  Another interesting area providing multi-represented data are biometric applications. A person can identify herself by finger prints, her iris pattern, her voice or her face.

- **Satellite Images**

Satellites often make several pictures of an area using different frequency spectra like infrared or ultraviolet.

This enumeration lists only some of the applications of multi-represented objects and is far from being complete. In the following, we will concentrate us on the first two applications. Data mining in molecular biological databases and image databases. For those two applications, we will introduce techniques that are capable to draw advantages out of the more meaningful input space. Figure 7.1 illustrates the first two examples.

Chapter 8

Clustering of Multi-Represented Objects

Clustering is one of the most important data mining tasks and many clustering algorithms were introduced by the research community so far. Usually, these methods are targeted at finding groups of similar objects in one type of object representation using one distance function. In this chapter, density-based clustering of multi-represented objects is examined and two clustering methods that are based on DBSCAN are introduced. The introduced methods are applied to two important types of multi-represented data, protein data and images.

8.1 Introduction

In recent years, the research community spent a lot of attention to clustering resulting in a large variety of different clustering algorithms [DLR77, EKSX96, ZRL96, WYM97, AGGR98, GRS98, ABKS99, HK01]. However, all those methods are based on one representation space, usually a vector space of features and a corresponding distance measure. But for a variety of modern applications such as biomolecular data, CAD-parts or multimedia files mined from the internet, it is problematic to find a common feature space that incorporates all given information. In this chapter, we therefore introduce a clustering method that is capable to handle multiple representation.

To cluster multi-represented data, using the established clustering methods would require to restrict the analysis to a single representation or to construct a feature space comprising all representations. However, the restriction to a single feature space would not consider all available information and the construction of a combined feature space demands great care when constructing a combined distance function. Since the distance functions best-suited for each representation might not even provide the same value set, it is difficult to find a proper combination that gives a meaningful distance. Another important problem is that several data objects might not provide all possible representations. For example, finding all representations of a protein is expensive and time consuming. Thus, there are much less three dimensional models of proteins than there are amino acid sequences available so far. In these cases, the combined distance function would need to handle missing representations adequately. A last drawback of combined feature spaces is the following. Since many clustering algorithms are based on similarity queries, the use of index structures is usually very beneficial to increase the efficiency, especially for large data sets. For the design of a proper combined distance measure, this is another important constraint to consider, since the combination of the distance functions needs to be at least

metric to allow the use of an index structure.

In this chapter, we propose a method to integrate multiple representations directly into the clustering algorithm. Our method is based on the density-based clustering algorithm DBSCAN [EKSX96] that provides several advantages over other algorithms, especially when analyzing noisy data. Since our method employs a separated feature space for each representation, it is not necessary to design a new suitable distance measure for each new application. Additionally, the handling of objects that do not provide all possible representations is integrated naturally without defining dummy values to compensate the missing representations. Last but not least, our method does not require a combined index structure, but benefits from each index that is provided for a single representation. Thus, it is possible to employ highly specialized index structures and filters for each representation. We evaluate our method for two example applications. The first is a data set consisting of protein sequences and text descriptions. Additionally, we applied our method to the clustering of images retrieved from the internet. For this second data set, we employed two different similarity models. The introduced solutions were published in [KKPS04a].

The rest of the chapter is organized as follows. After this introduction, we present related work. Section 8.3 formalizes the problem and introduces our new clustering method. In our experimental evaluation that is given in section 8.4, we introduce a new quality measure to judge the quality of a clustering with respect to a reference clustering and display the results achieved by our method in comparison with the other mentioned approaches. The last section summarizes the chapter.

8.2 Related Work

As mentioned in the introduction, the research community has developed a variety of algorithms to cluster data for various applications [ABKS99, EKSX96, HK98, GRS98, ZRL96, XEKS98]. Most of these algorithms are

designed for one feature space and one distance function to represent the data objects. Thus, to apply these algorithms to multi-represented data, it is necessary to unite the representations into one common feature space.

A similar setting to the clustering of multi-represented objects is the clustering of heterogenous or multi-typed objects [WZC+03, ZCM02] in web mining. In this setting, there are also multiple databases, each yielding objects in a separated data space. Each object within these data spaces may be related to an arbitrary amount of data objects within the other data spaces. The framework of reinforcement clustering employs an iterative process based on an arbitrary clustering algorithm. It clusters one dedicated data space while employing the other data spaces for additional information. It is also applicable for multi-represented objects. However, due to its dependency on the data space, it is not well suited to solve our task. Since to the best of our knowledge reinforcement clustering is the only other clustering algorithm directly applicable to multi-represented objects, we use it for comparison in our evaluation section.

The goal of clustering multi-represented objects is to find a global clustering for data objects that might have representations in multiple data spaces. The setting of reinforcement clustering is to cluster the data within one data space while using the related data spaces for additional information. Since the results may vary for different starting representations, the application of reinforcement clustering is problematic. It is unclear how many iterations are needed until a common clustering for all representations is found and if the algorithm reaches a common clustering at all for an arbitrary number of iterations. Let us note that this is not a problem in the original use of reinforcement clustering, but causes a major problem when applying it to multi-represented objects.

Our method is based on the density based clustering algorithm DBSCAN, that was introduced in chapter 2.2.3.

8.3 Clustering Multi-Represented Objects

Let DB be a database consisting of n objects. Let $R := \{R_1, ..., R_m\}$ be the set of different representations existing for objects in DB. Each object $o \in DB$ is therefore described by maximally m different representations, i.e. $o := \{R_1(o), R_2(o), ..., R_m(o)\}$. If all different representations exist for o, than $|o| = m$, else $|o| < m$. The distance function is denoted by $dist$. We assume that $dist$ is symmetric and reflexive. In the following, we call the ε_i-neighborhood of an object o in one special representation R_i its local ε-neighborhood w.r.t. R_i.

Definition 8.1 (local ε_i-neighborhood w.r.t R_i)
Let $o \in DB$, $\epsilon_i \in \mathbb{R}^+$, $R_i \in R$, and $dist_i$ the distance function of R_i. The local ε_i-neighborhood w.r.t. *R_i of o, denoted by $\mathcal{N}_{\varepsilon_i}^{R_i}(o)$, is defined by*

$$\mathcal{N}_{\varepsilon_i}^{R_i}(o) = \{x \in DB \mid dist_i(R_i(o), R_i(x)) \leq \varepsilon_i\}.$$

Note that ε_i can be chosen optimally for each representation. The simplest way of clustering multi-represented objects is to select one representation R_i and cluster all objects according to this representation. However, this approach restricts data analysis to a limited part of the available information and does not use the remaining representations to find a meaningful clustering. Another way to handle multi-represented objects is to combine the different representations and use a combined distance function. Then, any established clustering algorithm can be applied. However, it is very difficult to construct a suitable combined distance function that is able to fairly weight each representation and handle missing values. Furthermore, a combined feature space does not profit from specialized data access structures for each representation.

The idea of our approach is to combine the information of all different representations as early as possible, i.e. during the run of the clustering algorithm, and as late as necessary, i.e. after using the different distance functions of each representation. To do so, we adapt the core object property

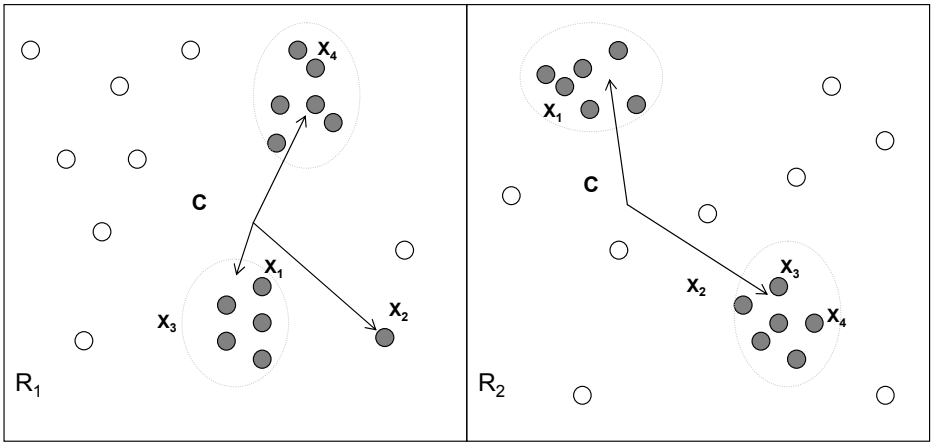


Figure 8.1: Local clusters and a noise object that are aggregated to a multi-represented cluster **C**.

proposed for DBSCAN. To decide whether an object is a core object, we use the local ε-neighborhoods of each representation and combine the results to a global neighborhood. Therefore, we have to adapt the predicate direct density-reachability as proposed for DBSCAN. In the next two subsections, we will show how we can use the concepts of union and intersection of local neighborhoods to handle multi-represented objects.

8.3.1 Union of Different Representations

This variant is especially useful for sparse data. In this setting, the clustering in each single representation will provide several small clusters and a large amount of noise. Simply enlarging ε would relief the problem, but on the other hand, the separation of the clusters would suffer. The union-method assigns objects to the same cluster if they are similar in at least one of the representations. Thus, it keeps up the separation of local clusters, but still overcomes the sparsity. If the object is placed in a dense area of at least one representation, it is still a core object regardless of how many other representations are missing. Thus, we do not need to define dummy values. Figure 8.1 illustrates the basic idea of the union method. We adapt some of

the definitions of DBSCAN to capture our new notion of clusters. To decide whether an object o is a union core object, we unite all local ε_i-neighborhoods and check whether there are enough objects in the global neighborhood, i.e. whether the global neighborhood of o is dense.

Definition 8.2 (union core object)
Let $\varepsilon_1, \varepsilon_2, ..., \varepsilon_m \in \mathbb{R}^+$, and $MinPts \in \mathbb{N}$. An object $o \in DB$ is called union core object*, denoted by $\text{CoreU}^{MinPts}_{\varepsilon_1,..,\varepsilon_m}(o)$, if the union of all local ε-neighborhoods contains at least $MinPts$ objects, formally:*

$$\text{CoreU}^{MinPts}_{\varepsilon_1,..,\varepsilon_m}(o) \Leftrightarrow | \bigcup_{R_i(o) \in o} \mathcal{N}^{R_i}_{\varepsilon_i}(o) | \geq MinPts.$$

Definition 8.3 (direct union-reachability)
Let $\varepsilon_1, \varepsilon_2, .., \varepsilon_m \in \mathbb{R}^+$, and $MinPts \in \mathbb{N}$. An object $p \in DB$ is directly union-reachable *from $q \in DB$ if q is a union core object and p is an element of at least one local $\mathcal{N}^{R_i}_{\varepsilon_i}(q)$, formally:*

$$\text{DirReachU}^{MinPts}_{\varepsilon_1,..,\varepsilon_m}(q,p) \Leftrightarrow \text{CoreU}^{MinPts}_{\varepsilon_1,..,\varepsilon_m}(q) \wedge \exists\, i \in \{1,..,m\} : R_i(p) \in \mathcal{N}^{R_i}_{\varepsilon_i}(q).$$

The predicate direct union-reachability is obviously symmetric for pairs of core objects because the $dist_i$ are symmetric distance functions. Thus, analogously to DBSCAN reachability and connectivity can be defined.

8.3.2 Intersection of Different Representations

The intersection method is well suited for data containing unreliable representations, i.e. there is a representation, but it is questionable whether it is a good description of the object. In those cases, the intersection-method requires that a cluster should contain only objects which are similar according to all representations. Thus, this method is useful if all different representations exist but the derived distances do not adequately mirror the intuitive notion of similarity. The intersection-method is used to increase the cluster quality by finding purer clusters.

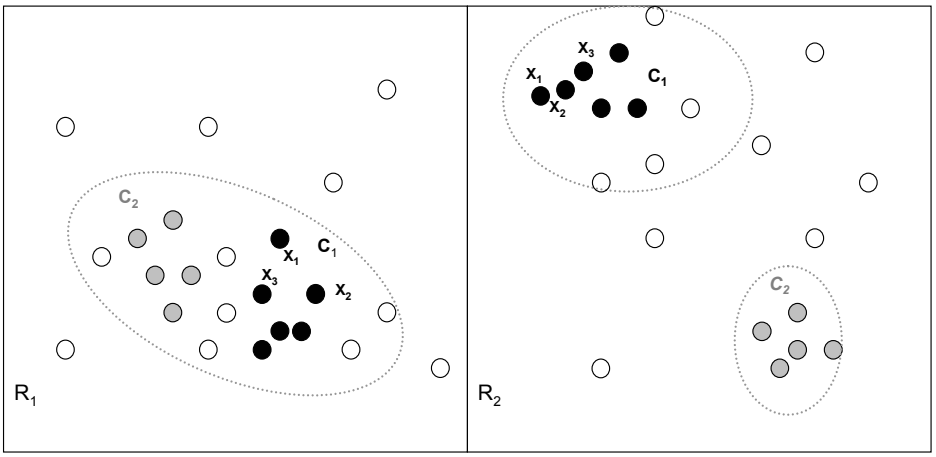


Figure 8.2: The right figure illustrates, how the intersection-method divides a local clustering into clusters C_1 and C_2.

To decide, whether an object o is an intersection core object, we examine whether o is a core object in each involved representation. Of course, we use different ε-values for each representation to decide whether there are locally enough objects in the ε-neighborhood. The parameter $MinPts$ is used to decide, whether there are globally still enough objects in the ε-neighborhood, i.e. the intersection of all local neighborhoods contains at least $MinPts$ objects.

Definition 8.4 (intersection core object)
Let $\varepsilon_1, \varepsilon_2, ..., \varepsilon_m \in \mathbb{R}^+$, and $MinPts \in \mathbb{N}$. An object $o \in DB$ is called intersection core object*, denoted by* $\text{CoreIS}^{MinPts}_{\varepsilon_1,..,\varepsilon_m}(o)$*, if the intersection of all its local ε_i-neighborhoods contain at least $MinPts$ objects, formally:*

$$\text{CoreIS}^{MinPts}_{\varepsilon_1,..,\varepsilon_m}(o) \Leftrightarrow | \bigcap_{i=1,..,m} \mathcal{N}^{R_i}_{\varepsilon_i}(o) | \geq MinPts.$$

Using this new property, we can now define direct intersection-reachability in the following way:

Definition 8.5 (direct intersection-reachability)
Let $\varepsilon_1, \varepsilon_2, ..., \varepsilon_m \in \mathbb{R}^+$, and $MinPts \in \mathbb{N}$. An object $p \in DB$ is directly

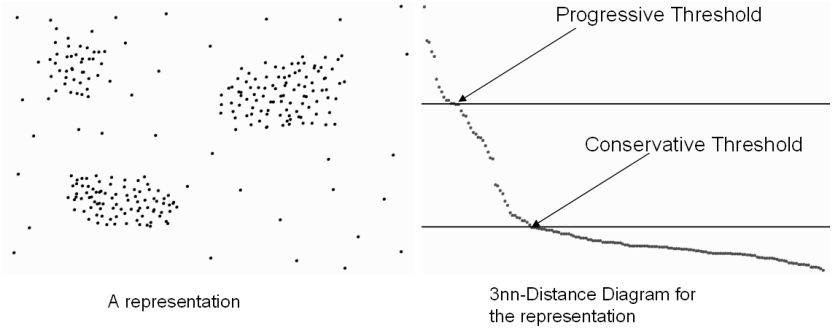


Figure 8.3: A 2D example data set and the corresponding *3nn*-distance diagram.

intersection-reachable *from* $q \in DB$ *if* q *is an intersection core object and* p *is an element of all local* $\mathcal{N}_\varepsilon^q$*, formally:*

$$\text{DirReachIS}_{\varepsilon_1,..,\varepsilon_m}^{MinPts}(q,p) \Leftrightarrow \text{CoreIS}_{\varepsilon_1,..,\varepsilon_m}^{MinPts}(q) \wedge \forall i = 1,..,m : R_i(p) \in \mathcal{N}_{\varepsilon_i}^{R_i}(q) \ .$$

Again, reachability and connectivity can be defined analogously to DBSCAN. Figure 8.2 illustrates the effects of this method.

8.3.3 Determination of Density Parameters

In [EKSX96], a heuristic is presented to determine the ε-value of the "thinnest" cluster in the database. This heuristic is based on a diagram that represents sorted *knn*-distances of all given objects. In the case of multi-represented objects, we have to choose ε for each dimension separately, whereas *MinPts* can be chosen globally. A user determines a value for global *MinPts*. The system computes the kNN-distance diagrams for the given global *MinPts*, i.e. one diagram for every representation. The user has to choose a so-called border object $o_b order$ for each representation. The ε for the i-th representation is given by the kkNN-distance of the border object of R_i. An example of a kNN-distance diagram is shown in figure 8.3. Let us note that this method still allows a certain range of ε-values to be chosen. The selection should mirror the different requirements of the proposed methods. For the union

	Set 1	Set 2	Set 3	Set 4	Set 5
Name	Isomerase	Lyase	Signal Transducer	Oxido-reductase	Transferase
Classes	16	35	39	49	62
Objects	501	1,640	2,208	3,399	4,086

Table 8.1: Description of the protein data sets.

method, it is more advisable to choose a lower or conservative value, since its characteristic demands that the elements of the local ε-neighborhood should really be similar. For the intersection-method, the ε-value should be selected progressively, i.e. at the upper rim of the range. This selection reflects that the objects of a cluster need not to be too similar for a single representation because it is required that they are similar with respect to all representations.

8.4 Performance Evaluation

To demonstrate the capability of our method, we performed a thorough experimental evaluation for two types of applications. We implemented the proposed clustering algorithm in Java 1.4. All experiments were processed on a work station with a 2.6 GHZ Pentium IV processor and 2 GB main memory.

8.4.1 Deriving Meaningful Groupings in Protein Databases

The first set of experiments was performed on protein data that is represented by amino acid sequences and text descriptions. Therefore, we employed entries of the SWISS-PROT protein database [BBA+03]and transformed each protein into a pair of feature vectors. Each amino acid sequence was mapped into a 436 dimensional feature space. The first 400 features are 2-grams of successive amino acids. The last 36 dimensions are 2-grams of 6 exchange groups that the single amino acids belong to [DK02]. To compare the de-

rived feature vectors, we employed the Euclidian distance. To process text documents, we rely on projecting the documents into the feature space of relevant terms. Documents are described by a vector of term frequencies weighted by the inverse document frequency (TFIDF) [Sal89]. We chose 100 words of medium frequency as relevant terms and employed cosine distance to compare the TFIDF-vectors. Many SWISS-PROT entries are mapped to the classes of Gene Ontology [Con00]. To have a reference clustering for evaluation, we chose five different functional groups of Gene Ontology which are linked by SWISS-PROT (cf. Table 8.1) and used the subclasses within these groups as clusters in the reference clustering. Thus, we are able to measure a clustering of SWISS-PROT entries by the degree it reproduces the class structure provided by Gene Ontology.

To have an exact measure for this degree, we employed the class entropy in each cluster. Let us note that we chose the entropy as measure for cluster quality because our reference clustering does not provide real clusters but classes. Many classes in Gene Ontology do not have a consistent character and are divided into sub classes itself. The entropy considers a cluster as good as long as its objects belong to the same class. The effect that the ideal cluster consists of a clustering where each object corresponds to its own cluster, is avoided by the constraint that any type of core object needs at least k elements in its ε-neighborhood in one of the representations. Thus, each cluster consists of at least k objects.

There are two effects that have to be considered to obtain a fair measure of a clustering with noise. First, a large cluster of a certain entropy should contribute more to the overall quality of the clustering than a rather small cluster providing the same quality. The second effect is that a clustering having a 5 % noise ratio should be ranked higher than a clustering having the same average entropy for all its clusters, but contains 50 % noise. To consider both effects, we propose the following quality measure for comparing different clusterings with respect to a reference clustering.

Definition 8.6 *Let O be the set of data objects, let $C = \{C_i | C_i \subset O\}$ be the set of clusters and let $K = \{K_i | K_i \subset O\}$ be the reference clustering of O. Then we define:*

$$Q_K(C) = \sum_{C_i \in C} \frac{|C_i|}{|O|} \cdot (1 + entropy_K(C_i))$$

where $entropy_K(C_i)$ denotes the entropy of cluster C_i with respect to $MinPts$.

The idea is to weight every cluster by the percentage of the complete data objects that are part of this cluster. Thus, smaller clusters are less important than larger ones and a clustering providing an extraordinary amount of noise can contribute only the percentage of clustered objects to the quality. Let us note that we add 1 to the cluster entropies. Therefore, we measure the reference clustering $MinPts$ with the quality score of 1 and a worst case clustering with the score of 0, e.g. no clusters are found at all.

To relate the quality of the clustering achieved by our methods to the results of former methods, we compared it to four alternative approaches. First, we clustered text and sequences separately, using only one of the representations. A second approach combines the features of both representations into a common feature space and employs the cosine distance to relate the resulting feature vectors. As this is the only other clustering method that is able to handle multi-represented data, we additionally compared reinforcement clustering using DBSCAN as underlying cluster algorithm. For reinforcement clustering, we ran 10 iterations and tried several values of the weighting parameter α. The local ε-parameters were selected as described above and we chose $k = 2$. To consider the different requirements of both methods, for each data set a progressive and a conservative ε-value was determined. All approaches were run for both settings and the best results are displayed.

The left diagram of figure 8.4 displays the derived quality for those four methods and the two variants of our method. In all five test sets, the union-method using conservative ε-values outperformed any of the other algorithms.

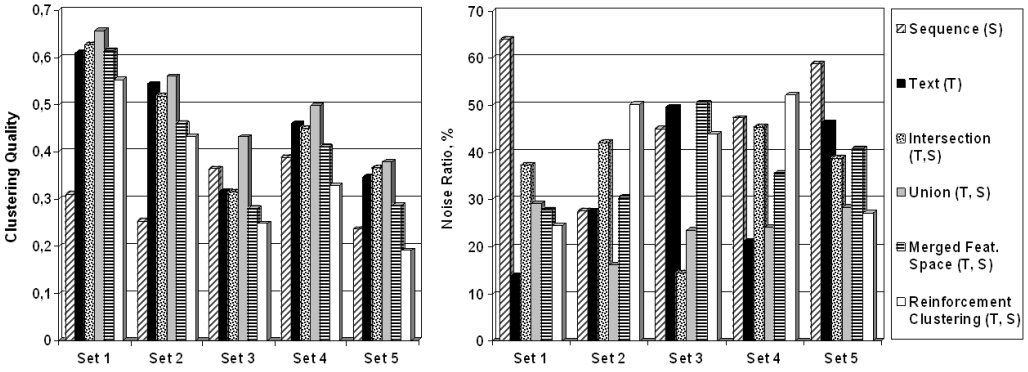


Figure 8.4: Clustering quality and noise ratio.

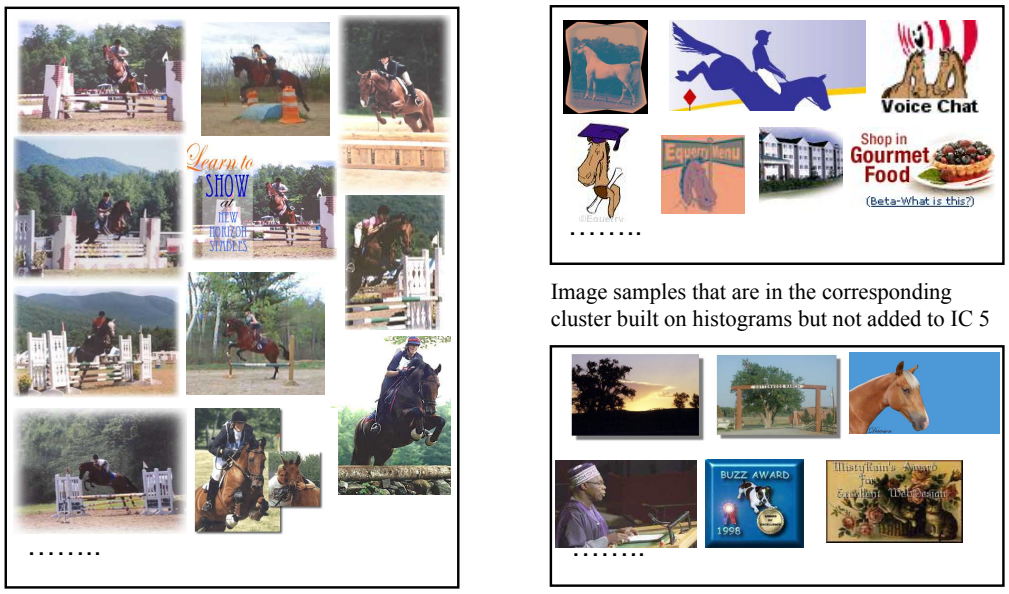
The improvement rates of the cluster quality for the union method were between 3% and 19%. Furthermore, the noise ratio for each data set was between 16% and 28% (cf. figure 8.4, right), indicating that the main portion of the data objects belongs to some cluster. The clustering based on a merged feature space always performed worse than clustering each of the representations one its own and was neither capable to outperform the intersection nor the union method. Reinforcement clustering was not well-suited to improve the clustering performance in any of the test sets either. The intersection method using progressive ε-parameters performed comparably well, but was to restrictive to overcome the sparseness of the data as good as the union-method.

8.4.2 Clustering Images by Multiple Representations

Clustering image data is a good example for the usefulness of the intersection-method. A lot of different similarity models exists for image data, each having its own advantages and disadvantages. Using for example text descriptions of images, the user is able to cluster all images related to a certain topic, but these images need not to be similar. Using color histograms instead, the images are clustered according to the distribution of color in the image. But as only the color information is taken into account a green meadow with

some flowers and a green billiard table with some colored shots on it can of course not be distinguished by this similarity model. On the other hand, a similarity model taking content information into account might not be able to distinguish images of different colors.

Our intersection approach is able to get the best out of all these different types of representations. Since the similarity in one representation is not really sound, the intersection-method is well-suited to find clusters of better quality for this application. For our experiments, we used two different representations. The first representation was a 64-dimensional color histogram. In this case, we used the weighted distance between those color histograms, represented as a quadratic form distance function as described for example in [HSE+95]. The second representation were segmentation trees. An image was first divided into segments of similar color by a segmentation algorithm. In a second step, a tree was created from those segments by iteratively applying a region-growing algorithm which merges neighboring segments if their colors are alike. In [KKSS04] an efficient technique is described to compute the similarity between two such trees, using filters for the complex edit-distance measure. Since our image data set was retrieved from the WWW, there does not exist any objective reference clustering. Thus, we simply describe the results we achieved. In general, the clusters we got using both representations were more accurate than the clusters we got using each representation separately. Of course, the noise ratio increased for the intersection-method. To demonstrate the improved cluster quality, figure 8.5 displays a sample cluster of images, we found with the intersection-method. The left rectangle contains images clustered by the intersection-method. The right rectangles display additional images that were grouped with the corresponding cluster when clustering the images with respect to a single representation. Using this method, very similar images are clustered together. When clustering each single representation, a lot of additional images were added to the corresponding cluster. As we can see, using the intersection-method the most similar images of both representations still belong to the cluster.



Cluster IC5 created by intersection of trees and histograms representations

Image samples that are in the corresponding cluster built on histograms but not added to IC 5

Image Samples that are in the corresponding cluster built on trees but not added to IC 5

Figure 8.5: Example of an image cluster.

8.5 Conclusions

In this chapter, we discussed the problem of clustering multi-represented objects. A multi-represented object is described by a set of representations where each representation belongs to a different data space. Contrary to existing approaches, our proposed method is able to cluster this kind of data using all available representations without forcing the user to construct a combined data space. The idea of our approach is to combine the information of all different representations as early as possible and as late as necessary. Thus, the core object property that was proposed for DBSCAN, is adapted to handle multi-represented objects. To decide whether an object is a core object, we use the local ε-neighborhoods of each representation and combine the results to a global neighborhood. Based on this idea, we proposed two different methods for varying applications. For sparse data, we introduced the union-method that assumes that an object is a core object, if $MinPts$

objects are found within the union of its local ε-neighborhoods. Respectively, we defined the intersection-method for data where each local representation yields rather big and unspecific clusters. Therefore, the intersection-method requires that at least $MinPts$ objects are within the intersection of all local ε-neighborhoods of a core object. In our experimental evaluation, we introduced an entropy based quality measure that compares a given clustering with noise to a reference clustering. Employing this quality measure, we demonstrated that the union method was most suitable to overcome the sparsity of a given protein data set. To demonstrate the ability of the intersection method to increase the cluster quality, we applied it to a set of images using two different similarity models.

Chapter 9

Database Integration using Classification of Multi-Represented Objects

Biological databases provide large collections of complex objects like genes and proteins. Though these databases are publicly available, their use is limited due to their varying data formats and access facilities. Thus data integration is an important task in bioinformatics. A promising approach to solve this problem is ontology-based data integration. In this chapter, we introduce a classification system mapping protein data into large ontologies of protein classes that can be used for ontology-based data integration. The introduced method is based on support vector machines and uses the class hierarchy within an ontology to speed up classification. Since biomolecules are often described by more than one representation, our approach uses multi-represented classification. Therefore, we introduce a technique called object-adjusted weighting that increases the classification accuracy by locally weighting the classification results in each representation. The methods were implemented and tested by mapping entries of the SWISS-PROT protein database [BBA+03] to the protein classes in Gene Ontology [Con00].

9.1 Introduction

In recent years, the amount of publicly available biological information has increased dramatically. As a consequence, many databases have emerged, offering diverse information on all kinds of biological entities such as proteins, nucleotides, pathways, etc. Though most of these information sources are accessible via the web, the use of the information is strongly limited due to the heterogeneity of data formats, data models, and access facilities between these sources [BK03].

A promising approach for overcoming these problems is the use of ontologies and taxonomies for data integration. Several ontologies have been developed for molecular biology but only a small fraction of them is widely accepted. One of the most popular ontologies in molecular biology is Gene Ontology (GO) [Con00] which models the function of genes and gene products, e.g. proteins. Most of the major protein databases such as SWISS-PROT [BBA+03] provide a mapping of their entries to GO. However, not all of the entries are already mapped and biologists all over the world produce new entries every day. To obtain a mapping of a so far unlinked protein database entity into GO, usually some information about the biological function of the protein representing this entry has to be explored. Since this usually has to be done manually throughout a series of biological experiments and tests, it is a very time consuming and costly task. It would be of great benefit if the mapping could be done automatically by computer-supported prediction of the biological function out of the raw data stored in the major protein databases, e.g. the amino acid sequence of a protein which can be obtained very easily, without laborious experiments. More generally, a framework for the automatic prediction of the function of biological entities such as proteins is needed to classify these entities according to ontologies such as GO.

In this chapter, we introduce a classification system that provides a good mapping for so far unlinked biological entities and gives a prediction of the biological function. The fact that the objects are not linked to classes or

other entities yet, restricts the use of general relationships modelled in the ontology. Thus, our system exploits the class inheritance of the ontology for classification, i.e. the taxonomic part of it.

Due to the nature of biological entities, our system copes with the following demands: Several instances may belong to more than one class in the taxonomy. Different instances might be placed at varying abstraction levels. At last, biological ontologies may employ multiple inheritance in order to model their classes.

Another important aspect is that the representations of biological entities can usually be derived from multiple sources, e.g. for most proteins, the amino acid sequence data and a textual descriptions of experimental results are available from databases such as SWISS-PROT [BBA+03]. For a smaller number of proteins additional data is available, e.g. the three dimensional structure. Usually such data can be derived from other public databases such as the Protein Data Bank (PDB) [BWF+00]. An important reason for the diversity of biological object representations is the fact that they are not directly observable. Thus each measuring technique might reveal another important aspect of these complex objexts that can be used for data mining.

To use all those different aspects for accurate class predictions, a flexible classifier has to be found that is capable to deal with the following problems. As the quality of each type of representation may vary for different entries and types of representations, the classifier should automatically weight the influence of each representation. Furthermore, the framework should be flexible enough to handle missing representations, i.e. if one of the object representations is missing, the classifier should still be able to make a prediction.

Since biological ontologies are built of large numbers of classes and biological entities occur in large cardinalities, good efficiency is mandatory to handle large problems in applicable time. To meet these challenges, we present a novel approach to hierarchical classification based on support vector machines which provides the following features:

- An efficient and accurate method for handling multi-classified instances employing support vector machines.
- A method for the classification of multi-represented objects that is capable to cope with missing representations.
- A discussion of methods for hierarchical classification under the aspect of large classification problems and taxonomic directed acyclic graphs instead of strict taxonomy trees.
- A thorough experimental evaluation, demonstrating effectiveness and efficiency of our prototype on several subsections of GO.

Our prototype was designed to automatically map proteins based on their SWISS-PROT entries into GO. We use sequence data and the text annotations as different representations for proteins. Let us note that further data sources such as secondary and tertiary structures can easily be incorporated into the prototype. The methods described in this chapter were published in [KKPS04b]. The rest of the chapter is organized as follows. In Section 9.2, we briefly review related work. In Section 9.3, the major concepts of our approach that cope with the challenges mentioned above are presented. The proposed prototype is evaluated based on a realistic experimental setting in section 9.4. Section 9.5 offers a summary of the presented work and gives perspectives for future work.

9.2 Related Work

Classifying biological sequences such as nucleotide or protein sequences is an active area of research. Common approaches are based on k-nearest neighbor classifiers (KNN) and all kinds of Markov models (MM), including simple MM, selective MM and higher-order MM. Examples of KNN and MM approaches are given by [DEKM98, Mou01]. KNN-methods for biological sequence classification usually use edit distance as similarity function.

Although being simple and comprehensible to biologists, these approaches suffer from the expensive computation. MM are widely used for biological sequence classification since they have an inherent ability to model sequential constraints in the data. Recently, SVMs have been applied to sequence classification by [DK02, SCW+03] and demonstrated excellent accuracy when a suitable feature extraction method is employed. To be suitable for sequence classification, an extraction method should model the sequential nature of the data. Since finding such an adequate extraction is not a trivial task, recent research addresses this challenge, e.g. [KH98, LZO99, WMSW01, DK02, SCW+03].

Text descriptions of biomolecules are transformed to feature vectors by the methods introduced in chapter 2.1.2. For the classification of the resulting vectors, several approaches have been proposed [HK00, Yan97, Roc71]. SVMs have also demonstrated their high value for making very accurate predictions in the field of text classification [Joa98].

Employing class hierarchies to improve large scale classification problems has predominantly been used in text classification. Therefore, the used taxonomies are taken from directory services for HTML documents [MRMN98, DC00], structural class systems like the U.S. patent codes or are constructed to have a proper testbed [Lar98]. The idea of hierarchical classification is that solving a set of small problems with less classes can be achieved faster and more effective than solving one large scale classification problem distinguishing a large amount of classes. To do so, several approaches have been introduced by [MRMN98, DC00, Lar98, KS97, DMS98, VMD00, WZL99]. Most of them achieved a big performance improvement and some gain in classification accuracy. However, none of these approaches examined an arbitrary shaped class system of functional data types employing multiple inheritance so far. Furthermore, only [DC00] examines the combination of support vector machines and class hierarchies, but the evaluation is based on a strict two level taxonomy tree. There have been several approaches to employ general ontologies for classification via relational learning like [CDF+99]. However,

since those approaches rely on general relations, the problem they try to solve is dissimilar to the task that is described in this chapter.

The task of learning from objects given by multiple representations has recently drawn some attention in the pattern recognition community [KHDM98, Dui02, KBD01]. In [Dui02] the author describes the method of classifier fusion to combine the results from multiple classifiers for one and the same object. Furthermore, [Dui02] surveys the four basic combination methods and introduces a combined learner to achieve combination rules offering better accuracy. However, the introduced methods does not adjust to the reliability of a local class prediction. Furthermore, the combination methods described in [Dui02] are based on the combination of distribution vectors for each representation and deriving distribution vectors from the type of multi-class SVMs, we use in our method is a not yet solved problem. In our evaluation section, we therefore compare our method to an unweighted average voting vector which is the easiest way to apply the already published methods.

9.3 Classification of Biological Objects

In this chapter, we address the problem of classifying biological objects like genes or proteins into a large class system like Gene Ontology [Con00].

The goal of classification is to learn a function $F : O \rightarrow C$ that maps as much objects $o \in O$ to their correct class $c \in C$ as possible. For training, a set of tuples (o, c) of objects o and their correct classes c is given to the classifier, the so-called training set. A variant of simple classification is multi-classification $F_{multi} : O \rightarrow 2^C$ which maps each object o to a subset of C. In our application, multi-classification is mandatory because large parts of the objects are associated with more than one class C.

In most application domains, the objects are represented by one (possibly complex) data type. Thus, the established way to process the objects $o \in O$ is to extract a set of meaningful features from the object representation. For most types of data representations, e.g. text, there already exist several

approaches of feature extraction. The derived features span the so-called feature space. Each dimension of the feature space represents a feature. Thus, an object o is represented by a feature vector. A classifier is trained on the set of feature vectors derived from the training set.

As stated above, biological entities are often built of multiple data types (representations), such as sequence data, text, etc. Thus, the input space O of our classifier F (analogously for F_{multi}) is composed of the set of different representations an object in O might have. Though building one joined feature space for all different representations like texts or sequences is principally applicable, the corresponding feature vectors might mirror the properties of the data object only poorly. We argue that classification benefits from incorporating the knowledge about the structure of the input space, i.e. the knowledge about the representation a feature is extracted from, into the classifier. In other words, features from the same representation should be treated in a more similar way than those from different representations.

A second property of biological entities could be utilized to enhance the performance of our prototype. Not only the input space O but also the output space C of F (analogously for F_{multi}) is structured. In fact, the classes in C are usually organized in a sophisticated class system like a taxonomy or an ontology (in our case the GO). To solve a classification problem, it is not necessary to consider any relations between the classes $c_i \in C$. But the fact that the classes in C are structured can be exploited for dealing with large cardinalities of C more efficiently.

In the following, we will first introduce an approach for multi-classification based on Support Vector Machines. Afterwards we will integrate the knowledge about varying object representations, i.e. the structure of the input space, to enable the accurate incorporation of as much information about the objects as possible into the classification process. Finally, hierarchical classification is discussed, utilizing the structure of the output space in order to enhance the performance of our framework.

9.3.1 Using Support Vector Machines for Making Set-Valued Predictions

Support Vector Machines (SVMs) are capable of providing superior accuracy compared to other classification methods for most representations of biological objects [Joa98, DK02]. Standard SVMs, also called binary SVMs, classifiy objects into two classes by finding a hyperplane that separates a given training set according to these classes in the best possible way. Since SVMs are only capable of making binary decisions, it is necessary to enhance them to distinguish between more than two classes and to make set-valued predictions. In [PCST00] three methods for implementing SVMs are compared that distinguish more than two classes. The first, the so-called *one-versus-rest approach* employs one binary SVM for each class to decide if an object belongs to that class or not. Thus, this approach is capable to predict any class combinations possible. For example, an object could be mapped to all classes, if all binary SVMs predict the class they distinguish from the rest of the classes. The second approach, the so-called *one-versus-one approach* uses $\frac{N \cdot (N-1)}{2}$ many binary SVMs for distinguishing between each pair of the given $N = |C|$ classes. When classifying an object, the results are aggregated into a so-called *voting vector*. This vector provides a dimension for each class and its components correspond to the number of binary SVMs that have predicted this class. Thus, there are $\frac{N \cdot (N-1)}{2}$ votes. Since there are only $N-1$ binary SVMs distinguishing a class from the other classes, a class can attain a maximum of $N-1$ votes. A single class decision is accomplished by returning the class having the maximum number of votes in the vector. Note that this maximum vote is not necessarily $N-1$. The last approach uses decision directed acyclic graphs to reduce the number of binary SVMs considered for classification to $N-1$ out of $\frac{N \cdot (N-1)}{2}$ trained SVMs. The approach trains the same number of SVMs, but does employ only a fraction of them for classification.

We choose the one-versus-one approach for our system, since voting vectors provide a meaningful intermediate result. To enable our classifier to predict a set of class combinations, we collect the set of all class combinations that occur within the training data. Afterwards, we extend the set of original classes by those class combinations. Thus, all valid combinations are predictable. This general approach is especially suitable for our application because there are several class combinations that do not make sense. For example, an object that is predicted to be a dog is unlikely to be a cat at the same time. Thus, we can limit the set of valid class combinations to all class combinations which occur in the training data. Let us note that the one-versus-rest approach also solves the problem of set-valued predictions, but offered inferior results due to the prediction of invalid class combinations (see Section 9.4 for experimental results).

A drawback of all of the mentioned approaches of multi-class SVMs is that the extra effort for introducing another class leads to the use of additional binary SVMs distinguishing the new class. To avoid this additional overhead, we do not extend the class set at once. Instead we refine the post processing of the voting vectors gained from the one-versus-one approach. Thus our classification function has the following form:

Let O be the set of objects, let C be the set of classes, let C^* be the extended class set including all valid class combinations and let V with $dim(V) = |C|$ be the vector space of voting vectors. Then our classifier has the following form:

$$Cl : O \rightarrow C^*$$
$$Cl(o) = F_2(F_1(o))$$

where $F_1 : O \rightarrow V$ and $F_2 : V \rightarrow C^*$ are classifiers. For F_1 and F_2, our prototype employs a one-versus-one multi-class support vector machine.

The vector space of voting vectors is well suited to describe the results of the first classifier. A one-versus-one multi-class SVM partitions the feature space along each of its binary SVMs. A voting vector corresponds to a

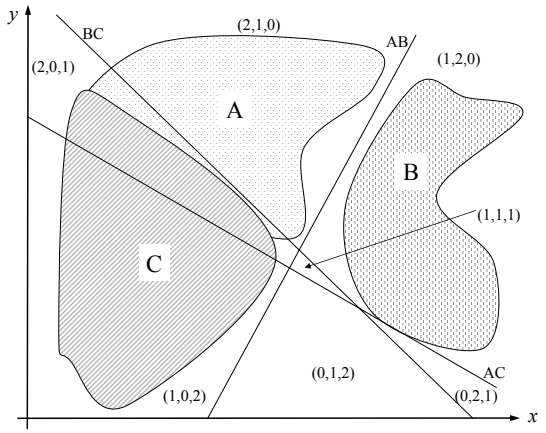


Figure 9.1: Three binary SVMs distinguish the classes A, B and C.

partition of the feature space. Note that those partitions might not be continuous, but are placed between a certain set of classes. Since the partitions are made to separate the objects with respect to their class, it is very likely that the majority of objects belonging to a partition belong to the same class combination. Figure 9.1 display an example of three classes that are separated by three binary SVMs. For each partition the corresponding voting vector is given. Note that voting vector (2,0,1) describes a partition where the majority of objects belongs to both classes A and C.

The reason why the classifier F_2 still offers better efficiency than using just one function $F : O \rightarrow C^*$ is that usually the cardinality of the output space C and therefore that of V is much smaller than the number of features describing an object o. Though F_2 might employ many additional SVMs, in most cases the method offers a performance benefit due to the much simpler input space V.

9.3.2 Multi-Represented Classification using SVMs

An important aspect of the proposed system is that the classification of biological objects should be based on all available information. Thus, the

classifier should be able to use as much different representations as possible or if the object is described in more than one form, it should use all available representations. For proteins, common representations are describing text, sequence data, secondary and three dimensional foldings. Our prototype uses text and sequence data but the introduced method for multi-represented objects is capable to handle any number of representations.

To extract features from each representation, there are several standard techniques for each kind of representation (see Section 9.2). Thus, the first step is to extract features from the objects in the training set for each representation. As mentioned before, a simple solution is given by building up one feature space incorporating the features drawn from all the representations. However, for the following reasons, a more sophisticated approach offers better results. The number of features best suitable for each representation yields an unbalanced weighting of the impact of each representation. For example, the number of features used for a suitable text representation might be orders of magnitude higher than those used for three dimensional foldings. Thus, most classifiers will favor the representation providing more features instead of the representation carrying more information. Furthermore, the techniques proposed to handle different representations vary in the parametrization of the classifiers. For example, the SVM for text and sequence data may use different kernel functions to distinguish the objects. By using a combined feature space, we are forced to find a compromise for these tuning decisions that might not offer optimal results. Last but not least, the handling of missing representations of a data object is difficult, since the classifier expects at least some values in the missing dimensions of the input space.

As a consequence our classification system considers varying representations separately. The idea is that each data source is handled by some specialized classifier first. Afterwards the results are combined to build up a prediction for the object.

Thus, our classifier has the following form: Let $O = R_1 \times .. \times R_n$ be the

set of objects $o = (r_1, \ldots, r_n)$ represented by an n-tuple of feature vectors $r_1, \ldots, r_n$ drawn from the single representations $R_1, \ldots, R_n$. Furthermore, let C be the set of classes, let C^* be the extended class set including all valid class combinations and let V with $dim(V) = |C|$ be the vector space of voting vectors. Then our classifier has the following form:

$$Cl : O \rightarrow C^*$$
$$Cl((r_1, .., r_n)) = F_2(comb(F_{1,1}(r_1)), .., F_{1,n}(r_n))$$

where $F_{1,j} : R_j \rightarrow V$, $F_2 : V \rightarrow C^*$ and $comb : 2^V \rightarrow V$.

Each of the feature vectors r_j is classified by a specialized classifier $F_{1,j}$ into a voting vector. The function *comb* combines the voting vectors of each available representation into one common voting vector which is afterwards mapped into the expanded class space C^* by F_2.

Due to this design each representation can be classified in the best possible way by a specially tuned SVM and the resulting voting vectors are combined without any influence of the dimensionalities of the feature spaces. Last but not least, missing representations can be handled by a properly designed combination function. Since the combination function is designed to handle an input of j voting vectors with $1 \leq j \leq n$ and generates an output vector that is independent from j, missing representations are processable. Note that though the missing representations can be processed, the quality of the prediction is still likely to suffer, depending on the significance of the remaining descriptions.

Our general combination function has the following form:
$comb : 2^V \rightarrow V$, where

$$comb(o) = \begin{pmatrix} f_1(r_{1,1}, \ldots, r_{1,m}) \\ \vdots \\ f_N(r_{N,1}, \ldots, r_{N,m}) \end{pmatrix}$$

and V is the feature space of voting vectors for N base classes. f is a normalized function to combine the components of the m input vectors,

where $1 \leq m \leq n$ and n is the number of representations. Common choices for f are the minimum, the product, the sum and the maximum, where the sum and the product have to be normalized by m. [Dui02] offers a survey which of those four strategies is suited best for which kind of object. Furthermore, [Dui02] introduces the idea of employing an additional learner to improve predictions. This idea is maintained by our second classifier as long as it does not collide with the requirement of handling objects with missing representations. As a result, we lose the possibility to consider correlations between votes for different classes drawn from different representations.

Since the results achieved by employing the methods described in [Dui02] were not capable to improve accuracy, we introduce a weighted strategy to achieve much better results. The main problem of the basic strategies is that each data source always has the same impact on the result without considering the reliability of the class prediction in each representation. For example, consider a two class classification task based on two representations. If both local voting vectors indicate different classes, an unweighted combination rule cannot predict any class. For this case, it would make sense to favor the voting vector offering more reliable information. Thus, we should increase the impact of each voting vector dependent on its reliability.

To model the influence of different data sources, we introduce weight factors for each representation j and each object o. These weight factors reflect the following aspect: How confident is a specialized classifier $F_{1,j}$ about the voting vector it produced for a special feature vector r_j. Our rule for calculating the components of the general voting vector is:

$$f_i(r_{i,1}, \ldots, r_{i,m}) = \frac{\sum_{j=1}^{m} w_{r_j} \cdot (F_{1,j}(r_j)_i)}{m}$$

where w_{r_j} is a weight describing the confidence of the prediction derived from $F_{1,j}$ for r_j and $F_{1,j}(r_j)_i$ is the i-th component of the voting vector derived from the j-th data source. Note that we choose the sum-function as base combination strategy, since all data sources should contribute to the result. Let us note that using the confidence vectors derived by a statistical classifiers

as proposed [Dui02] also weights the impact of representation. However, since these confidence vectors are the foundation of the class decision itself, they are often too unreliable to judge the class decision. In other words, if a statistical classifier predicts the wrong class, it often judges the reliability of the prediction still as very high because both aspects are judged considering the same model.

Our method to find a meaningful weighting uses an established method for deriving confidence values for binary SVMs. This method calculates the distance of the feature vector to the separating hyperplane. The idea is that the closer the feature vector is to the separating hyperplane the less confident is the prediction. This is based on the characteristic of SVMs that objects which are difficult to decide are placed in the surrounding of the hyperplane. To derive confidence values and to model the effect that after a certain distance to the separating hyperplane the decision is considered as secure, a sigmoid function is usually applied to the distance. Furthermore, the closer surrounding of the hyperplane is treated in a more sensitive way. Thus, the confidence $conf$ of a SVM svm is given by:

$$conf_{svm}(o) = \frac{1}{1 - e^{\alpha \cdot svmdist(o)}}$$

for object o, $svmdist(o)$ the distance of o to the separating hyperplane of svm and α a parameter for regulating the sensitivity.

Since our system employs multi-class SVMs that usually consist of more than one binary SVM, the process of deriving a proper weight has to consider several distances. Therefore, we determine the class having the maximum vote in the voting vector derived from one data source. For this class, we determine the minimum confidence value belonging to the SVMs that characterize the predicted class (cf. Figure 9.2).

Let $F_{1,j}$ be the multi-class SVM treating the representation j. Then $F_{1,j}$ is built from the following matrix of binary SVMs:

$$F_{1,j} = \begin{pmatrix} - & svm_{1,2} & \dots & svm_{1,N} \\ svm_{2,1} & - & svm_{2,3} & \dots \\ \dots & \dots & \ddots & \dots \\ svm_{N,1} & \dots & svm_{N,N-1} & - \end{pmatrix}$$

Note that this matrix of SVMs is symmetric, since the classifier distinguishing i from j is the same as the one distinguishing j from i. Then we determine the weight in the following way:

$$w_{r_j} = \min_{svm_{i,maxdim(v_j)} \in F_{1,j}} conf_{svm_{i,maxdim(v_j)}}(r_j)$$

where v_j is the voting vector derived by $F_{1,j}$ for r_j and $maxdim(v_j)$ is the class in v_j having the maximum number of votes.

The idea is that the class having the maximum count is most likely part of the prediction. If the feature vector is predicted with a high confidence value, it needs to have a sufficient distance from any of the other classes. Afterwards the weights are normalized and used in *comb* as described above. Thus, classifiers offering highly reliable results have significantly more impact on the resulting voting vector. Since the weights are calculated for every single instance to be classified, our combination function adjusts to the current object and does not prejudge complete representations. Thus, each object is predicted on the representation that is most significant for the current task. Therefore, we call this new method *object-adjusted weighting*. Let us note, that object-adjusted weighting in the introduced form implies that all weights are derived using the same method.

9.3.3 Structuring the Output Space

Classification into large class sets providing over 100 classes is a very time consuming task. Remember that a one-versus-one multi-class SVM needs

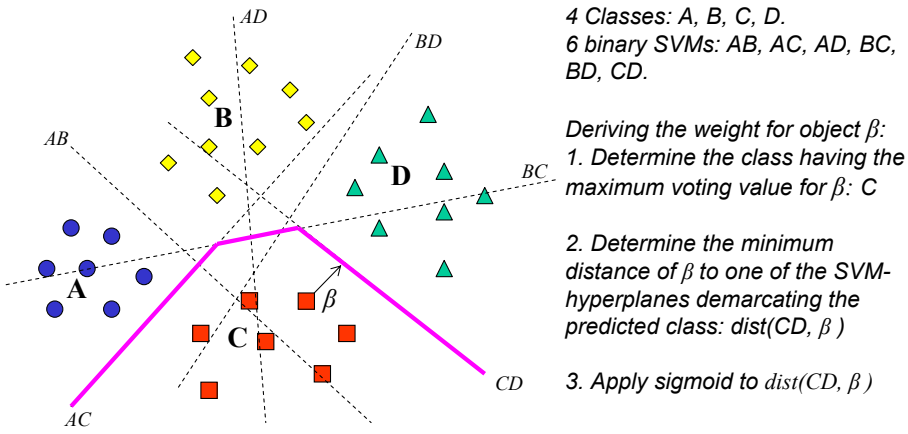


Figure 9.2: Illustration of the class confidence estimation (see text for details).

4,950 binary SVMs for 100 classes. Thus, to make the system scalable, it is necessary to find an efficient way to classify into large class sets. One way to speed up classification is to employ additional knowledge about the class set. Considering a class system like an ontology or a taxonomy and not just a simple set of classes, opens the possibility to split the large classification problem into several smaller ones which are faster to process. Let us note that the accuracy achieved on smaller systems also tends to be significantly higher because of the smaller problem.

Ontologies are a common approach to model class information in molecular biology. Though an ontology usually models all kinds of relations, most of them are not useable for classification in our system. The problem is that the objects we want to classify do not have any link to any other object yet. Thus, exploiting general relations to determine the class of an object is very difficult in our application. On the other hand, we can use the inheritance relations of the ontology because of our knowledge that an object which is part of a supertype opens up the possibility that it is part of a subtype, too. Thus, we use the taxonomy part of the given ontology. This taxonomy varies

from the majority of class hierarchies used in other projects, regarding the following three aspects:

- Instances can be placed at varying abstraction levels. It is common to biological ontologies to collect entities not further specified in non-leaf nodes of the ontology though there might be several refinements of the class.
- It is possible that database entries may link to varying classes in the class system. Thus, we have to treat multi-classified objects belonging to one or more classes.
- A class hierarchy of an ontology might use multiple inheritance for some of its classes. This characteristic leaves us without a taxonomy tree and demands a taxonomic graph.

According to these characteristics, we restrict a given ontology to a *taxonomic directed acyclic graph*. A *directed acyclic graph* (DAG) is a connected, directed graph that does not contain any cycles. An entry node to a DAG is a node without any incoming edge. If there is only one entry point, the node is called root and we have a rooted DAG. A *taxonomic directed acyclic graph* (TDAG) is a rooted DAG where each node is connected to a class of objects. The class of a predecessor node is a supertype to the classes of its successor nodes. Furthermore, we require that the entries belonging to the supertype are exactly the union of the entries belonging to its subtypes. Though this requirement is not fulfilled in the first place, we can easily fix it by introducing additional leaf nodes to the supertypes having instances that do not belong to any of the subtypes. Thus, we get a TDAG which is our choice of class system, providing a more general setting. A sample TDAG is depicted in Figure 9.3.

To find out which method of hierarchical classification is best suited for exploiting TDAGs, we will discuss two basic approaches and their ability to support our setting.

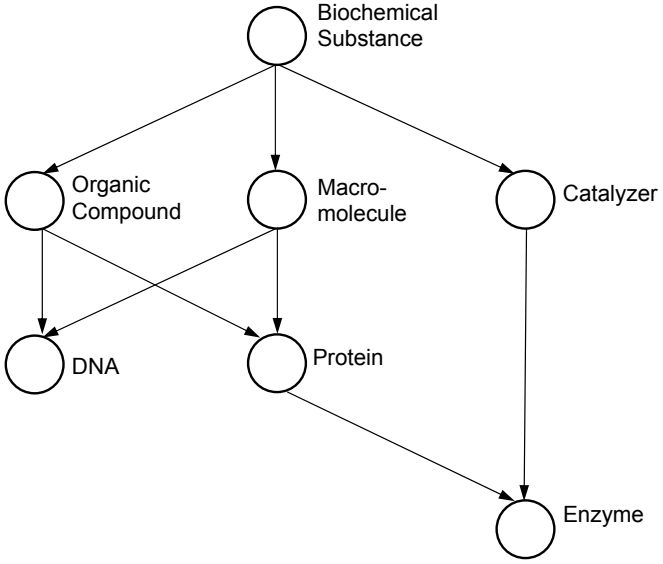


Figure 9.3: A sample TDAG.

The basic approach of hierarchical classification is to decompose a flat N class problem to several smaller problems of the size $n_i \ll N$. Thus, common hierarchical classifiers are class hierarchies where each supertype provides a classifier that predicts the subtypes a given object belongs to. The idea is that these smaller problems are easier and faster to decide than one big problem. The differences between the majority of introduced methods for hierarchical classification are mostly within the part of the class system that is traversed during classification. Principally, there are two strategies to tackle the problem:

- The probabilities (or a combination of classifier outputs) are considered for each leaf in the class hierarchy. Thus, the whole class hierarchy is visited and leaves getting smaller confidence values by the top-level classifiers might still be considered if the classifiers are confident on the rest of their decision paths.

- Step by step at each level, the sub-classes that are considered unlikely

are pruned. Thus, only a small portion of the classifiers in the system is employed for classification.

The first approach tries to achieve the best possible accuracy while the second approach offers better efficiency, but might lose accuracy due to its restrictiveness. Thus, the second approach is favorable for our target to employ large TDAGs providing over 100 classes if accuracy does not suffer too much. Further reasons for employing the second approach to achieve classification into large TDAGs are:

- The occurrence of multiple inheritance and leaves on varying abstraction levels makes it computationally demanding to calculate comparable probabilities for all leaves. To achieve such a calculation implies knowledge about all pathes leading to a leaf. Furthermore, the fact that leaves are placed at different abstraction levels requires proper normalization of the probabilities.

- Employing classifiers that do not consider the possibility that the object belongs to none of its classes, might generate confidence values that do not reflect a realistic estimation. Figure 9.4 shows an example of a hierarchical classifier based on SVMs employing the distance to the hyperplane as confidence value. In the described case, an object is misclassified due to an unrealistically high second level confidence value.

- The possibility of multiple paths leading to a class is able to compensate a wrong decision in the second approach. If one path to reach a class is pruned, it still might be reachable via another path in the TDAG.

Thus, we choose the second approach for building a classifier that employs TDAGs as a class system. Our System now consists of a TDAG organizing the classes we want to predict. At each node a classifier designed as described in the previous subsection is trained to decide the correct subtypes under the

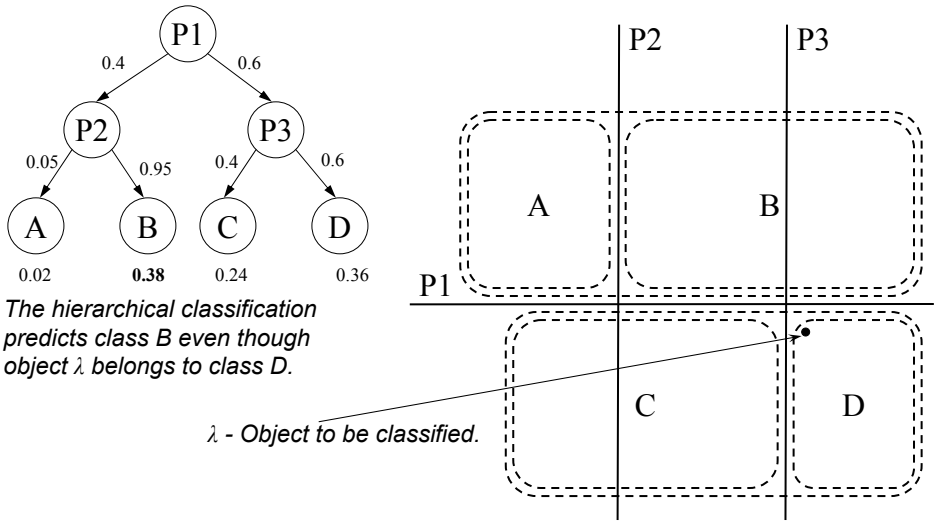


Figure 9.4: Example for a wrong decision due to a very high 2nd level confidence value.

precondition that the object already belongs to the class the node is attached to. Hierarchical classification is now achieved by starting the traversal of the TDAG at the root node and following all predicted paths until every branch of the process reaches a leaf. The set of reached leaf nodes is the prediction of the class set made by the system.

9.4 Experimental Evaluation

9.4.1 Testbed

In order to demonstrate the advantages of our system, we carried out a versatile experimental evaluation. The experiments were performed on five different classification problems. The testbeds consist of 17 to 107 Gene Ontology classes [Con00] and their “is-a“ relationships. The corresponding

	Set 1	**Set 2**	**Set 3**	**Set 4**	**Set 5**
Name	Response to external stimulus	Protein binding activity	Receptor binding activity	Oxidoreductase	Biosynthesis
Number of Goal Classes	17	19	26	94	107
References to proteins	1,832	1,166	1,857	9,907	18,111
Multi-class Proteins (%)	5.36	13.63	14.29	17.97	20.58

Table 9.1: Details of the test environments

objects were taken from the SWISS-PROT [BBA+03] protein database and consist of a describing text and the amino acid sequence of the described protein. The properties of each testbed is shown in Table 9.1. In order to obtain a TDAG with sufficient training objects per class, the original environment was pruned. The result of the pruning is a TDAG that fulfills the following conditions:

1. Every leaf class refers to at least *MinSupport* proteins.

2. Every inner node in the TDAG has at least *MinSonNumber* direct son classes.

3. The pruning process contains as much training objects as possible. This condition is fulfilled by moving proteins from pruned classes to their direct parent.

The details of the classification problems are listed in Table 9.1.

All algorithms are implemented in Java and were tested on a work station that is equipped with a 1.4 GHZ Pentium IV processor and 2 GB main memory. To measure the accuracy for multi-classified objects, we used the following definition of classification accuracy:

$$Accuracy = 1 - \frac{\sum_{o \in T} (|(A(o) \cup B(o)) - (A(o) \cap B(o))|)}{\sum_{o \in T} |A(o)| + |B(o)|}$$

where o is a test object, T is the set of test objects, $A(o)$ is the correct class set for object o and $B(o)$ is the predicted class set of object o. In order to avoid overfitting, the evaluation used 10-fold cross-validation.

To classify protein sequences, we employed the approach described in [DK02]. The basic idea is to use local (20 amino acids) and global (6 exchange groups) characteristics of a protein sequence. To construct a meaningful feature space, we formed all possible 2-grams for each kind of characteristic which provided us the 436 dimensions of our sequence feature space. For text descriptions, we employed a TFIDF vector for each description that was built of 100 extracted terms. Both representations were classified, employing a degree 2 polynomial kernel. Due to the superior results of the described hierarchical approach, all of the following experiments use a structured output space with the exception of the flat classifier approach. The feature selections were applied to each node separately as described in [KS97].

9.4.2 Experimental Results

To show that the one-versus-rest approach is not suitable for our application, we compared its accuracy on the text descriptions to the one-versus-one approach. Since it offered significantly inferior results to the settings employing an extended class set and the one-versus-one approach (4.49% - 12.01% less accuracy), we did not follow this approach any further (cf. Figure 9.5). For example, the classification accuracy achieved for the Set 4 testbed by the one-versus-one strategy was 82.12%, whereas the one-versus-rest approach only reached 70.11%.

Our second experiment demonstrates that a two-step classifier offers better results compared to a single classifier using a direct extension of the class set (cf. Section 9.3). The two-step approach achieved comparable accuracy and superior efficiency for all test sets (cf. Figure 9.6). In particular, our approach showed for Set 5 with 107 goal classes the classification accuracy

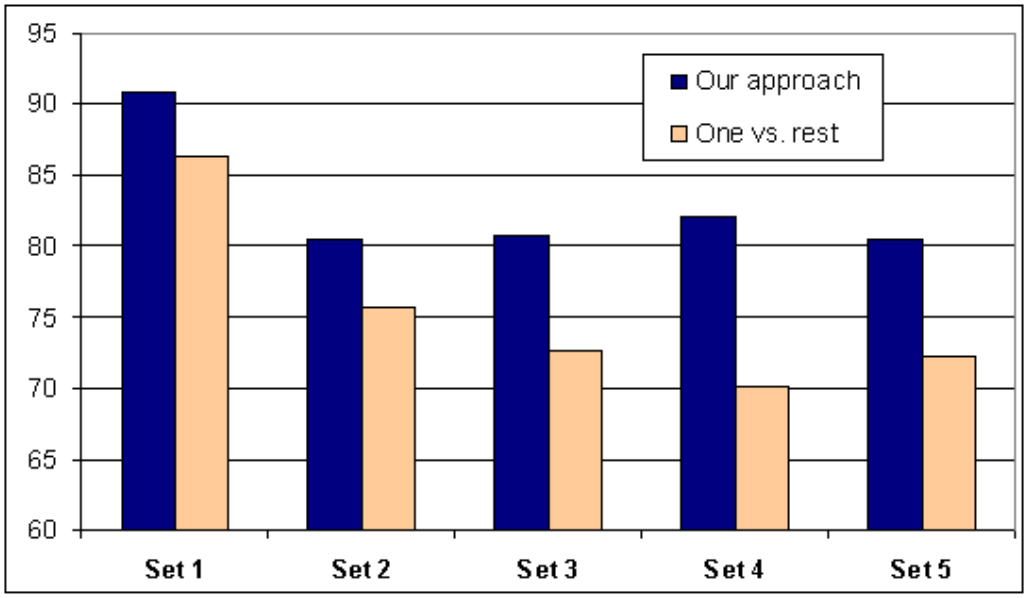


Figure 9.5: Classification accuracy (in %) of our method compared to the one-versus-rest approach.

of 81.37% and took on average 1.75 seconds as classification time per object. The competing method using only one classifier and a direct extension of the class set achieved ca. 1 % less classification accuracy and was evidently slower - 2.66 seconds as average classification time per object. According to our results, the two-step approach improved both efficiency and effectiveness of the classifier.

In order to show the advantages of the hierarchical approach against an unstructured class system, we compared both approaches for the introduced classifier on both representations. We observed better accuracy in most cases and an enormous improvement in classification time, especially when working with large class systems (cf. Figure 9.6). In case of Set 4 providing 94 target classes the flat-classifier achieved 69.92% accuracy and took on average 5.97 seconds for the classification of an object. The hierarchical approach achieved on the same data significantly higher accuracy (82.65%) and needed 0.85 seconds per object. Thus, hierarchical classification was processed up to 7

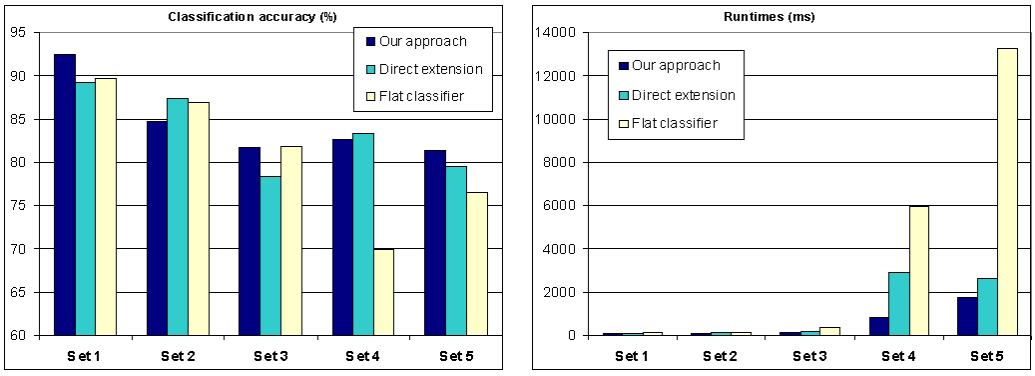


Figure 9.6: Accuracy and runtime for hierarchical classification employing a one-versus-one SVM with an extended class set (direct extension) and two subsequent one-versus-one SVMs (our approach). Additionally, we compare our approach without using a TDAG (flat classifier).

times faster than flat classification. Note that this considerable speed up was achieved especially in the large TDAGs where the performance is much more critical than in smaller problems. Furthermore, the classification accuracy surpassed the accuracy observed for the other approaches in the majority of test sets.

The next experiment compares the use of a compound input space for classification. Therefore, we compared the accuracy achieved by employing only the text part, only the sequence part, a combined feature space that incorporates the features of both representations and our combined classifier. The combined classifier was evaluated with and without object-adjusted weights (cf. Table 9.2). In all of our test environments, the classification of text was more accurate than that of sequence data based on the employed 436 dimensional feature space. Furthermore, the combination without object-adjusted weights and the variant employing a combined feature space were not capable to improve accuracy towards the text description in all cases. Thus, it would be more promising to restrict the classifier to employ text descriptions only. On the other hand, the variant that employs the object-adjusted weighting

Method	Set 1	Set 2	Set 3	Set 4	Set 5
text only	90.82	80.5	80.71	82.12	80.55
sequence only	89.4	80.3	77.96	75.22	71.09
combined feature space	88.6	80.56	74.76	77.87	77.89
combination with average	87.92	78.61	72.97	76.68	75.35
object-adjusted weighting	92.52	84.71	81.65	82.65	81.37
training on text and sequ. classif. on sequ. only	89.32	80.66	76.44	69.56	73.41

Table 9.2: Classification Accuracy (in %) for text descriptions, sequence data and varying combination methods.

increases the accuracy in all 5 testbeds up to 4%. Thus, it was the only examined method that was able to dynamically decide which representation is suited best and draw advantages from all representations.

Our last experiment examines the capability of the system to cope with incomplete data objects. Therefore, we trained the classifiers on both data sources and tested them by only classifying the sequence part of the test instances. For the majority of testbeds it turned out that the accuracy approximately reached the level achieved by classifying the sequence data alone (see last line of Table 9.2). In the case of Set 5, the classification accuracy of 73.41% even exceeded the values observed for sequences only (71.09%). Thus, the system is able to handle incomplete data. Let us note that this capability gets more and more important with an increasing number of representations, since it is very demanding to train classifiers that can handle the remaining representations with increasing numbers of representations in the best possible way. Furthermore, when incorporating several representations, the remaining representations are more likely to compensate the missing information.

9.5 Conclusions

In this chapter, we proposed a prototype for classifying data objects into taxonomic directed acyclic graphs and applied it to biological entities in molecular biological ontologies. Our method addresses the following problems: First, biological instances often consist of multiple representations such as sequence, text, etc. The classification process within our prototype is able to integrate all possible representations of an instance and can also handle the frequently occurring case when one or more representations are missing. Second, our prototype handles multi-classified objects, the occurrence of multiple inheritance and leaf nodes on different abstraction levels.

A thorough experimental evaluation of our prototype based on a versatile testbed for classifying proteins from SWISS-PROT into Gene Ontology is presented. Based on this testbed, we demonstrated that our method is capable to classify new entries with high accuracy and an efficiency adequate for real-world applications.

Chapter 10

Conclusions about Multi-Represented Data Mining

This part of the thesis discusses data mining in multi-represented objects. It is concluded by the following chapter which provides an overview of the introduced techniques with respect to multi-represented aspects. Furthermore, we draw general conclusions that are based on the analysis of the introduced solutions.

10.1 Summary of the Introduced Techniques

Before drawing general conclusions, we will briefly review the two solutions for multi-represented classification and clustering which are introduced in this thesis.

In chapter 8 density-based clustering of multi-represented data was introduced. To extend the established algorithm DBSCAN to this type of input data, we redefined the core-object property by the union and the intersection method. The idea of the intersection method is that in order to be a core object a data object should be placed in a dense region in all representations. Additionally, there have to at least k objects within in the local ε-neighborhood in all representations. Thus, it is well suited for applications in which the proximity of two object representations is necessary but not sufficient to indicate the proximity of the original objects. On the other hand, the union method defines a core object based on the presumption that it is enough that an object is placed in a dense region with respect to all representations. Thus, there have to be at least k objects in the union of all local ε-neighborhoods. The experimental evaluation demonstrated that our solution is capable to derive more meaningful clusterings compared to several other clustering methods.

Chapter 9 describes a solution for ontology based data integration for biological databases. To automatically map proteins to there ontology classes, we use multi-represented classification of text annotations and amino acid sequences. Our approach to multi-represented classification builds a multi-class support vector machine (SVM) for each representation. Afterwards a voting vector is derived from each multi-class SVM. To combine the voting vectors, we build a weighted average vector. The key to success is the so-called object-adjusted weighting which weights each representation depending on the reliability of the local class predictions. Therefore, the voting vector of a SVM predicting the class of a given object with more confidence provides more influence to the final class prediction. Our experiments indicate that

combining classification results without object-adjusted weighting often provides less accuracy than classification with respect to only one representation. However, by using object-adjusted weighting it is possible to increase the accuracy compared to separated classification in each of the representations.

10.2 Conclusions about Multi-Represented Data Mining

When using multiple representations for data mining there are two main problems that have to be solved in order to draw maximum benefits from the additional information.

10.2.1 Comparability of Local Results and Data Objects

Considering different aspects of the same data objects allows us to use widely varying object representations like vectors, graphs and sequences. To combine these representations for deriving a global pattern, the meaning of each representation has to be made comparable. To achieve comparability there exist several approaches:

- **Joined Data Spaces**
  In this approach, the features of underlying representations are joined into one feature space. This technique is quite common for a set of vector representations. However, by simply joining the data spaces into a single high dimensional vector space, we loose the information that different features are derived from different representations. Thus, we loose the ability to treat the same type of feature in a specialized way. For example, joining text and spatial features is technically easy. However, it is difficult to find a standard distance function that treats both types of features in a well-balanced way. On the other hand, there

are some data mining algorithms that are suitable for analyzing joined data spaces. For example, decision tree classifiers treat each available feature separately and thus do not suffer from these negative effects.

- **Combined Distance Functions**
  Another approach to achieve comparability is to use the local specialized distance functions in each representation and combine the local distance values to a global distance. The advantage of this approach is that we can use established techniques for each type of representation. However, a distance measure allows us only to use distance based data mining algorithms. Furthermore, in order to find a well-balanced influence of each representation, it is necessary to find suitable methods for normalization. Unfortunately, this is rather difficult in many cases. There are two ways to normalize distance values. The first is to use the theoretical maximum distance in a representation for normalization. The drawback of this approach is that for some distance measures like edit distance, there is no theoretical upper limit. Furthermore, the maximum distance in a data space is not necessarily a good upper bound for normalization. If all distances that actually appear in a given data set are much smaller, the comparability to other data spaces cannot be guaranteed. Therefore, another approach is to calculate the maximum and minimum distance for each representations and use these to normalize the distances. Though this approach avoids the problems of the former approach, it yields other drawbacks. Calculating the maximum and minimum distance in each representation has a quadratic time complexity with respect to the database size. Thus, this approach to normalization is rather inefficient. Furthermore, incremental algorithms might not be applicable because deletions an insertions into the database might change the normalization function.

- **Recombination of Local Patterns**
  The last approach employs data mining algorithms or parts of them

to derive local patterns and recombines them to a global result. Since calculating the distance between two objects can be considered as deriving a local pattern as well, this approach can be considered to be a generalization of the former approach. Examples for usable local patterns are the ε-neighborhood of an object like in our density based clustering method or a voting vector like in the introduced approach to multi-instance classification. Other meaningful patterns are predicates like the core object property, the class membership of an object and the confidence vector of a distribution based classifier. The advantage of this approach is that local patterns provide a higher level of abstraction and are rather independent from the data distribution in the single representations. Thus, comparability is unproblematic in these methods.

10.2.2 Semantics of the Representations

Besides the different techniques to store and compare the objects in the used representations, another problem is the meaning of a representation. Some representation might contain less reliable information than others. Another important question is the relationship of the given application to each of the representations. The following example illustrates the importance of the semantics of representations. Consider the case that there are two object representations and we want to compare these representations by the predicate "is similar" and "is dissimilar". In this simply case, there are two basic interpretations for the meaning of the local results. First, two data objects are similar if there exists at least one representation that states that they are similar. The second is, in order to be similar two objects have to be similar with respect to all of the given representation. Both methods allow valid interpretations of the data set but will provide strongly varying results. Let us note that this example is a simplification of the ideas behind the union and the intersection method. To conclude, integrating the correct semantics

is often mandatory to derive useful patterns.

In many applications, the meaning of each representation might be unknown or difficult to describe in advance. In these cases, finding the correct semantics should be estimated by the data mining algorithm. For classification the problem is easier to handle than for clustering. Since the combination can be optimized with respect to the correct classification of the training objects, the semantics can be discovered automatically. The introduced technique of object-adjusted weighting for SVMs derives the semantics of each representation by estimating the reliability of the local class decision. Therefore, the semantics varies for each object depending on the local characteristics of the underlying class models.

For clustering, finding an automatic way to discover the semantics of individual representations and their relationships is rather difficult. Since there is no information available which of the possible interpretations might provide the best clustering, it is hard to decide the semantics offering the best results. Therefore, we employ additional domain knowledge in our introduced method for multi-represented density-based clustering by selecting the union or the intersection method to handle one of two basic semantics. For the case, that similarity with respect to one representation is enough to indicate object similarity, the union method is selected. For the case, that similarity with respect to all available representations is necessary to indicate object similarity, the intersection method is chosen.

Part IV

Conclusions

Chapter 11

Summary and Future Work

The area of KDD deals with analyzing large data collections to extract interesting, potentially useful, so far unknown and statistical correct patterns. Data mining is the most important step within the process of KDD. Often the data objects to be analyzed are of complex nature. Thus, they are not represented in the best possible way by the common approach using feature vectors. Therefore, data mining algorithms have to handle more complex input representations. This thesis contributes to the development of clustering and classification algorithm that employ more complex input representations to achieve enhanced results. This chapter concludes the thesis by summarizing the introduced methods and presents several directions for future work.

11.1 Summary of Contributions

Recent technological advances have tremendously increased the amount of collected data. Besides the sheer amount of collected information, the complexity of data objects increases as well. To analyze these data collections, new data mining methods are needed that are capable to draw maximum advantage out of the richer object representations. This thesis contributes to the field of data mining of complex objects by introducing methods for clustering and classification of compound objects. In particular, it provides solutions for important data mining problems that employ multi-instance and multi-represented data objects as input for the introduced data mining methods. In the following, we give a summary of these contributions.

Preliminary

The first part of this thesis describes the area of KDD, the step of data mining and general data mining tasks. Furthermore, it contains a motivation why data mining using compound objects is a promising approach to cope with the increasing complexity of real-world data objects. The second chapter surveys important foundations of KDD and provides an introduction to the tasks of clustering and classification.

Data Mining in Multi-Instance Objects

Part II of the thesis deals with the data mining in multi-instance objects. Chapter 3 introduces the ideas of multi-represented objects and names several application areas for which the data objects can be naturally modelled as multi-instance objects.

Chapter 4 presents a solution for data mining in CAD databases that is based on multi-instance objects. Many established data mining algorithms are applicable for any kind of data objects as long as there is a distance measure describing the intuitive similarity of data objects. Therefore, the

chapter introduces a similarity search system that supports effective and efficient similarity queries which are the foundation of distance based data mining algorithms. In this system, a CAD part is described as a set of covers, and the distance between two parts is calculated using the so-called "minimal matching distance". To speed up similarity queries, a powerful filter step is introduced that is employed in multi-step query processing. An extensive evaluation is based on two real-world CAD data collections and demonstrates that the new similarity search system based on multi-instance objects offers a more intuitive notion of similarity. Therefore, it is more suitable for distance based data mining than the compared approaches based on feature vectors.

The next chapter deals with another important application area that can be significantly improved by using multi-instance objects which is called website mining. Website mining is a new direction within web content mining and is concerned with the data mining for websites. A website is a linked set of webpages that is published by the same person, group or organization and usually serves a common purpose. Websites often represent companies and other organizations in the WWW. In order to find these entities the web is searched for websites instead of single webpages. The chapter starts by introducing the idea of website mining and names additional advantages of this new approach.

To find a relevant website in the WWW, it is important to distinguish relevant from irrelevant sites. Thus, after giving some formalizations of the WWW, several methods of classification of websites are discussed. The first method tries to classify websites as homepages, i.e. the page that is found under the domain name of a website. Another simple approach is to condense the word vectors of all webpages into a so-called superpage and classify websites as superpages. Besides this very simple approaches, the chapter introduces two more sophisticated directions of website classification. The first uses a preprocessing step, using so-called page classes. A webpage classifier labels each webpage in a website with its most likely page class. Afterwards the website can be condensed into a so-called topic frequency vector that can

be classified by established classification methods. Another representation of websites are so-called website trees. A website tree incorporates the link structure that can be derived by a breadth-first search through a website to generate a labelled tree. To classify a website tree, Markov tree classifiers of varying orders are applied. The last direction of website classification that is described does not employ any page classes, since the effort of providing page classes and additional training pages tends to be very great. The idea is to transform each webpage into a feature vector and represent the website as set of the resulting feature vectors. This representation can be directly classified by using kNN classification based on a distance measure called "sum of minimum distance" (SMD). Though SMD provides a suitable notion of similarity for two websites, the efficiency of kNN classification of websites tends to be insufficient for real-world applications. Therefore, a solution of reducing all training websites of a website class into a so-called centroid set is proposed. After the reduction of "SMD" to the so-called "half-SMD", this centroid set classifier offers faster and more accurate classification than the ordinary kNN classifier using SMD. An important aspect of website classification is the number of webpages that has to be employed for the accurate prediction of the class of a website. Often it is not necessary to employ the complete set of webpages in a site, but only a minor fraction. Thus, a pruning rule is introduced that significantly restricts the area of a website that is used for classification. Using this pruning method in combination with an incremental website classifier provides faster and more accurate classification as it is shown in the evaluation section. In the evaluation, it turns out that the simple approaches achieved the worst classification results. The most accurate prediction was achieved by the approaches using the preprocessing step based on page classes. For the case that no page classes are provided, the centroid set classifier offered the best trade-off between accuracy and classification time.

After discussing website classification in general, a focused website crawler is introduced that efficiently extracts new unknown websites from the WWW

with high accuracy. This crawler is based on a two-level architecture which allows us to control the number of pages to be downloaded from each website. Thus, it is possible to achieve a good trade-off between accurate classification and efficient crawling. The first level is called external crawler and treats the web as a graph of linked websites. The task of the external crawler is to select the websites to be examined next and to invoke internal crawlers. The second level of the crawler consists of so-called internal crawlers. An internal crawler views the webpages of a single given website and performs focused page crawling within that website. Note that the internal crawler classifies a website as set of webpages, since it employs the crawling for page selection only. The experimental evaluation of the crawler demonstrates that reliable website classification requires to visit more than one but less than all pages of a given site. Furthermore, the introduced crawler was compared to a focused webpage crawler that handles the concept of websites in a corresponding step of post-processing. The website crawler achieved significantly higher classification accuracy than this comparison partner. For comparable accuracy, the website crawler needed a considerably smaller rate of visited pages per relevant site.

At the end of this part, chapter 6 sums up the introduced techniques with respect to the multi-instance specific solutions. Furthermore, several conclusion about multi-instance data mining and possible solutions are drawn.

Data Mining in Multi-Represented Objects

Part III of this thesis deals with data mining methods that employ multi-represented object representations to integrate more available knowledge into the KDD process. Chapter 7 motivates the use of multi-represented objects and names reasons for their appearance in real-world applications. Additionally, several important applications for this basic type of compound objects are surveyed.

After this introduction, chapter 8 discusses the problem of density based

clustering of multi-represented objects. To integrate the information provided by multiple representations, we adapted the core object property proposed for DBSCAN. We proposed two different methods for determining the core objects property for multi-represented objects that both rely on the local ε-neighborhoods of each representation. For sparse data, we introduced the union-method that is based on the assumption that an object should be a core object, if k objects are found within the union of its local ε-neighborhoods. Respectively, the intersection-method was introduced for data where each local representation yields rather big and unspecific clusters. The intersection-method requires that at least k objects are within the intersection of all local ε-neighborhoods of a core object. Thus, this method is much more restrictive. In our experimental evaluation, we introduced an entropy based quality measure that compares a given clustering with noise to a reference clustering. Employing this quality measure, we demonstrated that the union method was most suitable to overcome the sparsity of a given protein data set. To demonstrate the ability of the intersection method to increase the cluster quality, a set of images using two different similarity models was clustered.

In chapter 9, a solution to ontology-based data integration for biomolecular databases is proposed. Therefore, we developed a prototype for classifying multi-represented data objects into taxonomic directed acyclic graphs that is capable to provide the following requirements of this important application. Since the number in classes is rather big in biological ontologies, our method employs hierarchical classification to speed up the classification process. Another common characteristic of this type of problem is that many data objects belong to more than one class at the same time. Thus, our introduced method is capable to predict a set of classes a data object does belong to. Last but most important, we introduce the technique of object-adjusted weighting to draw maximum benefit from all object representations that are provided. The idea of object-adjusted weighting is to classify all object representations separately and than recombine the classification re-

sults with respect to the confidences in each representation. In an versatile evaluation, the developed prototype was tested by mapping entries of the SWISS-PROT [BBA+03] protein database to the corresponding classed in Gene Ontology [Con00]. The results demonstrate that our method is capable to classify new entries with high accuracy and an efficiency adequate for real-world applications.

Chapter 10 summarizes the introduced solutions to multi-represented data mining. Furthermore, a categorization of problems in multi-represented data mining is provided and approaches to solve these problems are surveyed.

11.2 Ideas for Future Work

The following section surveys ideas for further research in the area of data mining in compound objects and some of the mentioned application areas.

For the area of data mining in multi-instance objects, the following directions offer interesting opportunities for future work:

- In chapter 4, we introduced a filter step for efficient multi-step queries based on the minimal matching distance. However, there are additional distance measures on multi-instance objects that are suitable for other applications like the "sum of mininal distances " in chapter 5. Thus, an interesting area of research is the extension of well-known methods to speed up similarity queries to various similarity measures for multi-instance objects. To achieve faster similarity queries a possible direction is the development of new search algorithms that are based on established index structures storing the single instances of all objects. Another direction is the development of additional filter steps for multi-instance query processing.

- As mentioned in chapter 4, distance based data mining can be applied to a variety of complex object representations as long as there is a

suitable distance function, modelling a suitable notion of object similarity. However, for data mining in multi-instance objects there exists a variety of distance functions, each providing a different notion of similarity that is suitable for another object representation. For a new application, it is unclear which of these distance measures provides the best possible results. Therefore, an interesting area of research is the development of classification algorithms that are capable of learning a suitable distance function to provide the best possible results. To do so, it is necessary to find a general model for distance measures on set-valued objects. Within this general model each notion of similarity should be expressible. A data mining algorithm for multi-instance objects should be able to tune the general model to optimize its results. This approach is especially suitable for classification because the quality of the results achieved on the training set can be used to adjust the notion of similarity.

For the more particular area of website mining, there are the following additional developments that could further extend the usefulness of the results:

- A website crawler usually retrieves thousands of websites belonging to a special area of interest. However, within the result set of a crawl there are several types of websites that belong to a certain subgroup of relevant sites. Furthermore, a user interested in a certain kind of website has to be enabled to get convenient access to the structure of the result set. A possible solution to this problem is offered by clustering the resulting websites. By finding groups of closely related websites within the result set of a crawler, a user learns about the types of retrieved websites. Furthermore, a user can screen cluster representations and then investigate interesting clusters more closely.

- Another extension of focused website crawling is the use of website crawling to enhance the crawling for specific information. The more

specific a topic is the less likely it is that highly relevant webpages are directly linked with each other. For example, a webpage containing information about a product that was published by a dealer is unlikely to contain a link to a webpage published by another dealer presenting the same product. Thus, to reach a highly specific webpage, many less specific webpages must be crawled that are connected to the relevant topic in a more general sense. In our, example we could search the WWW for the websites of other dealers. Thus, one solution to solve this problem is to use a crawler with a three level architecture. The first two levels are the same as in the focused website crawler and are used to spot websites that are likely to contain the highly specific content. The third and new level of the crawler extends the search by screening the resulting websites for the highly specific content.

For future work on multi-represented objects, we plan to examine the following problems :

- In chapter 8, density-based clustering of multi-represented objects was introduced. However, there exist other directions of clustering (compare chapter 2.2) that are suitable for other kinds of applications. To exploit multiple representation in these applications, researching the use of multi-represented objects in combination with other directions of clustering yields many interesting aspects. Especially, partitioning clustering like k-Means is one of the most established approaches to clustering. Therefore, an interesting task for future work is the development of methods for partitioning clustering that are capable to find meaningful global clusterings for multi-represented objects.

- When classifying multi-represented objects into large class hierarchies like in chapter 9, the use of support vector machines (SVMs) tends to be very inefficient. The number of binary SVMs employed in established methods for multi-class SVMs increases with the square of the class

numbers. Since not all classification scenarios can provide a hierarchy of classes, an other solution for the efficient classification for problems providing very large class sets have to be found. A classification method that is able to cope with those large class sets is kNN classification. A kNN classifier only employs the kNN sphere of the object to be classified, regardless how many classes exist in the given problem. Thus, we plan to develop new classifiers for multi-represented objects that use kNN classification and are well suited for very large class spaces.

Last but not least, we plan to combine the introduced methods for data mining in multi-represented and multi-instance objects into a general approach for data mining in compound objects. For this approach, an object could be constructed arbitrarily of concatenations and sets of other feature representations like graphs, tree and feature vectors. Especially in the area of protein databases many representations in a multi-represented view might be modelled more precisely by an multi-instance object, e.g. the three dimensional structure. Thus, the use of even richer protein descriptions could yield an even better approach to clustering and classification of this kind of data.

List of Figures

List of Tables

References

[ABKS99] M. Ankerst, M.M. Breunig, H.-P. Kriegel, and J. Sander. "OPTICS: Ordering Points to Identify the Clustering Structure". In *Proc. ACM SIGMOD Int. Conf. on Management of Data (SIGMOD'99), Philadelphia, PA, USA*, pages 49–60, 1999.

[AFS93] R. Agrawal, C. Faloutsos, and A. Swami. "Efficient Similarity Search in Sequence Databases". In *Proc. 4th. Int. Conf. on Foundations of Data Organization and Algorithms (FODO'93), Evanston, ILL, USA*, Lecture Notes in Computer Science (LNCS), Springer, pages 730: 69–84, 1993.

[AGGR98] R. Agrawal, J. Gehrke, D. Gunopulos, and P. Raghavan. "Automatic Subspace Clustering of High Dimensional Data for Data Mining Applications". In *Proc. ACM SIGMOD Int. Conf. on Management of Data (SIGMOD'98), Seattle, WA, USA*, pages 95–105, 1998.

[AKKS99] M. Ankerst, G. Kastenmüller, H.-P. Kriegel, and T. Seidl. "3D Shape Histograms for Similarity Search and Classification in Spatial Databases". In *Proc. 6th Int. Symposium on Large Spatial Databases (SSD'99), Hong Kong, China*, Lecture Notes in Computer Science (LNCS), Springer, pages 1651: 207–226, 1999.

[ALSS95] R. Agrawal, K.-I. Lin, H.S. Sawhney, and K. Shim. "Fast Similarity Search in the Presence of Noise, Scaling, and Translation in Time-Series Databases". In *Proc. 21st Int. Conf. on Very Large Data Bases (VLDB'95), Zurich, Switzerland*, pages 490–501, 1995.

[BBA+03] B. Boeckmann, A. Bairoch, R. Apweiler, M.-C. Blatter, A. Estreicher, E. Gasteiger, M.J. Martin, K. Michoud, C. O'Donovan, I. Phan, S. Pilbout, and M. Schneider. "The

SWISS-PROT Protein Knowledgebase and its Supplement TrEMBL in 2003". *Nucleic Acid Research*, 31:365–370, 2003.

[BBJ+00] S. Berchtold, C. Böhm, H.V. Jagadish, H.-P. Kriegel, and J. Sander. "Independent Quantization: An Index Compression Technique for High-Dimensional Data Spaces". In *Proc. Int. Conf. on Data Engineering (ICDE 2000), San Diego, CA, USA*, pages 577–588, 2000.

[BFOS84] L Breiman, J.H. Friedman, R.A. Olshen, and C.J. Stone. *"Classification and Regression Trees"*. Wadsworth, 1984.

[BGG+99a] D. Boley, M.L. Gini, R. Gross, E.-H. Han, K. Hastings, G. Karypis, V. Kumar, B. Mobasher, and J. Moore. "Document Categorization and Query Generation on the World Wide Web Using WebACE". *Artificial Intelligence Review*, 13(5-6):365–391, 1999.

[BGG+99b] D. Boley, M.L. Gini, R. Gross, E.-H. Han, K. Hastings, G. Karypis, V. Kumar, B. Mobasher, and J. Moore. "Partitioning-based clustering for Web document categorization". *Decis. Support Syst.*, 27(3):329–341, 1999.

[BH98] K. Bharat and M.R. Henziger. "Improved Algorithms for Topic Distillation in a Hyperlinked Environment". In *Proc. 21st ACM SIGIR Conf. on Research and Development in Information Retrieval (SIGIR'98), Melbourne, Australia*, pages 104–111, 1998.

[Big96] J.P. Bigus. *"Data mining with neural networks: solving business problems from application development to decision support"*. McGraw-Hill, Inc., 1996.

[BK97] S. Berchtold and H.-P. Kriegel. "S3: Similarity Search in CAD Database Systems". In *Proc. ACM SIGMOD Int. Conf. on Management of Data (SIGMOD'97), Tucson, AZ, USA*, pages 564–567, 1997.

[BK03] F. Bry and P. Kröger. "A Computational Biology Database Digest: Data, Data Analysis, and Data Management". *Distributed and Parallel Databases*, 13:7–42, 2003.

[BKK96] S. Berchtold, D.A. Keim, and H.-P Kriegel. "The X-Tree: An Index Structure for High-Dimensional Data". In *Proc. 22nd Int.*

Conf. on Very Large Data Bases (VLDB'96), Mumbai (Bombay), India, pages 28–39, 1996.

[BKK97] S. Berchtold, D.A. Keim, and H.-P. Kriegel. "Using Extended Feature Objects for Partial Similarity Retrieval". *VLDB Journal*, 6(4):333–348, 1997.

[BKK+03] S. Brecheisen, H.-P. Kriegel, P. Kröger, M. Pfeifle, and M. Schubert. "Using Sets of Feature Vectors for Similarity Search on Voxelized CAD Objects". In *Proc. ACM SIGMOD Int. Conf. on Management of Data (SIGMOD'03), San Diego, CA, USA)*, pages 587–598, 2003.

[BM02] H. Brighton and C. Mellish. "Advances in Instance Selection for Instance-Based Learning Algorithms". *Data Mining and Knowledge Discovery*, 6(2):153–172, 2002.

[BMH92] A. Badel, J.P. Mornon, and S. Hazout. "Searching for Geometric Molecular Shape Complementarity using Bidimensional Surface Profiles.". *Journal of Molecular Graphics*, 10:205–211, 1992.

[Bur98] C.J.C. Burges. "A Tutorial on Support Vector Machines for Pattern Recognition". *Data Mining and Knowledge Discovery*, 2(2):121–167, 1998.

[BWF+00] H.M. Berman, J. Westbrook, Z. Feng, G. Gilliland, T.N. Bhat, H. Weissig, I.N. Shindyalov, and Bourne P.E. "The Protein Data Bank". *Nucleic Acid Research*, 28:235–242, 2000.

[CDF+99] M. Craven, D. DiPasquo, D. Freitag, A.K. McCallum, T.M. Mitchell, K. Nigam, and S. Slattery. "Learning to Construct Knowledge Bases from the World Wide Web". *Art. Int.*, 118(1/2):69–113, 1999.

[CDI98] S. Chakrabarti, B. Dom, and P. Indyk. "Enhanced Hypertext Categorization Using Hyperlinks". In *Proc. ACM SIGMOD Int. Conf. on Management of Data (SIGMOD'98), Seattle, WA, USA*, pages 307–318, 1998.

[Cen] Google Press Center. "Google Achieves Search Milestone With Immediate Access To More Than 6 Billion Items". http://www.google.com/press/pressrel/6billion.html.

[CGMP98] J. Cho, H. Garcia-Molina, and L. Page. "Efficient Crawling Through URL Ordering". In *Proc. 7th Int. World Wide Web Conf. (WWW'98), Brisbane, Australia*, pages 30(1–7):161–172. Computer Networks, 1998.

[Cha03] S. Chakrabarti. *"Mining the Web"*. Morgan Kaufmann, 2003.

[CHY96] M.S. Chen, J. Han, and P.S. Yu. "Data Mining: A Overview from a Database Perspective". *IEEE Trans. on Knowledge and Data Engineering*, 8(6):866–883, 1996.

[Con] Ranks Search Engine Optimising Consultancy. "default english stopwords". http://www.ranks.nl/tools/stopwords.html.

[Con86] M.L. Connolly. "Shape Complementarity at the Hemoglobin a1b1 Subunit Interface". *Biopolymers*, 25:1229–1247, 1986.

[Con00] The Gene Ontology Consortium. "Gene Ontology: Tool for the Unification of Biology". *Nature Genetics*, 25:25–29, 2000.

[CPS02] S. Chakrabarti, K. Punera, and M. Subramanyam. "Accelerated Focused Crawling through Online Relevance Feedback". In *Proc. 11th Int. World Wide Web Conf. (WWW'02), Honolulu, Hawaii, USA*, pages 148–159, 2002.

[CPZ97] P. Ciaccia, M. Patella, and P. Zezula. "M-Tree: An Efficient Access Method for Similarity Search in Metric Spaces". In *Proc. 23rd Int. Conf. of Very Large Data Bases (VLDB'97), Athens, Greece*, pages 426–435, 1997.

[CST00] N. Cristianini and J. Shawe-Taylor. *"An introduction to support vector machines and other kernel-based learning methods"*. Cambridge University Press, 2000.

[CV95] C. Cortes and V. Vapnik. "Support-Vector Networks". *Machine Learning*, 20(3):273–297, 1995.

[CvdBD99a] S. Chakrabarti, M. van den Berg, and B. Dom. "Distributed Hypertext Resource Discovery Through Examples". In *Proc. 25th Int. Conf. on Very large Databases (VLDB'99), Edinburgh, Scotland, UK*, pages 375–386, 1999.

[CvdBD99b] S. Chakrabarti, M. van den Berg, and B. Dom. "Focused Crawling: a new Approach to Topic-Specific Web Resource Discovery". In *Proc. 9th Int. World Wide Web Conf. (WWW'99),*

Toronto, Canada, pages 31(11–16): 1623–1640. Computer Networks, 1999.

[CZ00] Y. Chevaleyre and J-D. Zucker. "Noise-Tolerant Rule Induction from Multiple-Instance Data". In *Proc. 7th Int. Conf. on Machine Learning Workshop on Attribute-Value and Relational Learning, Stanford, CA, USA*, 2000.

[DC00] S. Dumais and H. Chen. "Hierarchical Classification of Web Content". In *Proc. 23rd ACM SIGIR Conf. on Research and Development in Information Retrieval (SIGIR'00),Athens, Greece*, pages 256–263, 2000.

[DCL+00] M. Diligenti, F. Coetzee, S. Lawrence, C.L. Giles, and M. Gori. "Focused Crawling Using Context Graphs". In *Proc. 26th Int. Conf. on Very Large Data Bases (VLDB'00), Cairo, Egypt*, pages 527–534, 2000.

[DEKM98] R. Durbin, S. Eddy, A. Krogh, and G. Mitchinson. *"Biological Sequence Analysis"*. Cambridge University Press, 1998.

[DK02] M. Deshpande and G. Karypis. "Evaluation of Techniques for Classifying Biological Sequences". In *Proc. 6th Pacific-Asia Conf. on Knowledge Discovery and Data Mining (PAKDD'02),Taipei, Taiwan*, pages 417–431, 2002.

[DL01] S. Dzeroski and N. Lavrac. *"Relational Data Mining"*. Springer, 2001.

[DLLP97] T.G. Dietterich, R.H. Lathrop, and T. Lozano-Perez. "Solving the multiple instance problem with axis-parallel rectangles". *Artificial Intelligence*, 89:31–71, 1997.

[DLR77] A.P. Dempster, N.M. Laird, and D.B. Rubin. "Maximum Likelihood from Incomplete Data via the EM Algorithm". *Journal of Royal Statistical Society, series B*, 1(39):1–31, 1977.

[DMO] DMOZ. "open directory project". http://dmoz.org.

[DMS98] S. D'Allesion, K. Murray, and R. Schiaffino. "The Effect of Hierarchical Classifiers in Text Categorization". In *Proc. 3rd Conf. on Empirical Methods in Natural Language Processing (EMNLP-3), Montreal, Quebec, Canada*, 1998.

[Dui02] R. Duin. "The Combining Classifier: To Train Or Not To Train?". In *Proc. 16th Int. Conf. on Pattern Recognition (ICPR'02), Quebec City, Canada)*, pages 765–770, 2002.

[Dun03] M.H. Dunham. *"Data Mining Introductory and Advanced Topics"*. Prentice Hall, 2003.

[EKS02] M. Ester, H.-P. Kriegel, and M. Schubert. "Website Mining: A new way to spot Competitors, Customers and Suppliers in the World Wide Web". In *Proc. 8th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining (SIGKDD'02), Edmonton, Canada*, pages 249–258, 2002.

[EKS04] M. Ester, H.-P. Kriegel, and M. Schubert. "Accurate and Efficient Crawling for Relevant Websites". In *Proc. 30th Int. Conf. on Very Large Data Bases (VLDB'04), Toronto, Cananda*, pages 396–407, 2004.

[EKSX96] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu. "A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise". In *Proc. 2nd Int. Conf. on Knowledge Discovery and Data Mining (KDD'96), Portland, OR*, pages 291–316, 1996.

[EM97] T. Eiter and H. Mannila. "Distance Measures for Point Sets and Their Computation". *Acta Informatica*, 34(2):103–133, 1997.

[FBF+94] C. Faloutsos, R. Barber, M. Flickner, J. Hafner, et al. "Efficient and Effective Querying by Image Content". *Journal of Intelligent Information Systems*, 3:231–262, 1994.

[Fis87] D. H. Fisher. "Conceptual clustering, learning from examples, and inference". In *Proc. 4th Int. Workshop on Machine Learning, Irvine, CA*, pages 38–50, 1987.

[FPSS96] U.M. Fayyad, G. Piatetsky-Shapiro, and P. Smyth. "Knowledge Discovery and Data Mining: Towards a Unifying Framework". In *Knowledge Discovery and Data Mining*, pages 82–88, 1996.

[FRM94] C. Faloutsos, M. Ranganathan, and Y. Manolopoulos. "Fast Subsequence Matching in Time-Series Databases". In *Proc. ACM SIGMOD Int. Conf. on Management of Data (SIGMOD'94), Minneapolis, MN, USA*, pages 419–429, 1994.

[GFKS02] T. Gärtner, P.A. Flach, A. Kowalczyk, and A. Smola. "Multi-Instance Kernels". In *Proc. 19th Int. Conf. on Machine Learning (ICML'02), Sydney, Australia*, pages 179–186, 2002.

[Goo] Google. "web search engine". http://www.google.com.

[GRG98] J. Gehrke, R. Ramakrishnan, and V. Ganti. "RainForest - A Framework for Fast Decision Tree Construction of Large Datasets". In *Proc. 24th Int. Conf. on Very Large Data Bases (VLDB'98), New York, NY, USA*, pages 416–427, 1998.

[GRS98] S. Guha, R. Rastogi, and K. Shim. "CURE: an efficient clustering algorithm for large databases". In *Proc. ACM SIGMOD Int. Conf. on Management of Data (SIGMOD'98), Seattle, WA, USA*, pages 73–84, 1998.

[HK98] A Hinneburg and D A Keim. "An Efficient Approach to Clustering in Large Multimedia Databases with Noise". In *Proc. 4th Int. Conf. on Knowledge Discovery and Data Mining (KDD'98), New York, NY, USA*, pages 224–228, 1998.

[HK00] E.-H. Han and G. Karypis. "Centroid-Based Document Classification: Analysis and Experimental Results". In *Proc. 4th European Conf. on Principles of Data Mining and Knowledge Discovery (PKDD'00), Lyon, France*, Lecture Notes in Computer Science (LNCS), Springer, pages 1910: 424–431, 2000.

[HK01] J. Han and M. Kamber. *"Data Mining Concepts and Techniques"*. Morgan Kaufmann Publishers, 2001.

[HKKM98] E.-H. Han, G. Karypis, V. Kumar, and B. Mobasher. "Hypergraph Based Clustering in High-Dimensional Data Sets: A Summary of Results". *IEEE Data Eng. Bull.*, 21(1):15–22, 1998.

[HNKW98] J. Han, S. Nishio, H. Kawano, and W. Wang. "Generalization-Based Data Mining in Object-Oriented Databases Using on Object Cube Model". *Data and Knowledge Engineering*, 1-2(25):55–97, 1998.

[HSE+95] J. Hafner, H.S. Sawhney, W. Equitz, M. Flickner, and Niblack W. "Efficient Color Histogram indexing for Quadratic Form Distance Functions.". *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 17(7):729–736, 1995.

[Jag91] H.V. Jagadish. "A Retrieval Technique for Similar Shapes". In *Proc. ACM SIGMOD Int. Conf. on Management of Data (SIGMOD'91), Denver, CO, USA*, pages 208–217, 1991.

[JB91] H.V. Jagadish and A.M. Bruckstein. "On sequential shape descriptions". *Pattern Recocnition*, 1991.

[JH98] T. S. Jaakkola and D. Haussler. "Exploiting generative models in discriminative classifiers". In *Neural Information Processing Systems Conf.,Denver, CO, USA*, pages 487–493, 1998.

[Joa98] T. Joachims. "Text Categorization with Suport Vector Machines: Learning with Many Relevant Features". In *Proc. 10th European Conf. on Machine Learning (ECML'98), Chemnitz, Germany*, Lecture Notes in Computer Science (LNCS), Springer, pages 1398: 137–142, 1998.

[KAH96] K. Koperski, J. Adhikary, and J Han. "Spatial Data Mining: Progress and Challanges". In *Proc. SIGMOD'96 Workshop on Research Issues on Data Mining and Knowledge Discovery (DMKD'96), Montreal, Canada*, 1996.

[KBD01] L.I. Kuncheva, J.C. Bezdek, and R.P.W. Duin. "Decision Templates for Multiple Classifier Fusion: an Experimental Comparison". *Pattern Recognition*, 34:299–314, 2001.

[Kei99] D.A. Keim. "Efficient Geometry-based Similarity Search of 3D Spatial Databases". In *Proc. ACM SIGMOD Int. Conf. on Management of Data (SIGMOD'99), Philadelphia, PA, USA*, pages 419–430, 1999.

[KH98] D. Kudenko and H. Hirsh. "Feature Generation for Sequence Categorization". In *Proc. 15th Nat. Conf. on Artificial Intelligence (AAAI'98),Madison, WJ,USA*, pages 733–738, 1998.

[KHDM98] J. Kittler, M. Hatef, R.P.W. Duin, and J. Matas. "On Combining Classifiers". *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(3):226–239, 1998.

[KHK99] G. Karypis, E.-H Han, and V. Kumar. "Chameleon: Hierarchical Clustering using Dynamic Modelling". *IEEE Computer*, pages 68–75, August 1999.

[KKM+03] H.-P. Kriegel, P. Kröger, Z. Mashael, M. Pfeifle, M. Pötke, and T. Seidl. "Effective Similarity Search on Voxelized CAD Objects". In *Proc. 8th Int. Conf. on Database Systems for Advanced Applications (DASFAA'03), Kyoto, Japan*, pages 27–36, 2003.

[KKPS04a] K. Kailing, H.-P. Kriegel, A. Pryakhin, and M. Schubert. "Clustering Multi-Represented Objects with Noise". In *Proc. 8th Pacific-Asia Conf. on Knowledge Discovery and Data Mining (PAKDD'04), Sydney, Australia*, pages 394–403, 2004.

[KKPS04b] H.-P. Kriegel, P. Kröger, A. Pryakhin, and M. Schubert. "Using Support Vector Machines for Classifying Large Sets of Multi-Represented Objects". In *Proc. SIAM Int. Conf. on Data Mining (SDM'2004), Lake Buena Vista, FL, USA*, pages 102–113, 2004.

[KKSS04] K. Kailing, H.-P. Kriegel, S. Schönauer, and T. Seidl. "Efficient Similarity Search for Hierachical Data in Large Databases". In *Proc. 9th Int. Conf. on Extending Database Technology, (EDBT'04), Heraklion, Greece*, pages 676–693, 2004.

[KS97] D. Koller and M. Sahami. "Hierarchically Classifying Documents Using Very Few Words". In *Proc. 14th Int. Conf. on Machine Learning (ICML'97), Nashville, TN, USA*, pages 170–178, 1997.

[KS04] H.-P. Kriegel and M. Schuber. "Classification of websites as sets of feature vectors". In *Proc. IASTED Int. Conf. on Databases and Applications (DBA 2004), Innsbruck, Austria*, 2004.

[KSF+96] F. Korn, N. Sidiropoulos, C. Faloutsos, E. Siegel, and Z. Protopapas. "Fast Nearest Neighbor Search in Medical Image Databases". In *Proc. 22nd Int. Conf. on Very Large Data Bases (VLDB'96), Mumbai (Bombay), India*, pages 215–226, 1996.

[KSS97] H.-P. Kriegel, T. Schmidt, and T. Seidl. "3D Similarity Search by Shape Approximation". In *Proc. 5th Int. Symposium on Large Spatial Databases (SSD'97), Berlin, Germany*, Lecture Notes in Computer Science (LNCS), Springer, pages 1262:11–28, 1997.

[Kuh55] H.W. Kuhn. "The Hungarian method for the assignment problem". *Naval Research Logistics Quarterly*, 2:83–97, 1955.

[Lar98] L. Larkey. "Some Issues in the Automatic Classification of U.S. Patents". In *Learning for Text Categorization. Papers from the 1998 Workshop*, pages 87–90. AAAI Press, 1998.

[LJF94] K.-I. Lin, H.V. Jagadish, and C. Faloutsos. "The TV-Tree: An Index Structure for High-Dimensional Data". *VLDB Journal*, 3(4):517–542, 1994.

[LZO99] N. Lesh, M.J. Zaki, and M. Ogihara. "Mining Features for Sequence Classification". In *Proc. 5th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining (SIGKDD'99), San Diego, CA, USA*, 1999.

[MCN98] A. Mc Callum and K. Nigam. "A Comparison of Event Models for Naive Bayes Text Classification". In *Proc. of AAAI-98 Workshop on Learning for Text Categorization, Madison, WJ, USA*, 1998.

[MDR94] S. Muggleton and L. De Raedt. "Inductive Logic Programming: Theory and Methods". *Journal of Logic Programming*, 19/20:629–679, 1994.

[MG93] R. Mehrotra and J.E. Gary. "Feature-Based Retrieval of Similar Shapes". In *Proc. 9th Int. Conf. on Data Engineering (ICDE'93), Vienna, Austria*, pages 108–115, 1993.

[Mou01] D.W. Mount. *"Bioinformatics: Sequence and Genome Analysis"*. CSHL Press, 2001.

[MRMN98] A. McCallum, R. Rosenfeld, T. Mitchell, and A. Ng. "Improving Text Classification by Shrinkage in a Hierarchy of Classes ". In *Proc. 15th Int. Conf. on Machine Learning (ICML'98), Madison, WI, USA*, pages 359–367, 1998.

[Mun57] J. Munkres. "Algorithms for the assignment and transportation problems". *Journal of the SIAM*, 6:32–38, 1957.

[MV97] M.V. Menshikov and S.E Volkov. "Branchin Markov Chains: Qualitative Characteristics". *Markov Processes Related Fields*, 3:1–18, 1997.

[NBE+93] W. Niblack, R. Barber, W. Equitz, M. Flickner, E. Glasmann, D. Petkovic, P. Yanker, C. Faloutsos, and G. Taubin. "The

QBIC Project: Querying Images by Content Using Color, Texture, and Shape". In *SPIE 1993 Int. Symposium on Electronic Imaging: Science and Technology Conf., Storage and Retrieval for Image and Video Databases, San Jose, CA, USA*, pages 1908:173–187, 1993.

[NRS+95] T. Newman, E.F. Retzel, E. Shoop, E. Chi, and C. Somerville. "Arabidopsis thaliana expressed sequence tags: Generation, analysis and dissemantation". In *In Plant Genome III: Internation Conference on the Status of Plant Genome Research, San Diego, CA, USA*, 1995.

[PCST00] J. Platt, N. Cristianini, and J. Shawe-Taylor. "Large Margin DAGs for Multiclass Classification". In *Advances in Neural Information Processing Systems 12 (NIPS Conference, Denver, CO, 1999)*, pages 547–553. MIT Press, 2000.

[Pla98] J. Platt. "Sequential minimal optimization: A fast algorithm for training support vector machines". Technical Report 98-14, Microsoft Research, Redmond, Washington, 1998.

[Pro] Zooknic Internet Geography Project. "History of gTLD domain name growth". http://www.zooknic.com/Domains/counts.html.

[PV98] M. Pontil and A. Verri. "Object recognition with support vector machines". *IEEE Transaction on Pattern Analysis Machine Intelligence*, 20:637–646, 1998.

[Qui93] J.R. Quinlan. *"C4.5: Programms of Machine Learning"*. Morgen Kaufmann, San Mateo, CA, 1993.

[RB01] J. Ramon and M. Bruynooghe. "A polynomial time computable metric between points sets". *Acta Informatica*, 37:765–780, 2001.

[RM99] J. Rennie and A. McCallum. "Using Reinforcement Learning to Spider the Web Efficiently". In *Proc. 16th Int. Conf. on Machine Learning (ICML'99), Bled, Slovenia*, pages 335–343, 1999.

[Roc71] J. Rocchio. "The SMART Retrieval System: Experiments in Automatic Document Processing". In *Relevance Feedback in Information Retrieval*, pages 313–323. Prentice-Hall Inc., 1971.

[Sal89] G. Salton. *"Automatic Text Processing: The Transformation, Analysis, and Retrieval of Information by Computer"*. Addison-Wesley, 1989.

[SCC+95] E. Shoop, E. Chi, J. Carlis, P. Bieganski, J. Riedl, N. Dalton, T. Newman, and E. Retzel. "Implementation and testing of an automated EST processing and similarity analysis system". In *Proc. 28th Annual Hawaii Int. Conf. on System Sciences (HICSS-28),Maui, HW, USA*, pages 5: 52–61, 1995.

[SCW+03] R. She, F. Chen, K. Wang, M. Ester, J.L. Gardy, and F.S.L. Brinkman. "Frequent-subsequence-based prediction of outer membrane proteins". In *Proc. 9th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining (SIGKDD'03), Washington, D.C., USA*, pages 436–445, 2003.

[SEKX98] J. Sander, M. Ester, H.-P. Kriegel, and X. Xu. "Density-Based Clustering in Spatial Databases: The Algorithm GDBSCAN and its Applications". *Data Mining and Knowledge Discovery*, 2(2):169–194, 1998.

[SH94] H. Sawhney and J. Hafner. "Efficient Color Histogram Indexing". In *Proc. Int. Conf. on Image Processing (ICIP'94),Austin, TX, USA*, pages 66–70, 1994.

[SK98] T. Seidl and H.-P. Kriegel. "Optimal Multi-Step k-Nearest Neighbor Search". In *Proc. ACM SIGMOD Int. Conf. on Management of Data (SIGMOD'98), Seattle, WA, USA*, pages 154–165, 1998.

[SKSH89] R. Schneider, H.-P. Kriegel, B. Seeger, and S. Heep. "Geometry-based Similarity Retrieval of Rotational Parts". In *Proc. Int. Conf. on Data and Knowledge Systems for Manufacturing and Engineering, Gaithersburg, ML*, pages 150–160, 1989.

[Tom04] S. Tomlinson. "Lexical and Algorithmic Stemming Compared for 9 European Languages with Hummingbird Search Server at CLEF 2003". In *Working Notes for the CLEF 2003 Workshop, Trondheim, Norway*, 2004.

[VM02] G. Valentini and F. Masulli. "Ensembles of learning machines". In *Proc. 13th Int. Workshop on Neural Nets (WIRN VIETRI'02), Vietri sul Mare (SA), Italy*, pages 3–22, 2002.

[VMD00] S. Vaithyanathan, J. Mao, and B. Dom. "Hierarchical Bayes for Text Classification". In *Proc. Int. Workshop on Text and Web Mining (PRICAI'00), Melbourne, Australia*, pages 36–43, 2000.

[VT00] R.C. Veltkamp and M. Tanase. "Content-based image retrieval systems: A survey". Technical Report UU-CS-2000-34, Department of Computing Science, Utrecht University, October 2000.

[WF99] I. Witten and E. Frank. *"Data Mining: Practical Machine Learning Tools and Techniques with Jave Implementations"*. Morgan Kaufmann, 1999.

[WMSW01] J.T.L. Wang, Q. Ma, D. Shasha, and C.H. Wu. "New techniques for extracting features from protein sequences". *IBM Systems Journal*, 40(2), 2001.

[WYM97] W. Wang, J. Yang, and R.R. Muntz. "STING: A Statistical Information Grid Approach to Spatial Data Mining". In *Proc. 23rd Int. Conf. on Very Large Data Bases (VLDB'97), Athens, Greece*, pages 186–195, 1997.

[WZ00] J. Wang and J.D. Zucker. "Solving Multiple-Instance Problem: A Lazy Learning Approach". In *Proc. 17th Int. Conf. on Machine Learning (ICML'00), Stanford, CA, USA*, pages 1119–1125, 2000.

[WZC+03] J. Wang, H. Zeng, Z. Chen, H. Lu, L. Tao, and W. Ma. "ReCoM: reinforcement clustering of multi-type interrelated data objects". In *Proc. 26th ACM SIGIR Conf. on Research and Development in Information Retrieval (SIGIR'03), Toronto, Canada*, pages 274–281, 2003.

[WZL99] K. Wang, S. Zhou, and S.C. Liew. "Building Hierarchical Classifiers Using Class Proximity". In *Proc. 25th Int. Conf. on Very Large Databases (VLDB'99), Edinburgh, Scotland, UK*, pages 363–374, 1999.

[XEKS98] X. Xu, M. Ester, H.-P. Kriegel, and J. Sander. "A Distribution-Based Clustering Algorithm for Mining in Large Spatial Databases". In *Proc. 14th Int. Conf. on Data Engineering (ICDE'98), Orlando, FL*, pages 324–331. AAAI Press, 1998.

[Yah] Yahoo! "web directory service". http://www.yahoo.com.

[Yan97] Y. Yang. "An Evaluation of Statistical Approaches to Text Categorization". Technical Report CMU-CS-97-127, Carnegie Mellon University, April 1997.

[YL99] Y. Yang and X. Liu. "A Re-Examination of Text Categorization Methods". In *Proc. 22nd ACM SIGIR Conf. on Research and Development in Information Retrieval (SIGIR'99),Berkley, CA, USA*, pages 42–49, 1999.

[YP97] Y. Yang and P.O. Pederson. "A comparative study on feature selection in text categorization". In *Proc. 14th Int. Conf. on Machine Learning (ICML'97), Nashville, TN, USA*, pages 412–420, 1997.

[Zak01] M. Zaki. "SPADE: An Efficient Algorithm for Mining Frequent Sequences". *Machine Learning Journal*, 42(1-2):31–60, January, February 2001.

[ZCM02] H. Zeng, Z. Chen, and W. Ma. "A Unified Framework for Clustering Heterogeneous Web Objects". In *Proc. 3rd Int. Conf. on Web Information System Engineering (WISE 2002), Singapore*, pages 161–172, 2002.

[ZRL96] T. Zhang, R. Ramakrishnan, and M. Livny. "BIRCH: An Efficient Data Clustering Method for Very Large Databases". In *Proc. ACM SIGMOD Int. Conf. on Management of Data (SIGMOD'96), Montreal, Canada*, pages 103–114, 1996.

Curriculum Vitae

Matthias Schubert was born on September 19, 1973 in Munich, Germany. He attended primary school from 1980 to 1984, and high-school from 1985 to 1993. From Juli 1993 until June 1994 he provided his basic military service in Stetten a.k.M, Germany as a staff soldier and driver.

He entered the *Ludwig-Maximilians-Universität München* (LMU) in November 1994, studying Computer Science with a minor in business studies. During his basic studies Matthias was employed as student trainee and in 1998 he started to work as a freelance database and SAP Consultant until March 2001. In September 2000 he received his diploma. His diploma thesis was on "Entwicklung und Evaluierung von Optimierungstechniken für die Ähnlichkeitssuche in Multimedia Datenbanken" ("Development and Evaluation of Optimization techniques for similarity queries in Multimedia databases") and was supervised by Professor Hans-Peter Kriegel and Dr. Bernhard Braunmüller.

In March 2003, Matthias Schubert started to work as a research and teaching assistant at the University of Munich in the group of Professor Hans-Peter Kriegel. The research interests of Matthias include knowledge discovery in databases in general and more particular the areas of web content mining, data mining in complex objects, text mining and multimedia databases.