Semantic Patterns

by

Peter Franz Teufl

A PhD Thesis
Presented to the Faculty of Computer Science in Partial Fulfillment of the Requirements for the PhD Degree

Assessors

Prof. Dr. Reinhard Posch (IAIK, Graz University of Technology, Austria)
Prof. Dr. Michael Granitzer (FIM, University of Passau, Germany)

March 2012

Institute for Applied Information Processing and Communications (IAIK)
Faculty of Computer Science
Graz University of Technology, Austria

Abstract

The process of extracting information and hitherto unknown relations from arbitrary data sets is known as knowledge discovery, and is widely deployed in academic and industrial processes. Thereby, machine learning algorithms play an important role due to their capability to analyze data for which only limited a priori knowledge is available. Unfortunately, their application and the extraction of information from the trained algorithm models highly depend on the nature of the analyzed data and the algorithm models. Therefore, the deployment of knowledge discovery processes in heterogeneous domains causes the requirement for time-consuming process adaptations.

This thesis argues that the value-centric feature vector representation used within machine learning is the main reason for the necessity to create such highly domain-specific setups. Therefore, the shift from the value-centric representation to a semantic representation is suggested. This transformation is achieved by a new method proposed in this thesis – the *Semantic Pattern Transformation*. The principle idea behind this process is to analyze the semantic relations between the feature values within the value-centric feature vectors, and store this information in a new semantic representation – the *Semantic Patterns*. With this new representation, many of the domain-specific processing steps and adaptations can be avoided. This significantly simplifies the deployment of knowledge discovery processes in heterogeneous domains. Furthermore, due to the employed model, which is independent of the analyzed data and the applied algorithms, many advantages regarding the interpretation and the analysis of arbitrary data are gained.

The *Semantic Pattern Transformation* is based on well-known techniques such as associative networks, spreading activation, and standard machine learning techniques. In previous works, the new method has been applied within a wide range of knowledge discovery processes. By conducting empirical evaluations on the gained results, many improvements and extensions to the initial versions of the transformation process could be made. This thesis extends these works in the following way:

First, an in-depth analysis on the integration of machine learning algorithms into knowledge discovery processes is conducted. Thereby, the problems associated with the deployment of such schemes in heterogeneous domains are identified. **Second**, by analyzing these problems and their core reasons, the shift from a value-centric to a semantic representation is motivated. **Third**, the *Semantic Pattern Transformation*, its application to arbitrary data, and the subsequent analysis and interpretation of the generated *Semantic Patterns* are explained in

detail. **Finally**, the proposed method is throughly evaluated by analyzing the results gained from unsupervised and supervised machine learning algorithms, and a semantic-aware search algorithm.

Kurzfassung

Aufgrund der signifikanten Menge an elektronischen Daten spielt sowohl in akademischen als auch in industriellen Prozessen das Extrahieren von bisher unbekannten Relationen und Wissen aus diesen Daten eine entscheidende Rolle. Dieser Prozess ist unter dem Begriff *Knowledge Discovery* bekannt und bedient sich dabei in vielen Fällen an Algorithmen aus dem Maschinellen Lernen. Diese Algorithmen spielen eine entscheidende Rolle, da sie sich sich aufgrund ihrer speziellen Eigenschaften hervorragend für die Analyse von Daten verwenden lassen, für die kein oder nur ein sehr beschränktes Vorwissen vorhanden ist. Werden diese Algorithmen aber in heterogenen Domänen mit unterschiedlichen Daten und/oder abweichenden *Knowledge Discovery* Zielen angewendet, muss im Normalfall eine aufwendige Anpassung der verwendeten Vorverarbeitungsschritte durchgeführt werden. Zusätzlich müssen die Prozesse für die Interpretation und Extraktion des Wissens sowohl an die Modelle der verwendeten Algorithmen als auch an die spezifischen Eigenschaften der analysierten Daten angepasst werden.

In dieser Arbeit wird argumentiert, dass die Verwendung der wert-basierten Featurevektoren, wie sie im Maschinellen Lernen eingesetzt werden, einer der Hauptgründe für diese aufwendigen Adaptierungsschritte ist. Aus diesem Grund wird ein neuer Transformationsprozess vorgestellt, welcher als *Semantic Pattern Transformation* bezeichnet wird. Dieser Prozess führt die wert-zentrischen Featurevektoren durch die Analyse der Relationen zwischen den einzelnen Featurewerten in eine semantische Repräsentation über, die als *Semantic Patterns* bezeichnet wird. Durch diese neue Repräsentation fallen viele der bisher notwendigen Verarbeitungsschritte, und daher auch deren aufwendige Anpassung beim Einsatz in unterschiedlichen Domänen weg. Außerdem kann die Wissensextraktion und Interpretation am Ende eines *Knowledge Discovery* Prozesses signifikant vereinfacht werden, da unabhängig von den analysierten Daten das gleiche Modell eingesetzt wird.

Die *Semantic Pattern Transformation* basiert dabei auf bekannten Technologien wie Assoziativen Netzwerken, Spreading Activation Algorithmen, und Standardalgorithmen aus dem Maschinellen Lernen. In jenen, dieser Arbeit vorangehenden Publikationen, wurde die *Semantic Pattern Transformation* bereits in vielen unterschiedlichen *Knowledge Discovery* Bereichen angewendet. Die dabei empirisch gewonnenen Ergebnisse wurden dazu verwendet, um den Transformationsprozess zu optimieren und um weitere Analyseverfahren zu ergänzen. Die vorliegende Arbeit ergänzt diese Punkte wie folgt: **Erstens**: Es wird eine detaillierte Analyse über die Verwendung von Maschinellem Lernen im *Knowledge Discovery* Bereich durchgeführt. Dabei können systematisch die typischen

Probleme erkannt werden, die beim Anwenden in heterogenen Domänen auftreten. **Zweitens**: Basierend auf diesen Ergebnissen wird die Umwandlung der wert-zentrischen Featurevektoren zu einer semantischen Repräsentation motiviert und argumentiert. **Drittens**: In weiterer Folge wird die *Semantic Pattern Transformation* und ihre Anwendung zur Analyse von beliebigen Daten im Detail erklärt. **Abschließend** wird eine detaillierte Evaluierung der neuen Methode in den Bereichen überwachtes-, unüberwachtes Lernen, und semantische Suche durchgeführt.

Table of Contents

List of Tables

List of Figures

...gold can be created only by stars and by intelligent beings. If you find a nugget of gold anywhere in the universe, you can be sure that in its history there was either a supernova or an intelligent being with an explanation. And if you find an explanation anywhere in the universe, you know that there must have been an intelligent being. A supernova alone would not suffice...

David Deutsch, The Beginning of Infinity [17]

1 Introduction

Knowledge Discovery is the process of extracting knowledge and retrieving previously unknown relations from arbitrary data sets. Due to the every increasing amount of electronically available data, it is a key process employed within many research areas and industrial processes. In order to cope with noise and the fuzzy nature of arbitrary data sets, machine learning algorithms are widely used within these processes. Unfortunately, in heterogeneous domains – even when the same knowledge discovery process such as clustering is applied – a specific machine learning algorithm needs to be selected for each data set according to the desired knowledge and the nature of the data. Depending on the selected algorithm, domain-specific processes for data preprocessing and transformation, as well as specific knowledge extraction procedures need to be defined. The necessary setup procedure is time-consuming and requires in-depth knowledge about the selected algorithm and the analyzed data. One of the main reasons for these issues is the value-centric feature vector representation, which is typically employed within machine learning.

In order to address these problems, this thesis presents a new method – the *Semantic Pattern Transformation* – which transforms the value-centric feature vectors into a new semantic representation – the *Semantic Patterns*. Due to this transformation many of the domain- and algorithm-specific preprocessing, transformation and knowledge extraction procedures can be avoided or replaced by general methods regardless of the employed algorithm and the analyzed data. Furthermore, the *Semantic Patterns* represent a generic model that **first** forms the basis for the application of arbitrary analysis processes including machine learning algorithms, and **second**, due to its semantic nature enables the simple extraction and interpretation of knowledge.

The remainder of this chapter describes the organization of this thesis and

provides an additional guide that focuses on the individual interests of the reader. Furthermore, the principle motivation for the development of the presented technique is discussed by looking back at my own experiences gained during the application of cross-disciplinary knowledge discovery processes. Finally, the main contributions of this thesis are summarized.

1.1 Organization

The thesis is organized in eleven chapters that describe all aspects of the *Semantic Pattern Transformation* and the gained *Semantic Patterns*. These are described as follows:

- *Chapter 1 – Introduction*: The first chapter describes the organization of the thesis and provides a guide for reading the eleven chapters. Furthermore, the principle motivation for the presented *Semantic Pattern Transformation* is given by looking back at my work within cross-disciplinary knowledge discovery. Thereby, all the published scientific works with relevance to this thesis are referenced and briefly explained.

- *Chapter 2 – Semantic Patterns - At a Glance*: This chapter is an extension to *Chapter 1 – Introduction*, and gives a superficial overview of the *Semantic Pattern Transformation* by analyzing its properties and benefits on the basis of a simple demonstration data set.

- *Chapter 3 – Knowledge Discovery and Machine Learning*: This chapter gives a detailed analysis of the main obstacles when utilizing machine learning algorithms within knowledge discovery processes in heterogeneous domains.

- *Chapter 4 – Towards a Semantic Representation*: In this chapter, the value-centric representation within feature vectors is identified as the main reason for the problems discussed in the previous chapter. This key issue forms the principle motivation for introducing a semantic-aware representation, which is achieved by transforming raw feature vectors into *Semantic Patterns* via the *Semantic Pattern Transformation*.

- *Chapter 5 – Techniques*: Here, the techniques required by the various processing layers of the *Semantic Pattern Transformation* are explained. The three main components are unsupervised clustering, associative networks and spreading activation.

- *Chapter 6 – Semantic Pattern Transformation*: The core contribution of this work is presented in detail in this and the following chapter. Thereby, this chapter focuses on the detailed aspects of the five processing layers required for the transformation of raw feature vectors into *Semantic Patterns*.

- *Chapter 7 – Semantic Pattern Analysis*: This chapter analyzes the structure of the *Semantic Patterns*, explains the interpretation of the stored semantic information and discusses the specific details of the wide range of analysis methods used for the extraction of knowledge.
- *Chapter 8 – Evaluation*: This chapter presents a detailed evaluation of the presented method by comparing the performance of unsupervised and supervised machine learning algorithms, and semantic-aware search algorithms applied to raw feature vectors and *Semantic Patterns*. The different algorithms are evaluated by various well-known data sets and standard quality measures used within machine learning.
- *Chapter 9 – Related Work*: Although, the relevant related work is discussed throughout the thesis at the appropriate locations, this chapter provides further details by looking specifically at related concepts and other works that had a significant influence during the development of the *Semantic Pattern Transformation*.
- *Chapter 10 – Applications*: This chapter discusses the applications of the *Semantic Pattern Transformation* in a wide range of knowledge discovery domains ranging from text-analysis within e-Participation, over event correlation in intrusion detection, to the analysis of data extracted from the semantic web. All of these examples have been published in scientific papers, which are described in detail in this chapter. Furthermore, the various applications also give an insight on the evolution of the *Semantic Pattern Transformation*.
- *Chapter 11 – Conclusions*: Finally the thesis is concluded in this chapter by looking at the achievements, the remaining issues and giving an outlook to future research work.

1.2 How to Read?

This section presents a short guide for reading this thesis by recommending chapters based on the desired information:

- *Getting an overview:* By reading the following chapters, the reader should get a good overview on the principle motivation and benefits of the *Semantic Pattern Transformation*.
  - *Chapter 1 – Introduction*: Here, the own experiences regarding knowledge discovery in heterogeneous domains and the encountered problems are given. Based on these issues the principal motivation for the *Semantic Pattern Transformation* is explained.
  - *Chapter 2 – Semantic Patterns - At a Glance*: The main idea of the *Semantic Pattern Transformation* and the structure of the *Semantic Patterns* are explained on the basis of a simple demonstration data

set. Although only superficial aspects are covered, the reader should get a good impression on the rationale behind the presented method.

  - *Chapter 10 – Applications*: In this chapter, the evolution of the development process is covered by explaining all the applications of the *Semantic Pattern Transformation* that have been published in scientific papers. For gaining a quick overview it is recommended to read the summary for each of these applications.
  - *Chapter 11 – Conclusions*: Finally, thoughts on the whole concept, remaining issues and an outlook are given.

- *Looking at the motivation:* When reading the following chapters, the reader should be able to understand the main problems, which form the principle motivation for developing the presented technique.
  - *Chapter 3 – Knowledge Discovery and Machine Learning*: This chapter describes how machine learning algorithms are used within knowledge discovery processes and describes the main problems when applied within heterogeneous domains.
  - *Chapter 4 – Towards a Semantic Representation*: This chapter discusses the reasons for shifting the value-centric feature vector representation typically employed in machine learning to the semantic representation used by the *Semantic Patterns*.
  - *Chapter 10 – Applications*: By looking at the various analysis methods employed by the presented applications, the motivation and benefits of a common semantic-aware model become clear.

- *Details on the Semantic Pattern Transformation:* The following chapters convey the in-depth details of the *Semantic Pattern Transformation* to the reader.
  - *Chapter 5 – Techniques*: This chapter describes all the techniques used by the transformation process, which include unsupervised learning, associative networks and spreading activation.
  - *Chapter 6 – Semantic Pattern Transformation*: In this chapter, the transformation process which is organized in five layers is described in detail. For a better understanding of these process layers, simple examples are given within each layer.
  - *Chapter 7 – Semantic Pattern Analysis*: This chapter explains the structure of the gained *Semantic Patterns*, their interpretation, and the basic methods used for their analysis. Furthermore, more details regarding the parameters that influence the transformation process are covered.
  - *Chapter 10 – Applications*: This chapter describes the fine aspects of the problems, which were solved by applying the *Semantic Pattern Transformation* to data from heterogeneous domains.

- *Evaluation and comparison to other work*: Finally, for understanding the relations to other works and the evaluation of the *Semantic Pattern Transformation* the following two chapters are of importance.
  - *Chapter 8 – Evaluation*: The *Semantic Pattern Transformation* is evaluated by analyzing the results of unsupervised and supervised learning, and the application of a semantic-aware search algorithm.
  - *Chapter 9 – Related Work*: This chapter describes other work that is either related to the presented technique or had a significant influence during the development phase.

1.3 Defining the Meaning of *Semantic*

The *Semantic Pattern Transformation*, as well as the gained representation – the *Semantic Patterns* – use the term "semantic" within their name. Furthermore, it is used throughout this thesis for the description of the presented method and various analysis processes, such as semantic-aware search algorithms. Before going into the details of the *Semantic Pattern Transformation*, and the associated techniques and processes, the meaning of the term "semantic" within the context of this thesis needs to be defined.

The term "semantic" is quite often used within the context of linguistics and describes the relationship between words or sentences. A good example is Wordnet [25], an electronic lexical database, which links terms of the English language according to different semantic relations. More recently, the term "semantic" started to play an important role within the "semantic web", where rich semantic relations are defined between arbitrary concepts. These concepts and their relations are modeled by the Resource Description Framework (RDF) language [59]. Furthermore, the concept plays an important role in search engines for increasing the quality of the results, and many other areas, such as semantic web services or modeling relationships within social networks.

However, within the *Semantic Pattern Transformation* associative networks are used to model the relationship between arbitrary feature values. In contrast to semantic networks, associative networks only define one type of generic relation between arbitrary concepts – an association. In this work, an association between two feature values is created, when they co-occur within a defined context. An example would be a weather station, where the temperature of 5 degrees Celsius and the wind speed of 40 knots are measured at a certain time. Both measurements are taken within the same context, which in this case is the time of measurement. Thus, an association between the feature value *5* of the feature *temperature*, and the feature value *40* of the feature *wind speed* is created. Such an association cannot be seen as a typical semantic relation yet, because only the fact, that these two feature values are associated due to some context, is modeled.

Thus, the *Semantic Pattern Transformation* would more accurately be described as an *Associative Pattern Transformation*, and the question on why the

term "semantic" is used, comes to mind. While the low-level analysis is based on associations, the high level analysis processes, which are applied to the *Semantic Patterns* are better described via the meaning of "semantic". A good example would be the application of "semantic" search-queries. While strictly speaking, this still would be an associative search query, the literature also uses the term "semantic" to refer to such processes. This is best highlighted by the well-known Latent Semantic Indexing (LSI) method [52], which focuses on the analysis of documents and terms by analyzing the associations between the contained terms. Therefore, the term "semantic" was chosen in concordance with its high level usage and meaning within the literature.

1.4 The *Semantic Pattern Transformation*

The remainder of this chapter explains the motivation for devising the *Semantic Pattern Transformation*, by looking back on my own experiences gained by the application of machine learning within cross-disciplinary knowledge discovery processes. This is followed by a short summary, which describes how the *Semantic Pattern Transformation* evolved over the last three years. Finally, the main contributions of this thesis and the *Semantic Pattern Transformation* will be summarized, and a list of relevant publications will be given.

1.4.1 Cross-Disciplinary Knowledge Discovery

This section describes my experiences with the application of knowledge discovery processes (machine learning) within intrusion and malicious code detection. Based on the experiences gained there, the development of the *Semantic Pattern Transformation* was motivated.

Polymorphic Shellcode Detection, Network Traffic Classification and WiFi Analysis

The initial work goes back to my master's thesis [76] which was conducted at the *Institute for Applied Information Processing and Communications (IAIK)* at the *Graz University of Technology*. The idea of this master's thesis was the combination of the security related areas of the *IAIK* with the knowledge I have gained in machine learning at the *Institute for Theoretical Computer Science (IGI)*, which is located at the same university. Therefore, the master's thesis analyzed two security relevant applications of machine learning algorithms. The first part focused on the detection and analysis of polymorphic shellcodes that employ camouflaging techniques to avoid the detection by network-based intrusion detection systems. Due to these techniques, the application of typical simple signature based detection methods was not possible. Thus, the network traffic data was analyzed with a trained neural network for the detection of possible decryption engines that are employed by polymorphic shellcodes. The fuzzy nature of machine learning improved the detection rates of the always changing

polymorphic shellcodes significantly. These promising results were published in [64] and [63].

In the second part of the master's thesis I focused on the identification of network traffic by analyzing histograms containing the byte-occurrence values extracted from TCP/IP packets. Again, the fuzzy nature of machine learning algorithms is of benefit, since a network protocol is represented by many histograms that have some kind of similarity. Here, Self-Organizing Maps (SOM) in their standard unsupervised nature were used to get a visual impression of the 256-dimensional histograms. In addition, a supervised version of the SOM was devised for classifying the histograms and assigning them to various protocols. The prototypes were later implemented as a Java framework called INFECT and the results were published in [62] and [82].

In later work, similar algorithms were utilized for the identification of WLAN chipsets by looking at the differences in the timing characteristics of the captured traffic. Here, similar to the network traffic classification project, histograms were used as feature vectors. However, this time the byte-occurrence values were replaced with the occurrence of the delay times for the acknowledge messages used by WiFi networks. The initial results and several extensions were published in [46], [47], and [48].

E-Particpation, Event Correlation and Machine Language Processing

Due to a shift of focus from security related areas to e-Government processes, the lessons learned in machine learning were soon applied to an e-Government related area – e-Participation. Here, the automated analysis of text documents and the understanding of relations between terms and concepts play an important role. Again, machine learning algorithms for clustering and semantic search were employed for implementing various knowledge discovery tasks. However, the feature vectors used in text analysis are quite different from those that were analyzed in the security related domains. While the previously described histograms contain values that can be put into relation via a distance measure, the main feature values used in text analysis – words – are of symbolic nature, and cannot be compared with standard distance measures. Therefore, in text analysis, a semantic-aware algorithm called Latent Semantic Indexing (LSI) is often used which analyzes the semantic relations between terms and documents. By using LSI, the semantic information contained in the data is retained and significantly improves the quality of the results gained by machine learning, and semantic search algorithms. However, the difficult interpretation of the LSI models and the lack of support for numerical features limited its application in text analysis and especially in other areas that involved the analysis of arbitrary combinations of features.

While investigating LSI for e-Participation, I revisited the security-related knowledge discovery area by analyzing events from intrusion detection systems. Here, the available features were quite different compared to those encountered in the previously described works about polymorphic shellcodes, and network and WiFi traffic analysis. While the histograms employed in these areas contain

numerical values, the features used in intrusion detection are a combination of symbolic and numerical features. Thereby, four main problems were identified:

The **first** problem is related to the nature of the numerical features, which include properties like error rates, connections per second or the amount of transferred data. In contrast to the numerical histograms, the value ranges of these features change from feature to feature, which leads to the requirement for various preprocessing operations such as normalization. The **second** problem occurs due to the combination of symbolic and numerical features. While meaningful similarity measures can easily be defined for numerical features vectors, these measures do not cover the similarity between symbolic feature vectors. The **third** problem is caused by the variation in the number of features used for the description of events. This variability requires further data-specific preprocessing operations. Finally, the **fourth** problem is related to the nature of the data. In contrast to text data or the byte-occurrence histograms, a priori knowledge was not available for the event correlation data. Thus, meaningful interpretation methods for getting an overview and discovering hitherto unknown relations were required. These methods had to be adapted to the specific data and the utilized machine learning algorithm models. This is a time-consuming and complex process that significantly increases the effort for the extraction of meaningful knowledge.

1.4.2 The Need for a Semantic Representation

When analyzing the experiences of the cross-disciplinary application of machine learning within knowledge discovery, three core problems can be identified that limit the possibility to rapidly deploy existing setups in heterogeneous domains:

- The requirement for using domain-specific machine learning algorithms even when the same knowledge discovery task – such as unsupervised clustering – is applied.
- The requirement for algorithm and domain-specific preprocessing tasks such as normalization, dealing with missing values, or handling symbolic and numerical features.
- The need for adapting the interpretation of the results depending on the algorithm model.

A closer look reveals that the main reason for these problems is the value-centric representation employed by feature vectors that are typically used within machine learning. Thus, the *Semantic Pattern Transformation* was developed, which shifts the focus from the feature values to the semantic relations between these values. Thereby, the feature vectors are transformed into a semantic representation – the *Semantic Patterns*. This new representation removes the requirement for a large number of preprocessing steps including normalization, dealing with missing values, or choosing a specific algorithm based on the nature of the data.

First Applications: e-Participation and Malware Analysis

The first tests and the the initial evaluation was conducted on an e-Participation related data set and presented in [84]. Here only symbolic features (words) were used, which simplified the initial algorithm and its evaluation. The *Semantic Pattern Transformation* was then extended to cover the heterogeneous features encountered in event correlation [83]. During the initial evaluation we also combined the lessons learned from Natural Language Processing (NLP) with the knowledge gained from polymorphic shellcode analysis and employed the new concept for the analysis of disassembled sequences of polymorphic shellcode engines. The results were presented in [81].

Refining the Transformation: Semantic Web Data Analysis and Privacy Aspects of WiFi Networks

The initial work on text-analysis and event correlation resulted in the extension of the *Semantic Pattern Transformation*, which enabled its application to complete heterogeneous data sets. An example for such a data set is the CIA World Factbook [11], which contains many numerical and symbolic properties of the world's countries. The results of applying the *Semantic Pattern Transformation* to this data are presented in [79] and [80].

Based on the learned lessons the new method was again refined and extended by anomaly detection capabilities, which were employed in the analysis of application meta data extracted from the Android Market [78], and in the analysis of privacy related problems in WiFi networks [50].

Adding Sophisticated Capabilities: Revisiting e-Participation

In the most recent work, the *Semantic Pattern Transformation* was extended with the capability to include and analyze time-based information. This was first used for the analysis of semantic developments over time frames of Twitter data covering the Egyptian revolution [77]. Also, a prototype of a user interface was created that enables knowledge discovery on arbitrary data sets transformed into *Semantic Patterns*.

1.5 The Main Contributions

The discussed publications are considered as prior work to this thesis. While they have played an important part in developing and improving the *Semantic Pattern Transformation* by empirically analyzing the results on a wide range of data sets, they lack a detailed description of the transformation process, and do not carry out a systematic evaluation. These shortcomings are addressed in this thesis. Its main contributions to the development and the analysis of the *Semantic Pattern Transformation* are: **First**, the development of the new method is motivated by doing an in-depth analysis of current knowledge discovery processes. **Second**, a detailed description of the whole transformation process and

the utilized techniques is given. **Finally**, the *Semantic Pattern Transformation* is systematically evaluated on a wide range of data sets in the fields of supervised classification, unsupervised learning, and semantic-aware search algorithms.

Regarding the *Semantic Pattern Transformation* itself, there are three main scientific contributions:

1. **Deployment of knowledge discovery processes in heterogeneous domains**: The *Semantic Pattern Transformation* analyzes the semantic relations between the feature values and transforms the value-centric feature vectors into *Semantic Patterns*. This transformation leads to a significant decrease in the complexity of the required adaptation processes, when knowledge discovery processes are deployed in heterogeneous domains. The *Semantic Pattern Transformation* can be applied regardless of the nature of the analyzed data, and the subsequently applied machine learning algorithms do not need specific preprocessing steps for analyzing the *Semantic Patterns*.

2. **Interpretation and Analysis**: The model employed by the *Semantic Patterns* can easily be interpreted and visualized. This plays an important role when analyzing data for which no, or only limited a priori knowledge is available. The way the semantic information is represented within the *Semantic Patterns* allows the application of simple techniques, such as addition or subtraction, as well as highly sophisticated analysis, such as machine learning.

   Furthermore, and most important, in stark contrast to the value-centric feature vector representation, the model employed by the *Semantic Pattern Transformation* remains the same regardless of the analyzed data, the defined knowledge discovery goals, and the applied algorithms. This hugely simplifies the application of analysis processes and knowledge extraction procedures, which do not need to be adapted whenever the analyzed data or the knowledge discovery process goals change. Also, existing analysis processes can be easily extended or arranged in analysis chains for the creation of more complex analysis processes based on the *Semantic Pattern* model.

3. **Improvements for machine learning algorithms**: Although the *Semantic Pattern Transformation* comes with many advantages in terms of application, and analyzing and interpreting the gained results, there remains an important question: How good is the quality of the results gained by applying analysis processes to the *Semantic Patterns*? In order to answer this question, an in-depth evaluation was conducted. This evaluation compares the results of supervised classifiers, unsupervised clustering algorithms, and semantic-ware search algorithms, when applied to the commonly used feature vector representation and to the transformed *Semantic Patterns*. By looking at these results, the following conclusions can be drawn: **First**, the *Semantic Pattern Transformation* can be safely

deployed without making any compromises in terms of result quality, while gaining all the advantages in simplifying the deployment in heterogeneous domains and simple knowledge extraction procedures. **Second**, the evaluation shows that huge quality improvements can be made for simple algorithms, such as K-Means. **Finally**, the application of semantic-aware search algorithms is not possible for the standard value-centric feature vector representation without applying further processing. Thus, the capability to execute such algorithms is an additional advantage of the *Semantic Patterns*.

While the details regarding the evaluation procedure and the gained results will be discussed in *Chapter 8 – Evaluation*, an executive overview is already given here: The analyzed data sets, algorithms and the gained results are presented in Table 1.1. The data sets are taken from the UCI Machine Learning Repository [28] and represent heterogeneous knowledge discovery domains. For the evaluation of the results the V-Measure [69] was used. In the table, columns 1 to 6 describe the details of the analyzed data sets. The remaining columns describe the results gained by the application of the supervised Support Vector Machine (SVM) algorithm, and the unsupervised clustering algorithms K-Means and EM. The algorithms have been applied to the raw value-centric feature vectors (not normalized, indicated by **NN**), the normed value-centric feature vectors (indicated by **N**), and the *Semantic Patterns* (indicated by **P**). The *Semantic Patterns* have been gained by applying the *Semantic Pattern Transformation*[1] to the raw value-centric feature vectors. The key results are observed when looking at the columns for the K-Means algorithm. Here, the application of the K-Means algorithm to the *Semantic Patterns* yields huge performance gains when compared to its application to the raw feature vectors. For the unsupervised EM algorithm and the supervised SVM algorithm, there are only insignificant performance gains or losses. However, the main purpose of the *Semantic Pattern Transformation* is not to improve the quality of the results, but to simplify the deployment of knowledge discovery processes within heterogeneous domains, and the extraction of knowledge by utilizing the semantic representation. Still, these results are important in showing that the new method can be safely deployed without making any compromises in terms of quality.

1.5.1 List of Publications

In this section, a short overview over the published works, which are relevant for this thesis, will be given. Thereby, the publications will be grouped into two main categories: **First**, the group of publications that have a direct relation to this thesis, and **second**, publications that are considered as prior work, or have

[1] The results were gained by using the parameter-set for the *Semantic Pattern Transformation* as described in the third row of the table. These parameters will be explained in *Chapter 8 – Evaluation*.

Data set	Label	Inst	DF	SF	Classes	SVM (N)	SVM (NN)	SVM (P)	KM (N)	KM (NN)	KM (P)	EM (NN)	EM (P)
						SVM			K-Means			EM	
						SP-Parameters: D=0.5, Comb=E, Norm=L, MDL=1.5, σ = 0.2							
		Categorical											
Breast Cancer	BC	286		9	2	0.03	**0.04**	**0.04**	0.01	0.01	**0.06**	0.00	**0.08**
Dermatology	DE	366	1	33	6	0.93	0.92	**0.95**	0.58	0.09	**0.86**	**0.87**	**0.87**
KR vs. KP	KR	3196		36	2	**0.75**	**0.75**	0.72	0.00	**0.01**	0.00	**0.04**	0.00
Lymph	LY	148		18	4	**0.53**	0.51	0.48	0.13	0.18	**0.25**	0.26	**0.27**
Mushroom	MU	8124		22	2	**1.00**	**1.00**	**1.00**	**0.48**	0.47	0.45	**0.61**	0.59
Soybean	SO	683		35	19	0.92	0.92	**0.93**	0.59	0.62	**0.73**	**0.79**	**0.79**
Splice	SP	3190		60	3	0.71	0.72	**0.80**	0.03	0.03	**0.44**	**0.41**	0.31
Vote	VO	435		16	2	**0.76**	0.74	0.67	0.47	**0.48**	0.47	**0.49**	0.45
Zoo	ZO	101		17	7	0.94	0.94	**0.97**	0.78	0.78	**0.82**	0.82	**0.85**
		Total				**0.73**	**0.73**	**0.73**	0.34	0.30	**0.45**	**0.48**	0.47
		Mixed											
Anneal	AN	898	6	32	6	0.86	0.86	**0.92**	0.23	0.03	**0.30**	0.31	**0.32**
Colic	CO	368	7	15	2	0.31	**0.32**	0.31	**0.13**	0.03	0.05	0.10	**0.12**
Credit-A	CA	689	6	9	2	**0.41**	**0.41**	0.39	0.16	0.02	**0.25**	0.17	**0.21**
Credit-G	CG	1000	7	13	2	0.11	0.10	**0.12**	**0.01**	**0.01**	0.00	0.01	**0.02**
Heart-C	HC	303	6	7	5	**0.36**	**0.36**	0.29	0.24	0.01	**0.36**	**0.31**	0.28
Heart-H	HH	294	6	7	5	0.32	0.31	**0.33**	0.27	0.01	**0.32**	**0.28**	0.25
Hepatitis	HE	155	5	14	2	0.25	**0.28**	0.21	0.13	0.00	**0.21**	0.22	**0.24**
		Total				0.37	**0.38**	0.37	0.17	0.02	**0.21**	0.20	**0.20**
		Numerical											
Breast-w	BW	699	9		2	**0.78**	**0.78**	0.77	0.73	0.74	**0.82**	**0.72**	0.58
Diabetes	DI	768	8		2	**0.18**	**0.18**	0.15	0.05	0.03	**0.10**	**0.10**	0.08
Glass	GL	214	9		7	0.30	0.30	**0.50**	0.34	**0.39**	0.33	**0.37**	0.36
Heart-Statlog	HS	270	13		2	**0.36**	**0.36**	0.37	0.25	0.02	**0.39**	**0.29**	0.27
Ionosphere	IO	351	34		2	0.48	0.48	**0.50**	0.12	0.12	**0.16**	**0.25**	**0.25**
Iris	IR	150	4		3	0.87	0.87	**0.87**	0.71	0.71	**0.75**	**0.81**	0.78
Segment	SE	2310	19		7	0.88	0.88	**0.90**	**0.61**	0.53	0.59	**0.62**	0.60
Sonar	SO	208	60		2	0.23	0.23	**0.23**	0.01	0.01	**0.02**	**0.01**	**0.01**
Vehicle	VE	846	18		4	**0.51**	**0.51**	0.48	0.11	**0.19**	**0.19**	0.10	**0.19**
Vowel	VO	990	10	3	11	0.63	0.63	**0.76**	0.06	**0.34**	0.23	0.19	**0.25**
		Total				0.52	0.52	**0.55**	0.30	0.31	**0.36**	**0.35**	0.34

Table 1.1: Summary of the V-Measure evaluation results.

an indirect relation to this thesis. For a detailed description of these works and their purpose within the context of this thesis, the reader is referred to *Chapter 10 – Applications*:

The following publications are directly related to this thesis:

- **2009**
  - [84] *Automated Analysis of e-Participation Data by Utilizing Associative Networks, Spreading Activation and Unsupervised Learning*
- **2010**
  - [83] *Event Correlation on the Basis of Activation Patterns*
  - [81] *From NLP (Natural Language Processing) to MLP (Machine Language Processing)*

  - [50] *User Tracking Based on Behavioral Fingerprints* (as co-author)
  - [79] *RDF Data Analysis with Activation Patterns*

- **2011**
  - [80] *Knowledge Extraction from RDF Data with Activation Patterns*
  - [78] *Android Market Analysis With Activation Patterns*
  - [77] *Extracting Semantic Knowledge From Twitter*

The following publications are considered as prior work, or are not directly related to the thesis, but to the publications listed above.

- [64] *Hybrid Engine for Polymorphic Shellcode Detection* (as co-author)
- [62] *Traffic Classification Using Self-Organizing Maps* (as co-author)
- [63] *Massive Data Mining for Polymorphic Code Detection* (as co-author)
- [46] *WiFi Chipset Fingerprinting* (as co-author)
- [82] *InFeCT - Network Traffic Classification*
- [60] *Android Security Permissions – Can we trust them?* (as co-author)

1.6 Demonstration Data Sets

For many of the explanations within this thesis, short examples will be provided that highlight certain aspects or allow to gain a quick overview of the discussed subject. Thereby, these examples and also other discussions are based on two data sets that are described in the appendix of this thesis. The first one covers information about the world's countries, while the second one contains Tweets extracted from Twitter during the Egyptian revolution in early 2011.

2

Semantic Patterns – At a Glance

The aim of a typical knowledge discovery process is to extract knowledge from a data set comprised of arbitrary information relevant for a certain domain. Thereby, the term knowledge is not exactly defined and strongly depends on the analyzed data and the desired information. Especially, when machine learning is used for knowledge extraction, the data needs to be organized as instances of objects that are described via certain properties. These instances are extracted from the data set according to a desired relation. A simple example would be a data set that contains instances of the world's countries, which are described via various features (properties), such as the size of the population, the country's birth rate, its export commodities or unemployment rate. Here, the most obvious relation would extract one instance for each country. However, other relations that extract instances describing continents or single export goods would also be possible.

When applying machine learning the extracted instances typically need to be modeled as high-dimensional feature vectors, where each dimension represents a feature and contains the value of this feature. Depending on the desired knowledge and the nature of the describing features, an adequate machine learning algorithm needs to be selected, and the raw feature vectors must be transformed according to algorithm-specific preprocessing steps. These steps include the normalization of feature values, the handling of missing values, and the transformation of symbolic into numerical features or vice versa. The model of the selected machine learning algorithm is then trained by applying the algorithm to the preprocessed feature vectors. Finally, the desired knowledge is extracted from the trained model. Unfortunately, the interpretation of these algorithm-specific and often complex models is in most cases a tedious task, which needs to be adapted to the domain-specific setup.

In summary, there are three main issues when applying machine learning algorithms to heterogeneous knowledge mining tasks: **First**, the requirement to select a specific machine learning algorithm according to the analyzed data, **second** the specific adaptation of the preprocessing steps required for this algorithm and **third**, the algorithm-specific interpretation of the trained models.

These three principle issues limit the reusability of machine learning algorithms in knowledge discovery processes due to the time-consuming setup and adaptation process. Based on these issues and my own experiences gained during the application of machine learning in heterogeneous knowledge discovery domains, I developed the *Semantic Pattern Transformation*, which transforms the raw feature vectors into so called *Semantic Patterns*. The remainder of this chapter gives an overview on the idea behind the transformation, highlights the employed techniques and demonstrates the benefits by analyzing a simple data set.

2.1 Demonstration Data Set

For the discussions in this chapter a simple artificial data set is used, which contains eight instances of countries listed in Table 2.2. Thereby, a country is described via a feature vector that forms the basis for the application of machine learning algorithms. The features and their possible feature values for this example are presented in Table 2.1.

Feature	Type	Feature values
Export commodity	Symbolic	coffee, cacao, machinery, chemicals
Unemployment rate	Numerical	5% to 20%
Fertility rate (average number of children per woman)	Numerical	2 to 5

Table 2.1: The three features and their feature values used within the data set: export commodity (symbolic), unemployment rate (numerical), and fertility rate (numerical).

The first feature is of symbolic nature and describes the export commodities of a country with four possible feature values: *coffee*, *cacao*, *machinery* and *chemicals*. Thereby, a country might have none, one or multiple export commodities[1]. The second feature is of numerical nature and describes the unemployment rate of a country. Here, the possible value range is from *5%* to *20%*. The final feature is also of numerical nature and describes the fertility rate, which corresponds to the number of births per woman. The value range for this feature is *2* to *5*.

[1] In a real world scenario the fact that no commodities are exported by a country would be modeled via a distinct value (e.g., *none*), because the non-presence of feature values could also mean that these values are missing.

Country	Exports	Unemployment rate	Fertility rate
C1	coffee	20%	5
C2	cacao	20%	5
C3	coffee, cacao	20%	5
C4	machinery	5%	2
C5	chemicals	5%	2
C6	chemicals, machinery	5%	2
C7	chemicals, cacao	20%	**missing data**
C8	**missing data**	20%	5
C9	coffee, cacao	**missing data**	**missing data**

Table 2.2: First category: C1 to C3 – developing countries as indicated by different export commodities and high unemployment and fertility rates. Second category: C4 to C6 – highly developed countries as indicated by the export commodities, and low unemployment and fertility rates. The countries C7 to C9 belong to one of the two categories, but contain missing feature values.

When looking at the demonstration data set in Table 2.2, one observes that there are basically two categories of countries. The first one includes the countries C1 to C3 that have *cacao* and/or *coffee* as export goods and high unemployment and fertility rates. The second category consists of the countries C4 to C6, which have *chemicals* and/or *machinery* as export goods and low unemployment and fertility rates. The last three countries, C6 to C9 belong to one of these categories, but have missing values, which will help to explain the nature of the *Semantic Patterns*. Furthermore, country C7 creates a link between the two categories by having export commodities from both of them – *cacao* and *chemicals*. This represents the fuzzy nature of data sets, which is addressed by the application of machine learning algorithms.

2.2 From Feature Vectors to *Semantic Patterns*

The *Semantic Pattern Transformation* introduced in this thesis addresses the three main problems stated in the introduction of this chapter. **First**, the problem of the domain and knowledge-specific selection of a machine learning algorithm is avoided by transforming arbitrary data into a generic semantic representation. **Second**, the semantic nature of the transformation removes the need for many preprocessing operations including normalization, handling of missing values, or data transformation. **Finally**, in contrast to the high-dimensional feature vectors the generated *Semantic Patterns* can easily be interpreted, which shifts the algorithm-specific task to interpret the model to the direct interpretation of the *Semantic Patterns*. The advantage of this process is that it does not depend on the employed algorithm and the nature of the analyzed data.

These three major benefits are enabled by the shift from a value-centric representation as employed by the raw feature vectors, to a semantic representation

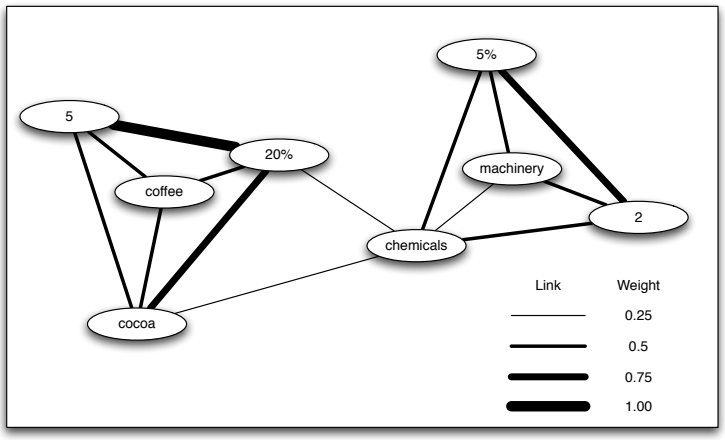

Figure 2.1: The trained associative network consists of nodes that represent the feature values of the analyzed instances, and weighted links between these nodes which model the semantic relations between these feature values. The weights of the links must be normed for the subsequent application of a spreading activation algorithm.

that focuses on the relations between different feature values.

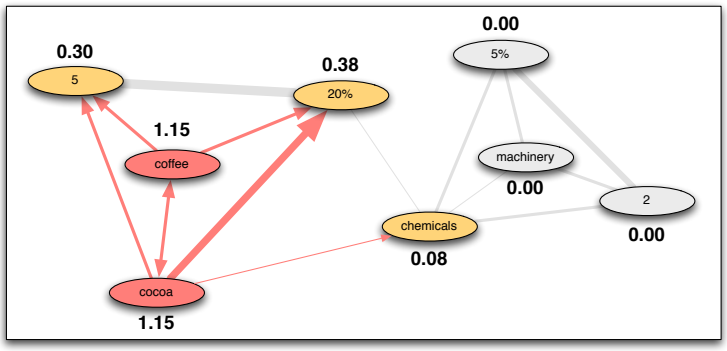

Figure 2.2: The feature values *cacao* and *coffee* of C9 are activated with an initial value of *1.0*. These activations are then distributed to the neighboring nodes according to the weighted links by applying spreading activation techniques. The red arrows indicate the spreading activation direction. The red nodes represent the activated nodes and the yellow ones receive activation energy form the activated (red) nodes by using spreading activation. The grey nodes do not receive activation energy from the activated nodes.

Before explaining the basic idea behind the *Semantic Patterns*, a short introduction on associative networks and spreading activation must be given. An

associative network consists of nodes representing some kind of information, and weighted links that connect these nodes according to an arbitrary relation. The strength of these associations are modeled with weights that are assigned to the respective links. One of the most important methods for extracting information form such a network is the application of spreading activation techniques. Thereby, one or more nodes are activated with a certain activation value that exceeds a certain threshold. This is merely the process of assigning values to the selected nodes. Then, those nodes that have an activation value larger than a pre-defined threshold participate in the following spreading activation process: The activation values, or activation energies of the activated nodes are spread to the neighboring nodes via the weighted links. The amount of the transferred activation energy depends on the link weights, which model the strength of the associations. In the first iteration of this algorithm only the neighboring nodes receive activation energy from the active nodes. However, this process can be repeated for multiple iterations in order to reach more distant regions in the network. By analyzing the size of the activation values the other nodes receive, information about the connections (relations) within the network and, thus, the modeled data can be extracted.

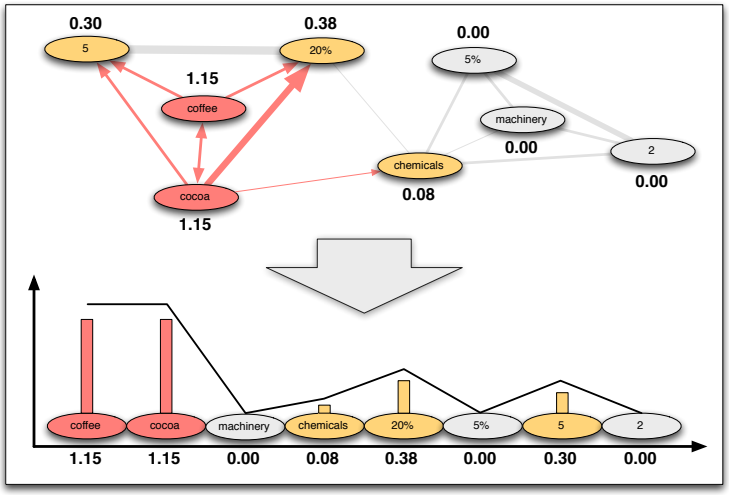

Figure 2.3: After completing the spreading activation process, the activation values of the network nodes are extracted an arranged in a vector – the *Semantic Pattern*.

Based on these techniques, the principle idea behind the layered *Semantic Pattern Transformation* is to represent the feature values as nodes within such an associative network. The semantic relations of these feature values within the analyzed instances are modeled as links between the network nodes. By activating network nodes, and applying spreading activation, the previously stored semantic relations are queried. The resulting node activation values of the whole

network are extracted an arranged in a vector – the *Semantic Pattern*. These patterns, which represent single or multiple feature values, or complete instances form the basis for all subsequent analysis processes and enable the direct interpretation without knowing the specifics of the employed machine learning algorithm.

In order to highlight the principle components of the transformation process, the instances of the country data set are transformed according to the following steps:

First, for each feature value within the data set a node is created within an associative network[2]. **Second**, the semantic relations of these feature values within the data set are modeled with weighted links connecting the network nodes. Here, two feature values are considered to be semantically related when they co-occur within an instance. The strength of such relations is defined by the number of co-occurrences and modeled as weights that are assigned to the corresponding links. The result of this process is a trained associative network, which is depicted in Figure 2.1.

Country	Coffee	Cacao	Machinery	Chemicals	20%	5%	5	2
C1	1.30	0.53	0.00	0.08	1.45	0.00	1.45	0.00
C2	0.45	1.38	0.00	0.15	1.53	0.00	1.45	0.00
C3	1.45	1.53	0.00	0.15	1.68	0.00	1.60	0.00
C4	0.00	0.00	1.30	0.38	0.00	1.38	0.00	1.38
C5	0.00	0.08	0.38	1.30	0.08	1.38	0.00	1.38
C6	0.00	0.08	1.37	1.37	0.08	1.53	0.00	1.53
C7	0.30	1.30	0.08	1.15	1.30	0.15	0.45	0.15
C8	0.30	0.38	0.00	0.08	1.30	0.00	1.30	0.00
C9	1.15	1.15	0.00	0.08	0.38	0.00	0.30	0.00

Table 2.3: The vectors of the *Semantic Patterns* gained by transforming the feature vectors of countries C1 to C9 according to the *Semantic Pattern Transformation*.

Third, for the transformation of an instance represented by a feature vector into a *Semantic Pattern*, the nodes corresponding to the instance's feature values are activated within the network and the spreading activation algorithm is applied for one iteration. Thereby, the activation values of the activated nodes are spread to semantically related (neighboring) nodes according to the strength of the relations. This process is depicted in Figure 2.2, where the nodes corresponding to the feature values *coffee* and *cacao* of the country instance C9 are activated with an initial value of 1.0. By applying spreading activation these activations are spread to the neighboring nodes, which is indicated by the red arrows. After completing the process, the activation values of all network nodes are extracted and stored in a vector – the *Semantic Pattern*. The extraction

[2] For numerical features, some kind of discretization operation needs to be applied. However, for the sake of simplicity this is not shown in this example, but discussed in detail in later chapters.

process and the structure of a *Semantic Pattern* is visualized in Figure 2.3. The process is repeated for all instances, which completes the transformation of raw feature vectors into *Semantic Patterns*. The generated patterns are listed in Table 2.3 and the contribution of the activation values to an instance is visualized in Figure 2.4.

2.3 Understanding *Semantic Patterns*

The transformed *Semantic Patterns* listed in Table 2.3 represent the information gained by selecting and activating nodes within the associative network and spreading their activation to the related nodes. In this section an overview of the pattern structure is given, and several basic interpretation and analysis techniques are discussed.

2.3.1 Semantic Relations

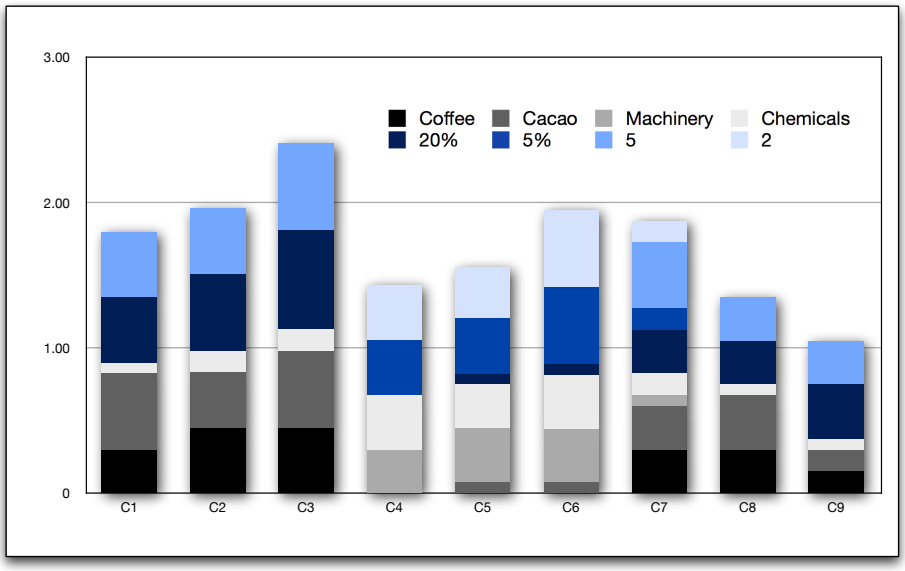

Figure 2.4: The contribution of the node activation values to the country instances.

After completing the transformation process, each instance feature vector has been transformed to a *Semantic Pattern*. However, the transformation process is not limited to complete instances but can also be applied to single feature values. In order to show how *Semantic Patterns* can be analyzed and interpreted, three patterns are created for the two export commodities *cacao* and *coffee*, and the fertility rate *2*. These patterns are visualized in Figures 2.5 and 2.6. While the first figure contains the unmodified patterns, they have been sorted

according to the activation values or the represented feature values in the latter. Especially in real applications, were high-dimensional patterns are involved this representation can be used to gain a quick overview on the principle information stored within the patterns. In both representations, the x-axis is used for the nodes representing the feature values and the y-axis contains the node activation values.

When looking at the pattern for *cacao* in Figure 2.5, one observes that the export commodity *cacao* has the highest activation value, because its node was activated during the pattern transformation process. Furthermore, there are also strong activations for the unemployment rate *20%*, fertility rate *5* and the export commodity *coffee*. The relation to *chemicals* is weaker and there is no relation to *machinery*, the unemployment rate *5%* and the fertility rate *2*. The magnitude of these values can easily be verified by looking at Table 2.2 and the associative network in Figure 2.1. Similar information can be extracted from the patterns for *coffee* and the fertility rate *2*.

This interpretation enables the analysis of the semantic relations between the feature values, which reveals important information about the analyzed data. Furthermore, this example shows that *a single feature value is represented by its semantic relations to other feature values* and not its value alone. This is a significant difference to the standard feature vector representation.

2.3.2 Similarity

By looking at the unsorted patterns in Figure 2.5, one observes that there is a similarity between *coffee* and *cacao*. This is explained by the similar network regions that are active in both patterns. In contrast, the pattern for the fertility rate *2* is not related to *coffee* at all, and there is only a insignificant similarity to *cacao*, because the node for *chemicals* has a small activation in both patterns. Since *Semantic Patterns* are simple vectors, a similarity measure, such as the Euclidean distance or the Cosine similarity can be used to determine their similarity. In case of *coffee* and fertility rate *2*, the Cosine similarity would yield the maximum distance, because the vectors are orthogonal due the distinct activated network regions.

2.3.3 Semantic Search

Based on the similarity between *Semantic Patterns* and the semantic information contained in the patterns, semantic-aware search queries can be implemented. Compared to algorithms based on simple keyword matching, such algorithms are able to retrieve more relevant results, which is demonstrated with the following query on the demonstration data set:

Assuming, a simple keyword matching search query is executed that retrieves countries related to the feature value *coffee*, then the results would only include those countries that use *coffee* as export commodity. These countries – C1, C3 and C9 – are listed in the upper part of Table 2.4.

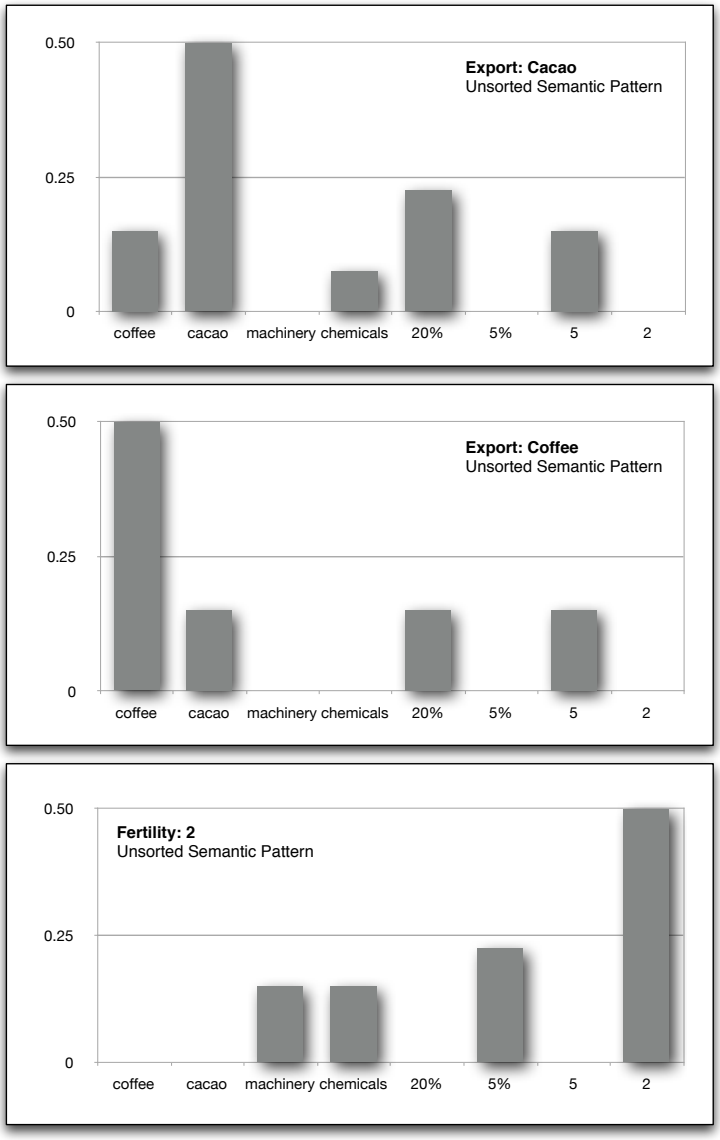

Figure 2.5: The *Semantic Patterns* for the export commodities *cacao* and *coffee*, and the fertility rate *2*. The x-axis represents the nodes of the associative network, whereas the y-axis represents the activation values of these nodes. For visualization purposes the maximum value of the y-axis has been limited to *0.5*.

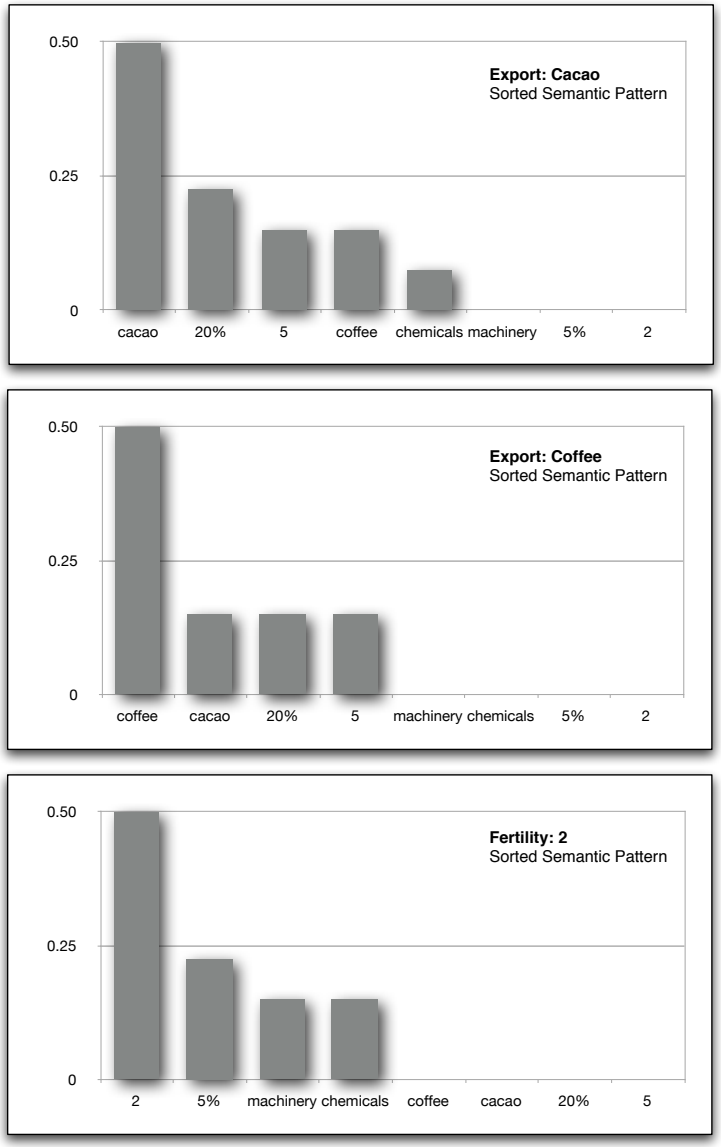

Figure 2.6: The nodes of the *Semantic Patterns* from Figure 2.5 are sorted according to their activation values. For visualization purposes the maximum value of the y-axis has been limited to *0.5*.

However, when utilizing the semantic nature of *Semantic Patterns* for the search algorithm, more relevant results can be achieved. The algorithm is based on these two steps: **First**, a *Semantic Pattern* is generated for the feature value *coffee* and **second**, this pattern is compared to the *Semantic Patterns* of the country instances. Obviously those countries that have *coffee* as export commodity are the three best matching results. However, the fourth result is country C4, which exports *cacao*. Although *coffee* is not exported by C4, there is a significant semantic similarity between C4 and *coffee*. By looking at the *Semantic Pattern* of *coffee* in Figure 2.6 and the descriptions of the countries in Table 2.2, two observations can be made: **First**, *coffee* and *cacao* are related due to the co-occurrences in C3 and C9. **Second**, *coffee* has a strong relation to high fertility and unemployment rates as defined by the co-occurrences in C1 and C3. Since this relation information is stored in the *Semantic Pattern*, the search algorithm is also able to retrieve those countries that are semantically related to *coffee* but do not export this commodity themselves. The countries C8 and C7, which are retrieved as the fifth and sixth result contain missing values, but are still have some similarity to the search query due to other features that have strong relations to *coffee*. The last three results – countries C5, C6 and C4, have the least similarity to the *Semantic Pattern* of *coffee*. Thereby, C5 and C6, which export *chemicals* and have low fertility and unemployment rates, are still retrieved before C4 due to the relation defined between *cacao* and *chemicals* in C7. As *cacao* is quite similar to *coffee* (Figure 2.6) there is a slightly higher similarity to C5 and C6 than to the last result C4, which has a *Semantic Pattern* that is almost orthogonal to *coffee*.

Apart from the possibility to apply semantic-aware search algorithms, this example shows that *Semantic Patterns* are able to cope with missing values. This is best indicated by the retrieval of C8 which does not contain any information about export commodities but is related to the search query for *coffee* due to high unemployment and fertility rates.

2.3.4 *Semantic Pattern* Arithmetic

Due to the vector representation of the *Semantic Patterns*, standard vector arithmetic can be applied to modify patterns, extract information or generate new patterns:

Addition: The semantic information contained in multiple *Semantic Patterns* can be combined by adding the underlying vectors. The resulting pattern is then a mixture of the input patterns. In the demonstration data set a *coffee-cacao* pattern can be generated by adding the patterns for these two export commodities.

Subtraction: By subtracting a *Semantic Pattern* from another one, calculating the absolute values of the resulting activation values and sorting the *Semantic Pattern* elements according to the activation values, the main differences between the two input patterns can be extracted. As example, this procedure is applied to the patterns for *coffee* and *cacao* from the demonstration data set. The results are shown in Figure 2.7.

Keyword search for *coffee*			
C1	coffee	20%	5
C3	coffee, cacao	20%	5
C9	coffee, cacao	**missing data**	**missing data**
Semantic aware search for *coffee*			
C9	coffee, cacao	**missing data**	**missing data**
C1	coffee	20%	5
C3	coffee, cacao	20%	5
C2	cacao	20%	5
C8	**missing data**	20%	5
C7	chemicals, cacao	20%	**missing data**
C5	chemicals	5%	2
C6	chemicals, machinery	5%	2
C4	machinery	5%	2

Table 2.4: This table compares the results when searching for *coffee* with a keyword matching search algorithm (upper part) and a semantic-aware search algorithm based on the similarity between *Semantic Patterns* (lower part).

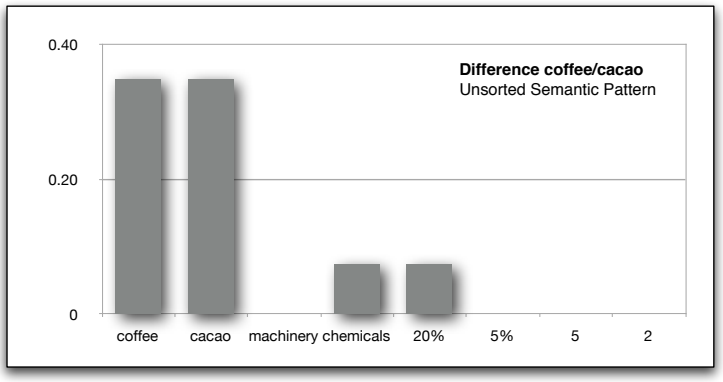

Figure 2.7: Differences between the patterns for *cacao* and *coffee*. The large activation values for *cacao* and *coffee* are related to the significant influence of the initial activation values.

Mean: By calculating the mean pattern for multiple input patterns or complete pattern sets, a quick overview of the information contained in these patterns is gained. An example for such a pattern set could be a pattern cluster as found by an unsupervised learning algorithm. In Figure 2.8, two mean patterns for the two country categories C1 to C3 and C4 to C6 are presented. Thereby, the activated nodes and the size of their activation values give a quick overview about the principle properties of the two categories.

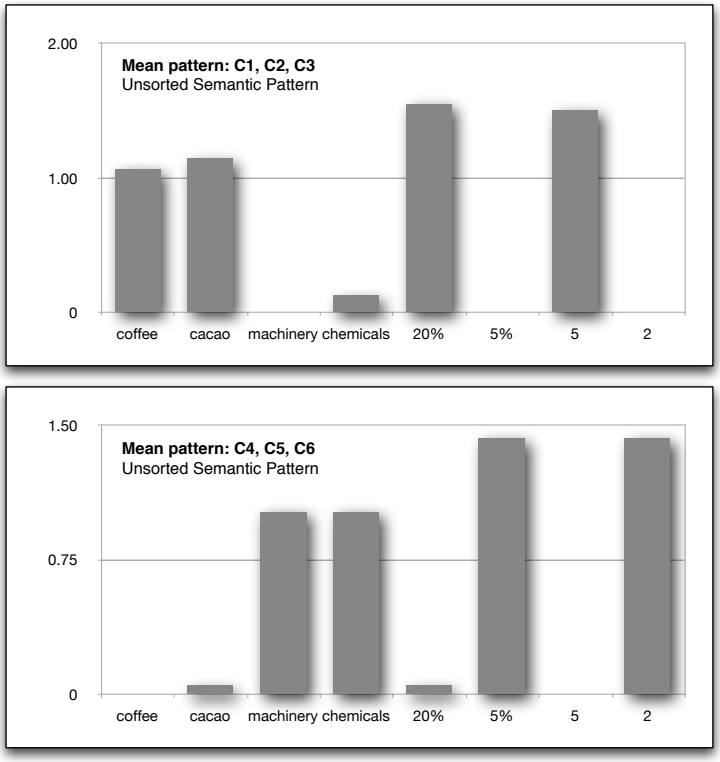

Figure 2.8: Mean *Semantic Patterns* for C1, C2, C3 (upper half) and C4, C5, C6 (lower half).

Variance: By calculating the variance of multiple patterns in a pattern set, a quick overview of the similarities and differences within this set is gained.

These are only some examples on how simple vector arithmetic can be used for the processing and interpretation of *Semantic Patterns*. However, any other simple operation or more sophisticated operation, such as machine learning can be applied.

It is also important to note, that regardless of the applied operation, one common model is maintained as basis for any subsequent analysis.

2.3.5 Machine Learning

The main purpose of the *Semantic Pattern Transformation* is the creation of one generic model that forms the basis for the application of machine learning algorithms. The transformation into *Semantic Patterns* removes many of the time-consuming steps that are typically necessary when employing such algorithms in heterogeneous domains.

Since *Semantic Patterns* only contain numerical values – regardless of the input data – a wide range of machine learning algorithms can be directly applied. One interesting application is in the area of unsupervised learning, where a priori knowledge about the analyzed data is typically not available.

2.4 Feature Vectors vs. *Semantic Patterns* in Unsupervised Clustering

Unsupervised clustering plays an important role within machine learning, since a good overview can be gained about the principal structure of the analyzed data set. This is especially important when only limited or no a priori knowledge about the data is available. In a typical knowledge discovery process, the application of such an algorithm requires several steps that need to be adapted according to the available data and the desired information. These steps include

- the definition of a relation that extracts the desired instances and features describing those instances,
- the selection of an algorithm capable of handling the available data,
- finding an appropriate representation of the data that can be used by the algorithm,
- applying various preprocessing steps depending on the algorithm, the available data and the desired information,
- applying the selected algorithm to the preprocessed data,
- and interpreting the results.

The remainder of this section gives an overview of these steps and demonstrates how many of them can be simplified or removed by applying the *Semantic Pattern Transformation*.

2.4.1 Extracting Instances and Features

The first step includes the definition of a relation that extracts the instances and the features describing those instances. Apart from the process of selecting the appropriate features, it is also possible to construct new features by applying mathematical operations on the existing feature values, or using external information to augment the available data. Within the demonstration data set,

the country instances are described by the three features shown in Table 2.1. Since all of these feature are taken for the analysis, the extraction process is not required here. Furthermore, no additional features are constructed, nor is the available information augmented by external data.

These extraction steps are also required when employing the *Semantic Pattern Transformation*. However, due to the easy interpretation of the generated *Semantic Patterns*, additional information about the extracted features is gained. This information could then be used to modify the extraction process, place the emphasis on certain aspects or ignore non-relevant features.

2.4.2 Selection of the Algorithm

Feature vectors: There is a wide range of unsupervised clustering algorithms that use heterogeneous models and techniques to analyze instances. Thereby, those algorithms have different capabilities regarding the processing of symbolic or numerical data, require different preprocessing steps and employ different models that are interpreted differently. Therefore, a careful choice must be made according to the nature of the features and the desired information that should be extracted from the data.
Semantic Patterns: The *Semantic Pattern Transformation* transforms arbitrary feature vectors comprised of symbolic and/or numerical data into a generic semantic representation. In this representation a *Semantic Pattern* contains the activation values of the associative network nodes, which are numerical values. Therefore, any unsupervised algorithm capable of handling numerical feature values can be chosen. Since the interpretation of the results can be shifted from the algorithm model to the *Semantic Patterns*, the algorithm does not need to be chosen according to the interpretability of its model.

2.4.3 Instance and Feature Representation

For the further discussions an unsupervised algorithm is assumed, which uses the Euclidean distance measure for determining the similarity of the instances. Examples for such algorithms are the K-Means algorithm, the Neural Gas algorithm family, Self-Organizing Maps, or the Expectation Maximization (EM) algorithm.

Feature vectors: Since the model of the chosen algorithm depends on the calculation of the Euclidean distance, one needs to make sure that the employed feature vectors use an adequate structure. For the two numerical features – unemployment and fertility rate – this representation is simple, because the values are directly stored as elements within the feature vector. However, for the symbolic export commodities feature this process is not straightforward for two reasons: **First**, the number of export commodities varies from country to country and **second**, the symbolic values cannot be directly used for the distance calculation. Thus, some kind of transformation process must be applied to these feature values. A simple approach would assign numerical values to each export commodity. However, this would cause two problems: **First**, the variation in the

number of exported commodities cannot me modeled by this approach. **Second**, although, the numerical values allow the application of distance measures, the gained information would be meaningless: The question whether *cacao* is closer to *coffee* than to *machinery* cannot be answered without any additional information. A typical method to overcome these problems is the creation of an entry for each symbolic feature value and storing whether the feature value is present in the given instance. This is depicted in Table 2.5, where the two numerical features are directly stored in the feature vector and the export commodity feature was transformed according to the described procedure.

Country	Coffee	Cacao	Machinery	Chemicals	Unemployment rate	Fertility rate
C1	1	0	0	0	20%	5
C2	0	1	0	0	20%	5
C3	1	1	0	0	20%	5
C4	0	0	1	0	5%	2
C5	0	0	0	1	5%	2
C6	0	0	1	1	5%	2
C7	0	1	0	1	20%	**missing data**
C8	**missing data**				20%	5
C9	1	1	0	0	**missing data**	**missing data**

Table 2.5: The value-centric feature vectors representing the country instances.

Semantic Patterns: For the *Semantic Pattern Transformation* the feature vector representation depicted in Table 2.5 contains all the required information. However, for the creation of the network the ordered structure of the features and feature values is not required. It is sufficient to have all feature values and their feature listed for each instance, which is shown in Table 2.6. The instances are then transformed to *Semantic Patterns* according to the previously described process.

Country	Feature values
C1	E:coffee, UR:20%, FR:5
C2	E:cacao, UR:20%, FR:5
C3	E:cacao, E:coffee, UR:20%, FR:5
C4	E:machinery, UR:5%, FR:2
C5	E:chemicals, UR:5%, FR:2
C6	E:chemicals, E:machinery, UR:5%, FR:2
C7	E:chemicals, E:cacao, UR:20%
C8	UR:20%, FR:5
C9	E:coffee, E:cacao

Table 2.6: The unstructured input representation for the *Semantic Pattern Transformation*. The features are abbreviated as follows: *E*: export commodity, *UR*: unemployment rate, and *FR*: fertility rate.

2.4.4 Preprocessing

Normalization

Feature vectors: The application of an unsupervised learning algorithm based on the Euclidean distance requires the normalization of these vectors. The reason is highlighted by looking at the unemployment and fertility rate features. There are two categories of countries in the data set – one with a low unemployment rate with *5%* and the other one with *20%*. For the fertility rate, there are groups with *2* and *5* children per woman. In this example, there is a strong correlation between a high unemployment and a high fertility rate. By only taking these two features into consideration, the Euclidean distance between the countries C1 and C5 can be calculated in the following way: $d^2 = (20-5)^2 + (5-2)^2 = 15^2 + 3^2 = 225 + 9 = 234$. This calculation reveals that the distance between the two unemployment rates contributes a significant part to the overall distance (225 vs. 9).

The unemployment rate is a good indicator for the economic properties of a country, and high values like *20%* typically come along with other properties, such as certain export commodities, or the size of the agricultural, industrial and service sectors. In a similar way the fertility rates of industrial countries are significantly lower than those of developing countries. However, the information carried by the same feature value ranges is quite different in both features. The *2%* difference between unemployment rates of *5%* and *7%* does not carry much information, but the same difference – in terms of the value – between a fertility rate of *2* and *4* children per woman is very significant. For other combinations of features, such as the population count and the birth rate, there is a much larger deviation of the value ranges. Since the Euclidean distance does not consider the different amount of information carried by the value ranges of the features, an appropriate normalization technique must be applied. This choice depends on the analyzed data, but typically the main purpose is to distribute the significance of the features' value ranges equally.

The deviation of the value ranges can also be observed in the demonstration data set: *0* to *1* for the export commodities, *5* to *20* for the unemployment rate and *2* to *5* for the fertility rate. Thus, the feature vectors must be normed before applying the unsupervised clustering algorithm.

Semantic Patterns: Normalization of the input feature vectors is not required, because the *Semantic Pattern Transformation* analyzes the relations between the feature values and not the values themselves.

Missing Values

Feature vectors: The missing values within the instances also need to be considered when applying the machine learning algorithm. In this example the algorithm is based on the Euclidian distance. A simple method would ignore the features with missing values when calculating the distance. Assuming the distance between C6 and C7 is calculated, then the missing fertility value in C7 would cause the removal of the fertility feature for the calculation.

Semantic Patterns: The handling of missing values is not required, since they are simple ignored during the creation of the associative network. In contrast to the raw feature vector representation, a feature value in a *Semantic Pattern* is represented by the semantic relations to other feature values. Thus, when calculating the similarity of the two countries C6 and C7, there is not need to exclude the fertility rate feature as in the raw feature vector representation: **First**, the other feature values of C7 contribute their information about the relations to the missing fertility rate value and **second**, in C6 the fertility rate feature also provides information about the other features in C6. This is a significant difference to the similarity calculation of the raw feature vectors where the fertility rate of C6 was also dropped. The additional semantic information also provides an interesting opportunity for the application of semantic-aware search algorithms.

2.4.5 Applying the Algorithm

Feature vectors: The algorithm is applied to the preprocessed feature vectors, which are shown in Table 2.5.

Semantic Patterns: The algorithm is applied to the *Semantic Patterns* shown in Table 2.3. In this case the dimensionality of the *Semantic Patterns* is equal to those of the feature vectors. However, in general when numerical values are used the number of dimensions is higher than that of the value-centric feature vectors[3].

2.4.6 Interpretation of the Results

After applying the unsupervised algorithm, the desired information can be extracted via the just found feature vector clusters and/or the trained model of the algorithm. Depending on the employed algorithm the following information can be extracted from the model: the cluster quality, the quantization errors made by the cluster prototypes, the complexity of the model, information about outliers, or the trained data in general. The nature and availability of this information depends on the techniques the machine learning algorithm is based on. Whereas some algorithm models allow the direct extracting of certain properties, others require the application of further processing in order to gain the desired information. The variation in the algorithm model structure is an obstacle when analyzing data from heterogeneous domains. In order to avoid the model-specific setups and adaptations, information can also be extracted by interpreting the feature vectors or the *Semantic Patterns*, which are organized according to the partitions created by the machine learning algorithm.

Feature vectors: When analyzing the feature vectors directly, several issues need to be taken into consideration:

[3]In this example the numerical values were directly represented by the network nodes. This was only possible because of the low number of feature values used by these features. Typically, additional discretization operations would be required for mapping the numerical values to network nodes.

- The feature vectors represent the data after applying the required preprocessing steps. However, in contrast to the *Semantic Patterns* the preprocessed data does not contain any additional information like semantic relations when compared to the raw data.
- Typically some normalization procedure was applied to the feature vectors prior to the application of the machine learning algorithm. Thus, any information extracted from the gained feature vector partition sets must always be mapped back to the raw data contained in the feature vectors and vice versa.
- The raw feature vectors must be transformed according to the capabilities of the algorithm during the preprocessing operations. Therefore, the procedure applied for the extraction of information must take the applied transformations into consideration.
- When further analysis processes, such as search algorithms, the extraction of relations within the analyzed data, or any other operation are required, then these operations always need to be adapted to the data stored within the feature vectors and must consider the differences between symbolic and numerical data and possible value ranges.

Semantic Patterns: The interpretation and extraction of information from the *Semantic Patterns* is a simple process that remains the same regardless of the analyzed data. The common model and the simple interpretability are major advantages when compared to the specific analysis required for the interpretation of the feature vectors or the trained models. Furthermore, the transformation into *Semantic Patterns* enables the direct application of additional techniques, such as semantic search algorithms without any additional data-specific processing steps.

2.5 Chapter Conclusions

This chapter gives a high level overview of the *Semantic Pattern Transformation* and the information contained in the *Semantic Patterns*. In summary, the shift form the value-centric to the semantic representation comes with many advantages for preprocessing the data and interpreting the results. The concepts and ideas presented in this chapter will be discussed in detail in the remainder of this thesis.

3

Knowledge Discovery and Machine Learning

Knowledge discovery in databases (KDD) – also known as knowledge mining, or data mining – is described in [24] as the *nontrivial process of identifying valid, novel, potentially useful, and ultimately understandable patterns in data.* KDD is a generic description of a process that is based on multidisciplinary areas such as machine learning, statistics or artificial intelligence (AI), and is nowadays widely deployed in scientific research and industrial processes. Regardless of the utilized techniques, many different steps need to be applied to the raw data before the end product – *knowledge* – is gained.

Especially, machine learning plays an important role within knowledge discovery for a wide range of tasks including the extraction of unknown relations, the unsupervised clustering of data, or the training of a classifier for identifying distinct classes of data. Although the employment of machine learning algorithms yields many benefits, quite a large number of domain and task-specific procedures are necessary that require time-consuming setups. These setups strongly depend on the desired knowledge, the nature of the input data and the specific machine learning algorithms.

In order to understand the steps involved in a generic knowledge discovery application, this chapter introduces two knowledge discovery models from the industrial and academic sectors. The focus will then be placed on the relation between machine learning and knowledge discovery by describing the processing steps involved in the application of a machine learning algorithm. Finally, the deployment of machine learning within knowledge discovery will be discussed and the main obstacles for the utilization in heterogeneous domains will be identified.

3.1 Knowledge Discovery

In this section the principle steps of generic knowledge discovery processes will be explained on the basis of abstract knowledge discovery models. These models provide a framework for the extraction of knowledge from arbitrary data sets. In addition to these models, the knowledge discovery process will be further refined by providing additional definitions that form the basis for the deployment of machine learning algorithms within knowledge discovery.

3.1.1 Knowledge Discovery Models

The knowledge discovery process involves many different processing steps that include the extraction of data, the selection of adequate algorithms, the transformation of this data into a suitable algorithm-specific representation, the application of the algorithm and the final knowledge extraction process. Due to the large variability in the process structure, when used for different domains, abstract models have beed defined that identify common procedures. Although the details of the models depend on the targeted environment, they all address similar problems. In the remainder of this section two important models from academia and industry will be discussed in detail and further references to other relevant models within the literature will be given. For a thorough discussion on the well-known models we recommend reading Chapter 2 of [12] and the work by Kurgan et al. [45].

Model According to Fayyad et al.

According to [12], the model introduced by Fayyad et al. in 1996 [24] is perceived as the leading research model. Thereby, Fayyad et al. describe the generic knowledge discovery process with nine steps that are applied whenever knowledge from a specific domain data set needs to be extracted. These steps are depicted in Figure 3.1 and described as follows:

1. **Developing an understanding of the application domain**: Prior to the application of any algorithm for the extraction of knowledge, the specific domain and its data must be understood. Furthermore, based on the broader aim of the knowledge discovery process applied to the data, the specific goals for the employed algorithms must be defined. This step also includes the integration of prior knowledge, which might be available from other sources.

2. **Creation of a target data set**: Based on the defined goals, a target data set composed of the relevant variables is composed. Typically, this extraction process is related to querying the data base that contains the data set.

3. **Data cleaning and preprocessing**: This step applies the necessary preprocessing steps for transforming the data into a representation that is

suitable for the algorithms applied later. The principle procedures include restructuring the data, applying normalization, handling missing values and noisy data, and applying other transformation operations.

4. **Data reduction and projection**: Depending on the defined goals, specific variables contained in the data set are selected, extracted, or transformed into an adequate representation.

5. **Selection of a data-mining task**: This step matches the goals – defined in Step 1 – to a specific data-mining method such as clustering, training a classifier, applying regression etc.

6. **Selection of a specific data-mining algorithm**: Depending on the more general data-mining method, one of the many available algorithms must be chosen. Among the pre-defined goals, the chosen algorithm depends on the available data, the nature of the desired knowledge mining tasks and the domain-specific environment.

7. **Data-mining**: The chosen data-mining algorithm is applied to the data in order to extract information. The quality of the process and the ability to achieve the previously defined goals strongly depends on the choices made in the previous steps.

8. **Interpretation**: The results gained from the application of the data-mining algorithm need to be interpreted in order to extract the desired knowledge. The methodology used for the knowledge extraction process strongly depends on the specific setup defined in the previous steps, and especially on the models employed by the chosen data-mining algorithms.

9. **Using the discovered knowledge**: After making the appropriate choices and successfully executing the previous steps, the extracted knowledge represents the final product of the knowledge discovery process, or forms the input for further operations.

Due to the well founded establishment of this model within the KDD area, it will be used as a basis for the discussions in the remainder of this chapter.

Six-Step Model by Cios et al.

While the model of Fayyad et al. comes from an academic environment, the five-step CRISP-DM knowledge discovery model [73] was specifically adapted to address the requirements for knowledge discovery processes within industrial environments. In order to combine academic and industrial aspects, hybrid models have also been developed. An example for this category is is the model defined by Cios et al. [61], which is comprised of six principal processing steps that clearly indicate that the model inherited the industrial nature of CRISP-DM:

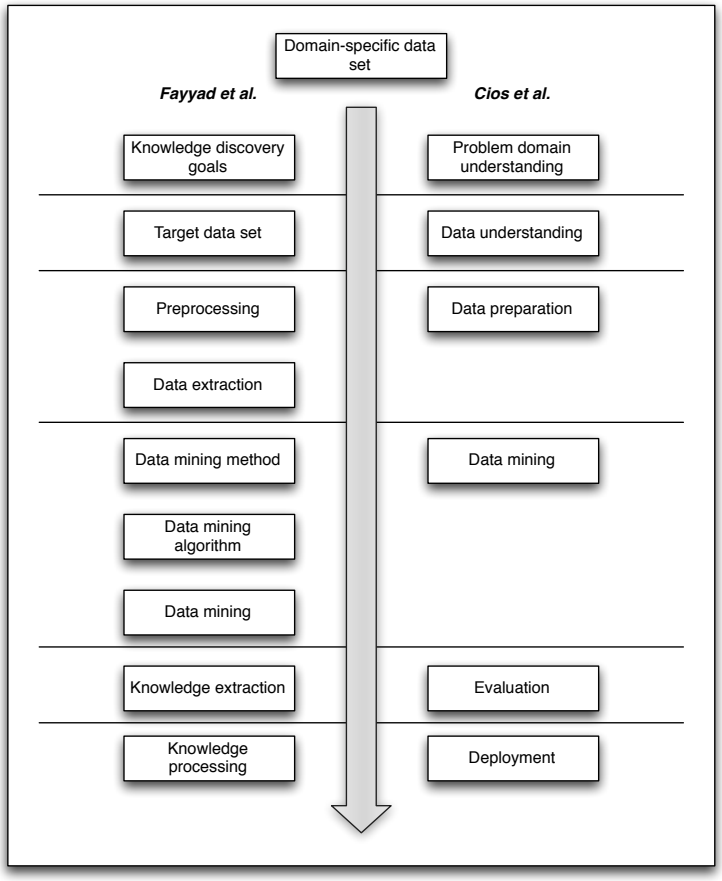

Figure 3.1: The knowledge discovery process defined according to the nine steps of the Fayyad et al. model (left) and according to the six steps of the Cios et al. model (right).

1. **Understanding of the problem domain**: In this initial step of the project, the problem domain is investigated in detail, the key people are identified and project goals are determined and then transformed into knowledge discovery goals.

2. **Understanding of the data**: Here, the appropriate data is selected, checked for various properties such as redundancy, missing values, completeness, and plausibility. After applying these processes, a verification process ensures that the available data is useful for the knowledge discovery goals determined in the previous step.

3. **Preparation of the data**: In this step, the available data is prepared for the actual application of the specific knowledge discovery algorithms

within the next step. The preparation procedures include the application of various preprocessing steps for the selection and extraction of features, the derivation of new features, the reduction of the dimensionality, the cleaning of data, and the application of other procedures required by the employed knowledge discovery algorithms.

4. **Data mining**: The knowledge discovery algorithms, which were selected in the first step, are applied to the prepared data.

5. **Evaluation of the discovered knowledge**: In this step the results of the knowledge discovery algorithms are evaluated. This evaluation process includes the understanding of the results, determining whether the gained knowledge is valuable, the interpretation of this knowledge by domain experts, and finally, understanding the impact of the gained knowledge.

6. **Use of the discovery knowledge**: The final step includes the utilization of the gained knowledge within the defined project and the possible application of the existing setup within other domains.

Other Models

Depending on the environment and the specific purposes various academic, industrial or hybrid knowledge discovery models have been developed. Within the academic category the previously described nine-step model of Fayyad et al. introduced in 1996, and the eight-step model of Anand et al. [2] and [3] introduced in 1998, play the most important role. The CRISP-DM model is a pure industrial standard and together with the academic models it forms the basis for the definition of hybrid models that combine aspects from both environments. The previously described model by Cios et al. [61], and the model introduced by Cabena et al. [10] belong to this last category.

For an in-depth discussion and comparison of these models the reader is referred to [12] and [45].

3.1.2 Knowledge Discovery Processes (KDPs) and Knowledge Discovery Tasks (KDTs)

Based on the knowledge discovery model by Fayyad et al., an additional definition that describes a knowledge discovery task (KDT) is required to allow more generic setups. Typically, when knowledge should be extracted from an arbitrary domain data set, multiple algorithms need to be applied in order to gain the desired information. For each of these algorithms the steps defined within the knowledge discovery model need to be applied. Based on these assumptions the following definitions are made: The application of an arbitrary knowledge mining algorithm and the setup of the necessary process steps according to the knowledge discovery model is defined as Knowledge Discovery Task (further denoted as KDT). The application of one or multiple KDTs to extract the desired

information from an arbitrary domain data set is combined within a Knowledge Discovery Process, or KDP. The complete KDP is visualized in Figure 3.2 and includes the selection of a domain-specific *data set*, the definition of the desired knowledge to be extracted (*Knowledge discovery goals*), the application of multiple *Knowledge Discovery Tasks (KDT)* required for the extraction of information, the subsequent reasoning, and finally the appropriate representation (*Knowledge processing*) of the gained knowledge. Thereby, the KDTs can either be independent or arranged in a processing chain, where the output of a single KDT or multiple KDTs is used as input for another KDT. The final output is the gained *Knowledge*, which depends on the specific domain and the desired information.

In order to avoid the adaptation of the process for each different domain, the employed techniques should be as reusable as possible. However, the setup of the KDTs within a KDP is domain-specific and needs to be adapted whenever the nature of the data changes, even when the knowledge discovery goals remain the same.

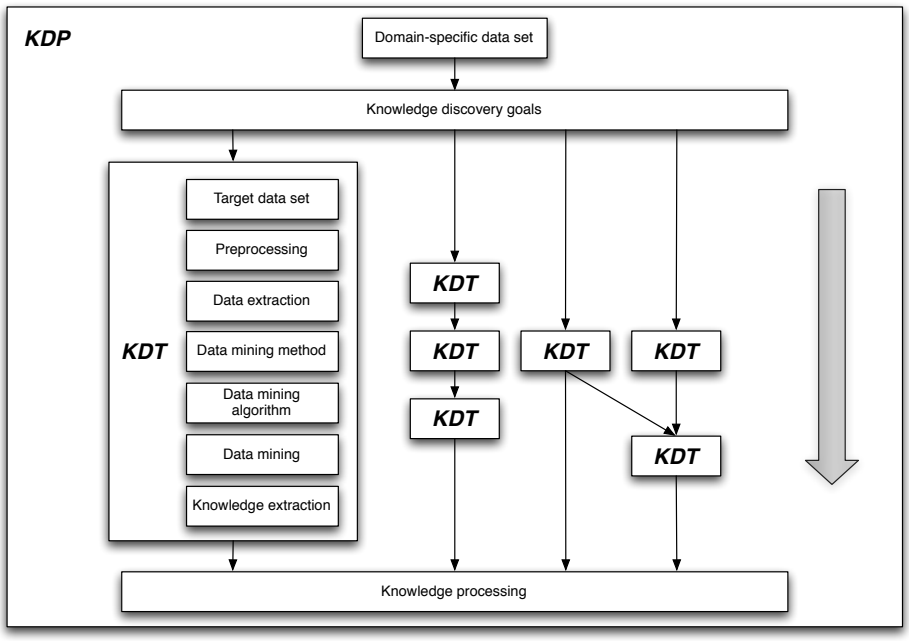

Figure 3.2: A knowledge discovery process (KDP) for a domain-specific data set involves the application of multiple knowledge discovery tasks (KDTs) that may depend on the output of other KDTs.

3.2 Machine Learning

This chapter focuses on the processing steps required for a typical machine learning setup. Thereby, an arbitrary machine learning algorithm will be seen as a black box that takes data as input, analyzes this data by training a model, and yields a certain result that needs to be interpreted for the extraction of knowledge. This superficial view on the algorithms is adequate for the description of the processing steps involved in a machine learning setup. For more details on machine learning and the algorithms utilized in this thesis, the reader is referred to *Chapter 5 – Techniques*.

3.2.1 Data

Prior to the application of a machine learning algorithm, the analyzed data must be transformed into the appropriate representation. For most machine learning algorithms this is the value-centric feature vector representation. Before going into the details of this representation, a closer look at the nature of the data needs to be taken.

Data Set

A domain-specific data set is the basis for the application of all machine learning algorithms and contains data of arbitrary nature. Typically, before the application of an algorithm, certain data needs to be selected, extracted and transformed into a representation that is adequate for the applied machine learning algorithm.

Raw Data or Variables

Before taking a closer look on how the data needs to be prepared prior to the application of machine learning algorithms, the nature of the data itself needs to be discussed. Thereby, the variables, which are used for the description of arbitrary objects, can be assigned to various categories depending on the nature of their corresponding values. In statistics, the differences between these variables play an important role, since they influence the applicability of different algorithms. The same applies to machine learning, where the choice of the algorithm, the applied preprocessing steps, the possible data transformations, and the applied knowledge extraction methods depend on the nature of the data.

The following categorization shows the different categories of variables and explains their differences. There are more detailed categorizations in the literature, but for the scope of this work the subsequent descriptions are sufficient. For a discussion of this topic within a machine learning context we refer to *Part 1 - Chapter 2* of the book *Data Mining* written by Witten et. al [90].

- **Categorical variables**: Variables of this category contain values that are represented by distinct symbols. Such symbols could be names, labels,

characters or even numerical values. The distinct property which clearly separates those variables from numerical ones is not the nature of the values itself, but the fact that the values of such variables cannot be put into a relation via a distance measure. Thereby, two major groups of variables can be defined:

  - **Nominal**: The values of such variables can neither be related via a distance measure nor can they be sorted. When looking at export commodities such as *iron*, *coffee* or *machinery* it is not possible – without taking any other information into account – to say whether *iron* is closer to *coffee* or to *machinery*. In this case it also not possible to rank the values.
  - **Ordinal**: Ordinal variables contain values that cannot be related via a distance measure, but the values themselves reveal their rank within the variable. As example, the ordinal values *low*, *medium* and *high* of a variable that describes the speed of a fan are considered: In this case the values can be ordered but it is not possible to define a meaningful distance between the values themselves: Is the distance between *low* and *medium* smaller than between *medium* and *high*? Also, the addition or subtraction of such values is not possible: Is the result of adding *low* and *medium* equal to *high*?

- **Numerical variables**: The common properties among all numerical values and the further described sub-categories are the rank and the distance properties. The first one, which is also fulfilled by the ordinal variables, means that a feature value itself carries the information on how it is ranked among the other feature values of the given variable. The second property indicates that a meaningful distance can be calculated between the feature values of a given feature.

  These two properties are highlighted by the following example for a variable that stores the elevation of a geographic location: In this case, the rank property is fulfilled, because without any further information it can be derived that an elevation of 1903 meters is higher than an elevation of 1666 meters. The distance property is also fulfilled, because a distance between the evaluation values can be defined, and used to compare these values: an elevation of 1473 meters is much closer to 1410 meters than it is to 2007 or 1982 meters.

  Other properties, such as the linear or nonlinear scale of a variable, or the possibility to apply mathematical operations such as division or multiplication depend on the exact nature of the variable.

  - **Continuous**: Such variables contain real values that describe some numerical aspect of an object. Examples are the values of a temperature sensor, the birth rate of a country, or the latitude and longitude values of a geographical location. Thereby, the term continuous is a very general term that can be categorized in further sub-categories

such as *interval scale* or *ratio scale* variables. It is important to note that integer based values, such as the number of wheels on a car, are often associated with this category. However, in a mathematical sense these values are not considered as continuous.

- **Interval scale**: Such variables are based on a linear scale and may contain any real positive or negative values. Furthermore, a rank can be introduced based on the values and the same importance of the values is kept throughout the value range due to the linear property. Another important aspect of interval scales is that multiplication and division is not possible in the general case, which is highlighted by another example for elevation values: In general, interval-based values use a defined zero point. E.g., zero degrees Celsius or the elevation at sea level, which is 0 meters. However, this zero point does not indicate that there is no temperature or elevation at this level, it is simple a point of reference. Therefore, when looking at the elevation and taking sea depths into account one cannot say, that 200 meters of elevation are twice as much as 100 meters, since the true zero point is the maximum depth of -10924 meters at the Mariana Trench. Further examples for intervals are temperature values or the years as defined in a calendar.
- **Ratio scale**: In contrast to the *interval scale*, these variables are based on a nonlinear scale. Otherwise the same properties hold. Examples are the exponential growth function of bacteria or the logarithmic response to light of the human eye.

When transforming the data into a representation that is suitable for a machine learning algorithm, the category of each variable needs to be determined in order to apply the appropriate preprocessing and transformation steps based on the specific machine learning algorithm. For the further discussions in this chapter and the *Semantic Pattern Transformation*, these definitions need to be extended by introducing two further categories to which all of the different variable-categories described above can be assigned.

- **Symbolic variables (SF):** These variables or features contain values that cannot be put into a relation by applying a distance-measure. Thus, the *nominal* and *ordinal* variable fall into this category. As example, the feature values *iron*, *fruits* and *coal* of the feature *export commodity* are considered. They cannot be brought into a relation by a simple distance measure[1]. Although, one could still assign numerical values to the feature values and calculate the distance, this would not result in a meaningful relation.

[1]This would only work if we would use other aspects for describing these commodities: e.g., weight, density etc. However, here it is assumed that this data is not available and the values themselves represent the only available information.

- **Distance-based variables (DF):** The *feature values* of such features can be brought into a relation by defining a distance-measure such as the Euclidean distance. The previously described *continuous, interval scale* and *ratio scale* variables belong to this category. Here, the most important aspect is not only the calculability of the distance between different values, but the information conveyed by the calculated distance value. If the information carried by this distance is meaningful, the feature is considered as distance-based. It is not possible to give a general definition of "meaningful" since this depends on the nature of the data and the corresponding domain. Thus, the terminology is highlighted with several examples: The continuous feature *unemployment rate* which contains percentage-based feature values can clearly be assigned to distance-based features since the distance between values carries information about the relation between different values: One can clearly say that an unemployment rate of *5%* is closer to *10%* than it is to *30%*. However, in certain cases the feature values themselves might carry more information than could be expressed by the distances between these values. A good example would be the number of wheels of a vehicle. Here, the information whether a vehicle has two, four or six wheels reveals much more information than the distance between these values. For further details on this subject the reader is referred to Section 6.1.1 of *Chapter 6 – Semantic Pattern Transformation*.

Features, Properties or Attributes

Within the physical world the previously discussed raw variables are properties that describe certain aspects of arbitrary objects ranging from natural ones such as plants or animals, over man-made ones such as cars or buildings, to abstract concepts, such as countries or political parties. In machine learning these properties are named *properties*, *attributes* or *features*. The latter – *features* – will be used for the remainder of this thesis. In the context of features the following concepts play an important role:

- **Feature value**: A feature value is an arbitrary numerical or categorical value that describes some aspect of a feature. According to the previous description this could be a numerical value like the unemployment rate of a country or a temperature measurement of a sensor. It might also be a categorical value like the name of an export commodity (e.g., *iron*), or a single term within a textual description. Regardless of the value and its type, the feature value gets a meaning when put into the context with a feature that describes some aspect of an object.

- **Feature**: A feature is a property that describes some aspect of an arbitrary object with one or more feature values. Typically, multiple features are used for the object description. An example would be a country that is described by various features such as export commodities, unemployment rate, literacy and their corresponding feature values. The type of a feature is characterized by its feature values and thereby can be assigned to either

the symbolic or the distance-based category described previously in this section.

Instances or Examples

An instance is any kind of entity within the data set that is described by symbolic features, distance-based features or an arbitrary combination of both types. Thereby, the number of features can vary from instance to instance and the same features can be used multiple times within the same instance. An example for instances would be the countries within the CIA World Factbook [11] that are described by various symbolic and distance-based features such as *population size*, *language*, *unemployment rate* or *export commodity*. Since a country exports more than one commodity, the corresponding feature might occur multiple times within the same instance. Due to missing values (e.g., the unemployment rate) or non-existent properties (e.g., the length of the coastline), a feature might not be present in all of the instances.

Extracting Data According to a Relation

An instance could also be defined as the collection of features that co-occur according to a defined relation. Thereby, a relation could already exist due to related properties used to describe an entity, or arbitrarily defined on the data within a data set. Typically, when analyzing a data set, these relations are already defined or can easily be derived. The hierarchy of the instance extraction process is depicted in Figure 3.3, and Example 1 illustrates how data is arranged within a data set.

Example I: Extracting instances from a data set containing descriptions about the world's countries

The following examples describes the aspects of feature values, features, instances and relations. *Demo Data Set 1* is a data set comprised of various symbolic and distance-based features related to the world's countries. By applying a relation that combines features describing a country, instances (countries) can be extracted from the data set. These instances are described by multiple symbolic (e.g., export commodities) or distance-based features (e.g., unemployment rate) with corresponding feature values such as an unemployment rate of *10%*, or the export commodity *iron*.

3.2.2 Data Representation – Feature Vectors

Based on the definitions of feature values, features, instances and the high level relations that extract data from a given data set, the basic representation used for machine learning algorithms can be defined – the feature vectors. In this

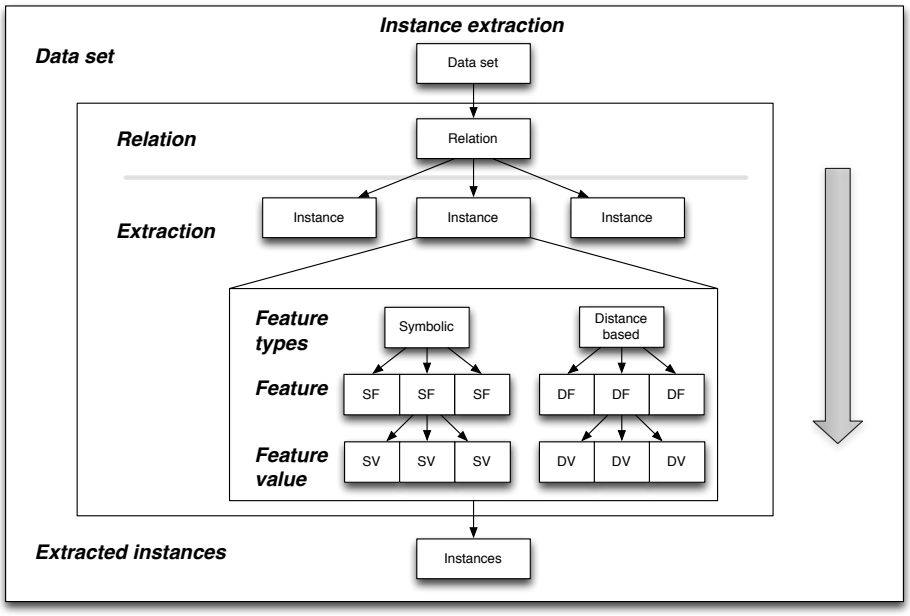

Figure 3.3: Hierarchy for the description of a domain-specific data set. The instances are extracted from the raw data set according to a relation. Each instance is described by various symbolic and/or distance-based features that contain feature values. The feature values and features are the basic components of the data set.

Country	Coffee	Cacao	Machinery	Chemicals	Unemployment rate	Fertility rate
C1	1	0	0	0	20%	5
C2	0	1	0	0	20%	5
C3	1	1	0	0	20%	5
C4	0	0	1	0	5%	2
C5	0	0	0	1	5%	2
C6	0	0	1	1	5%	2
C7	0	1	0	1	20%	**missing data**
C8	**missing data**				20%	5
C9	1	1	0	0	**missing data**	**missing data**

Table 3.1: The feature vectors representing the country instances of the demonstration data set used in *Chapter 2 – Semantic Patterns – At a Glance*. Each vector contains the same number of features, which remain at the same indices within the instance set. The feature values of the various features are stored within the elements of the feature vectors.

representation the instances, which are extracted via an arbitrary relation, are represented as vectors that contain the describing features and their feature values. Thereby, each element of a feature vector corresponds to a feature and contains the feature value that is used for the description. The position or the index of each feature within the vector must remain the same over all instances that are represented as feature vectors. As example, the feature vectors describing the countries of the demonstration data set from *Chapter 2 – Semantic Patterns – At a Glance* are shown again in Table 3.1.

Unfortunately, this commonly used feature vector representation is the main obstacle when deploying the same machine learning scheme within multiple domains.

3.2.3 Machine Learning Setup

Before machine learning algorithms can be applied, the data needs to be transformed into a feature vector representation. Thereby, the processing steps discussed in this section depend on the nature of the data, the capabilities of the specific algorithm, and the desired knowledge that should be extracted by the machine learning algorithm.

In [42], Kotsiantis et al. give a detailed presentation of the preprocessing steps required for the application of a supervised learning algorithm. Although the authors focus on the supervised nature of a machine learning algorithm, many of the explained preprocessing steps are also applicable to other learning scenarios, such as unsupervised clustering or extracting information about the relations between features and feature values. In order to provide a more generic machine learning model, the remainder of this section summarizes the principle operations presented in [42] and adds further processing steps. The resulting machine learning model is depicted in Figure 3.4.

Domain-Specific Data Set

Similar to a more general knowledge discovery process, an arbitrary domain-specific data set forms the basis for all further tasks required for the application of a machine learning algorithm. The nature of this data and the aspects covered by the data set form the basis for all further steps. Thereby, the data set could be an existing source of data, or the result of a specific data collection operation that was initiated due to the definition of arbitrary machine learning goals.

Machine Learning Goals

Prior to the selection of features and instances from the domain-specific data set, the goals of the machine learning scheme must be defined. In principle these goals define which knowledge should be gained by the application of the machine learning algorithm. Thereby, they could range from training a supervised classification algorithm, over the application of unsupervised learning for retrieving clusters of related concepts, to any arbitrary analysis that retrieves

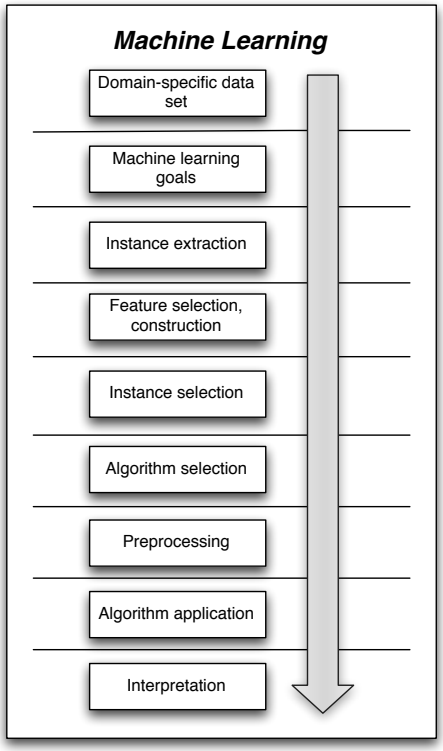

Figure 3.4: Machine learning model based on the processing steps defined by Kotsiantis et al. [42] and extended by further steps in order to cover different machine learning schemes.

hitherto unknown relations from the data set. Thus, the definition of the goals also determines the required algorithm families. The process of selecting specific algorithms from the chosen families needs to take place in a later step, because this choice depends on the nature of the data contained in the selected features.

The goal finding process is a vital basic step that influences all further choices including the specific machine learning algorithm, all preprocessing steps required to transform the data into an algorithm-specific representation and the employed method for extracting information from the trained algorithm model.

Instance Extraction

After defining the desired knowledge, and thereby the project goals, the available data must be transformed into the feature vector representation required by the employed machine learning algorithms. In order to do so, a relation must be defined on the existing data, that extracts instances from the raw data (e.g., country descriptions as in Example 2). These instances contain the feature and their feature values, which are later used for the training of the employed machine learning algorithm.

Example II: Defining relations on a data set containing information about the world's countries.

Assuming a data set that contains the raw data of the world's countries, relations can be defined on this data in order to extract instances for the subsequent application of the selected machine learning algorithms. The most obvious relation in this case would be an operation that extracts the country instances, which are described with features such as *population count*, *unemployment rate* or *export commodities*. However, other relations that group the available data into other instance sets are also possible. An example would be a relation that extracts instances for the world's continents, and thereby combines all the features and feature values of the countries located on a given continent.

Feature Selection

This process selects and extracts features that are considered as relevant for the extraction of the desired knowledge, and removes other non-relevant or redundant features. Thereby, the number of features, which corresponds to the dimension of the later used feature vectors, should be kept as small as possible due to the accompanied decrease in computational complexity and the increase of the expected quality of the trained model.

The feature selection process can either be focused on single features that are either removed or selected depending on their relevance for the desired information, or on subsets of different features that are selected according to their relations. In the latter case, e.g., redundant features that are strongly correlated could be removed. Many different simple and sophisticated feature selection algorithms exist and their application depends on the available data and the desired information. Although parts of the selection process can be automated via feature selection algorithms, in most cases the manual selection of features according to the available data and the defined machine learning goals still plays a significant role.

Feature Construction

In this step, new features are constructed on the basis of existing features as highlighted by Example 3. Basically, any operation that uses some existing information as input and yields a new feature as output can be utilized here. Typically, such operations need to be defined manually by the domain experts. However, in certain cases the construction of new features can also be automatized by employing algorithms that use the information contained in the existing features and create new features based on this information. Examples for such algorithms are the GALA and the FICUS algorithms [37], [56].

> ***Example III: Feature construction.***
>
> Given the numerical features *population count* and *television sets* used for the description of a country, then a new feature *television sets per 1000 people* could be easily constructed by dividing the number of *television sets* by the *population count* and multiplying the result with 1000.

Instance Selection

After selecting the available features, constructing new ones and arranging them into instances, an instance selection process is typically applied to create the instance set that is used as input for the later applied machine learning algorithms. It is important to note that this selection process is not related to the relation for extracting the basic instances as discussed in the previous *Instance extraction* step (e.g., by extracting only European countries). Instead, this selection process focuses on the filtering of instances that contain obvious irrelevant or corrupt data, outliers, or duplicate instances, and deals with instance sets that have imbalanced class distributions. The latter case plays an important role during the training phase of a supervised learning algorithm.

The application of the instance selection processes depends on the machine learning goals, the employed algorithms and the nature and the quality of the data. Thereby, the process may include operations based on simple methods that filter instances according to their contained data, or more sophisticated operations: Examples for the latter are statistical methods for discovering feature values that do not fit into the probability distribution defined by the other values, or unsupervised algorithms that are applied to the instances for the identification of outliers.

The instance selection process does not play an important role within the context of *Semantic Patterns*.

Algorithm Selection

Depending on the machine learning goals and the thereby required algorithm families, specific machine learning algorithms need to be chosen. Although,

many machine learning algorithms are available within an arbitrary family, a careful choice must be taken that takes various properties of the data and the desired knowledge into account:

- **Nature of the data**: The nature of the features and feature values contained in the to-be-analyzed instances has a major influence on the algorithm selection process. Certain algorithms cannot handle categorical or numerical data, require a certain structure within the data, or have a bad performance on certain kind of data. In order to cope with the limitations of a selected algorithm, transformation operations can be applied to the data. However, such operations might be complicated or even cause a degradation of the data quality. An example would be a transformation of numerical values into categorical ones, also known as discretization, when an algorithm is used that is only capable of handling the latter category. Here, the distance information contained in the numerical values is lost during the transformation.
- **Algorithm model**: Although many algorithms achieve the same principle goal – e.g., the unsupervised clustering of data – the employed techniques and thereby the algorithms models are quite different. The nature of the algorithm model strongly influences the knowledge extraction methods that need to be applied at the end of the machine learning processing scheme. Certain models are easier to interpret than others, or enable the extraction of specific information.
- **Desired knowledge**: The type of knowledge that should be gained after the application of the machine learning algorithm must carefully be analyzed before an algorithm is chosen, because the available knowledge depends on the employed algorithm and its specific model. When a clustering algorithm shall be applied in order to find anomalies within the data, certain algorithms might be better suited than others depending on the employed techniques and models.

The selected algorithm family and the specific algorithm have a strong influence on all subsequent steps prior to the application of the machine learning algorithm. In order to apply the appropriate preprocessing steps one needs to know exactly which model the selected algorithm employs, which data it can handle and which limitations it has.

Nowadays, there is a vast number of machine learning algorithms that are based on different techniques and have been adapted according to the requirements of different environments. A good overview on the most important ones is given in [90]. The algorithms that play a role within this thesis are discussed in *Chapter 5 – Techniques*.

Preprocessing – Data Transformation

Most machine learning algorithms are able to handle only categorical or numerical values. Thus, depending on the data and the selected algorithm, categorical

values must be transformed into numerical ones or vice versa.

Furthermore, special care must be taken when the number of features is not constant over the extracted instances. Thereby, a feature might not occur, occur once or multiple times within each of the analyzed instances. One needs to make sure that this is represented adequately in the feature vector representation needed for the machine learning algorithm.

In contrast to the direct representation of distance-based values within a feature vector, transformations must be applied to categorical values. Here, an example would be the bag-of-words model that is widely used within text analysis but can also be extended to other categorical data. Thereby, an element for each feature value of a given categorical feature is created within the feature vector and the absence or existence of the feature value is marked via a numerical value.

Regardless of the applied transformation operations, they must always be chosen and adapted in concordance with the selected machine learning algorithm.

Preprocessing – Handling of Missing Feature Values

Within the feature vector representation, the number of features and their position within the feature vector need to be constant within the set of analyzed instances. However, quite often features are missing within a subset of the analyzed instances. Thereby, the term missing refers to several possible explanations: **First**, the feature is not appropriate for the given instance, **second**, the corresponding data is not available due to errors in the data set, damaged sensors etc., and **third**, the value of a feature does not play a role within the analysis. While the first case is already covered by the previous data transformation step, the other cases are related to missing values that need to be considered. The reason why the missing values cannot be left within the data, is that the employed machine learning algorithm typically is not able to handle them.

Thus, there are various strategies that deal with missing feature values, which are summarized by Kotsiantis et al. [42]:

- *Removal of the instances with missing feature values*: While this is the simplest approach, it also removes the information contained in these instances. This might have a negative influence on the quality of the results, especially when small data sets are analyzed.

- *Replacement of the missing values with the most common feature value of the corresponding feature*: This can either be seen in the context of all instances or only for the class the instance with the missing value belongs to. Obviously, the last method is only relevant when class information is available.

- *Replacement of the missing value by the mean value of all feature values of the corresponding feature*: Again, this can also be seen within the context of all instances or within the class of the instance with the missing value.

- *Retrieval of the instance that has the largest similarity with the instance that contains the missing value:* The missing value is then replaced with the feature value from the retrieved instance.
- *Treatment of missing values as special values*: For categorical values a new feature value "missing" could be introduced.

As discussed in the next chapter, the need to handle missing feature values is caused by the value-centric representation that forms the basis of the commonly used feature vectors.

Preprocessing – Data Normalization

Within machine learning algorithms feature vectors are typically related via some kind of distance measure. Thereby, when multiple numerical features are used for the description of an instance, their value ranges typically diverge and therefore have a different influence on the distance calculated between two feature vectors. These differences can also be compared to weights that correspond to the relevance of the features. Especially in unsupervised learning no additional information such as class labels is available that might help the algorithm during the training phase to determine the importance of each feature within the available feature vectors. Therefore, normalization procedures need to be applied that aim to equalize the influence of different numerical features on the distance calculation.

However, normalization does not consider other available information about the importance of features that could be used to emphasize certain features while attenuating others. Even if such information is available, the assignments of appropriate weights to the features and their values is no straight-forward process.

Algorithm Application

In this step the machine learning algorithm is finally applied to the feature vectors representing the instances. Depending on the algorithm family, specific modes of training might be necessary that use different subsets of the available feature vectors for algorithm training and for evaluating the results. The description of these schemes is beyond the scope of this thesis and the reader is referred to e.g., [9].

Interpretation

The trained model of an arbitrary machine learning algorithm forms the basis for extracting and interpreting the knowledge that was defined in the second step (*Machine learning goals*). Although there is a wide range of algorithms that in principle yield the same results, the employed models often are based on distinct techniques. Therefore, the knowledge extraction and interpretation procedures must be adapted to this model. Also, the availability of information depends on

the employed model and cannot be guaranteed for all algorithms within a given family. Due to these reasons, the specific choice of the machine learning algorithm in the step *Algorithm selection* must take the previously defined machine learning goals into consideration.

3.3 Machine Learning and Knowledge Discovery

In Figure 3.5 the knowledge discovery process based on the Fayyad et al. model is compared with the typical processing steps required by a machine learning algorithm. As the visualization indicates, many of the operations from knowledge discovery can be directly mapped to operations from machine learning. This is not surprising since the previously discussed knowledge discovery models are generic ones that cover all possible techniques and algorithms employed in a KDP. The application of machine learning is merely one of many methods that can be applied within knowledge discovery.

Similar to the previously defined knowledge discovery task (KDT), a machine learning based knowledge discovery task (ML-KDT) can be defined that represents a module within a complete KDP process as it was already depicted in Figure 3.1. This figure has been adapted to machine learning in Figure 3.6.

3.4 Deployment in Heterogeneous Domains

The deployment of a KDP characterized by the domain and the knowledge discovery goals requires various processing steps. Unfortunately, many of these steps require manual interactions and adaptations that require a large effort and need to be changed whenever the nature of the analyzed data or the goals change. Furthermore, in a typical scenario the setup even needs to be changed even when the defined goals remain the same, but only the nature of the input data changes. The following summary is based on the processing steps discussed in this chapter, and identifies the main obstacles in the heterogeneous application of the same KDP within multiple domains:

- *Specific setup for each algorithm*: The choice of the algorithm depends on the nature of the data and thereby on the domain. I.e., some supervised and unsupervised algorithms were developed only for continuous data, while others can only be applied to categorical data and the application to mixed data requires preprocessing of the data. Furthermore, certain algorithms require different normalization steps in order to work correctly.

- *Need for an interpretable model*: Given a data set from a specific domain, the analysis must be based on the best available algorithm for the desired knowledge mining tasks. However, for the extraction of meaningful knowledge, the applied algorithm also needs to employ a model that is

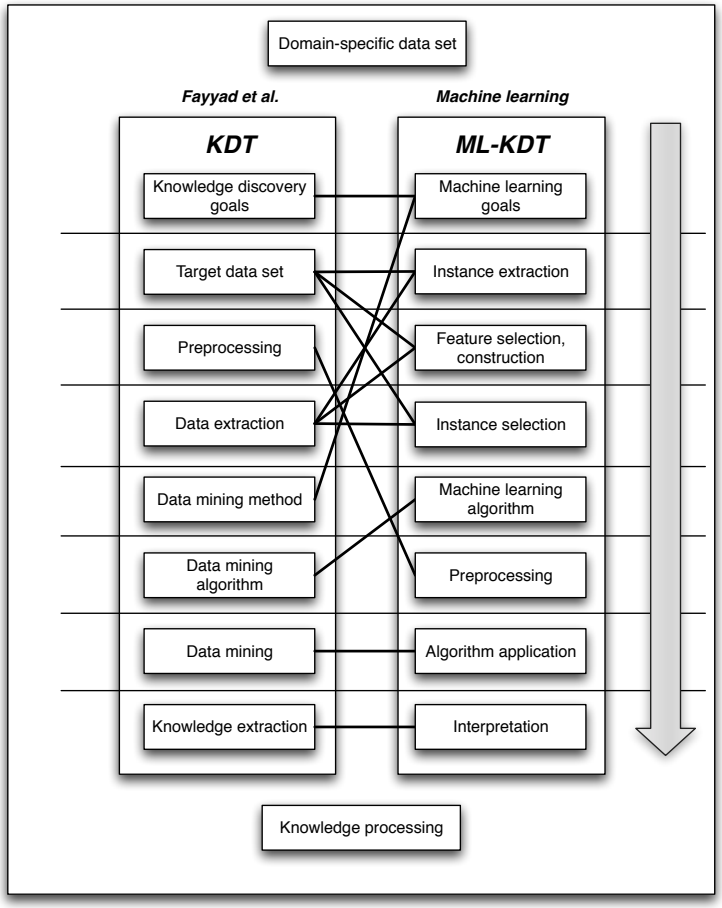

Figure 3.5: This figure compares the KDP model (left) to a machine learning scheme (right). The connections indicate how the processes of both models are related. Although the order and the specific names of the processing steps differ, they both represent the same tasks on an abstract level.

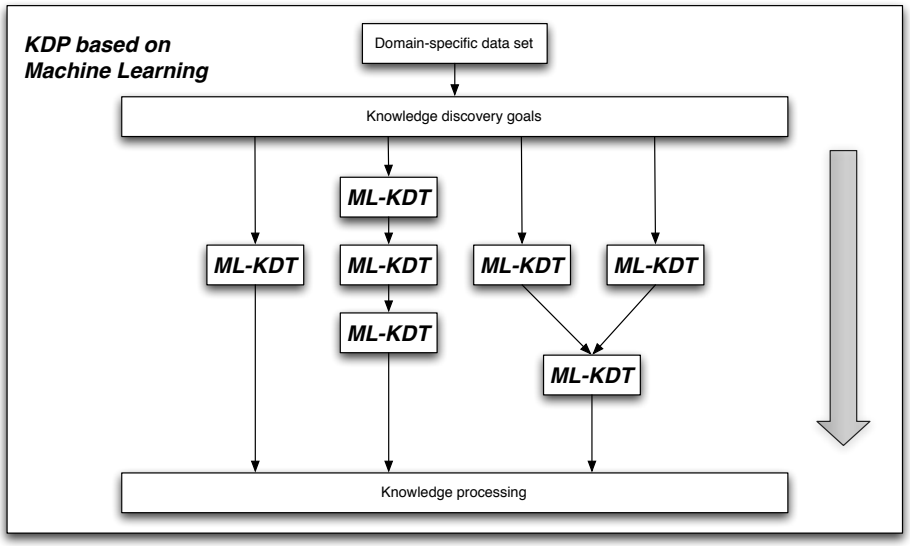

Figure 3.6: The KDP model adapted to incorporate machine learning based knowledge discovery tasks (ML-KDT).

easy to interpret. Depending on the desired knowledge and the appropriate algorithms, a good balance between these two – often conflicting – requirements must be found.

- *Domain-specific issues*: Typically, different knowledge mining tasks require a specific setup, which includes choosing a specific algorithm, preprocessing the data according to the algorithm requirement and devising a method that extracts the desired knowledge from the generated model. Since the complete knowledge discovery process within a given domain and data set typically involves multiple of such heterogeneous knowledge mining tasks, much effort must be placed into the specific setup of each algorithm. When additional tasks are added, this setup process needs to be repeated due to the general incompatibility of preprocessing steps, the employed algorithms and the trained models.

- *Non-extensibility*: The previous issue also implies that existing knowledge mining tasks cannot be easily extended to cover other aspects of knowledge extraction. Therefore, simple additional requirements might cause time-costly changes in preprocessing steps and algorithms.

3.5 Chapter Conclusions

This chapter **first** discusses generic models for knowledge discovery processes (KDP), which consist of one or multiple knowledge discovery tasks (KDT) that

are either independent from each other or use the output of other KDTs as input. **Second**, the application of a machine learning scheme and the required processing steps are discussed in relation to the deployment within KDPs. Especially, when data of heterogeneous domains is analyzed, many of these processing steps and the employed algorithms must be adapted to the existing data and desired knowledge. This is a rather time-consuming setup, which includes the adaptation of many steps which limits the heterogeneous deployment of KDPs. In the next chapter the value-centric feature vector representation will be identified as the main reason for the required setup changes, and the shift towards a semantic representation will be motivated.

4

Towards a Semantic Representation

The integrating of machine learning procedures into heterogeneous Knowledge Discovery Process (KDP) enables highly sophisticated analysis process. Unfortunately, the setup procedure for such KDPs heavily depends on the domain-specific data set, the desired information and the employed algorithms, which significantly limits the reusability of existing setups.

In this chapter, the domain-specific setup procedure and the associated processes will be analyzed in respect to the complexity of the required adaptations. Based on this analysis, the core issue – the value-centric representation of feature values within the feature vectors – will be identified, and the benefits of shifting to a semantic representation will be discussed. The principles behind this semantic representation form the basis for the *Semantic Pattern Transformation*.

4.1 Analysis of the Processing Steps

The application of a machine learning algorithm within knowledge discovery requires the application of many preprocessing steps. In this section, these steps will be discussed in the context of the adaptations that are necessary when the domain data set, or the knowledge discovery goals change.

4.1.1 Complexity of the Required Adaptations

In Figure 4.1 the basic processing steps of a machine learning based knowledge discovery task (ML-KDT) are depicted. These steps are subject to adaptation when changes are made to the machine learning setup. The complexity of these adaptations is indicated by the following colors: green (low complexity), yellow

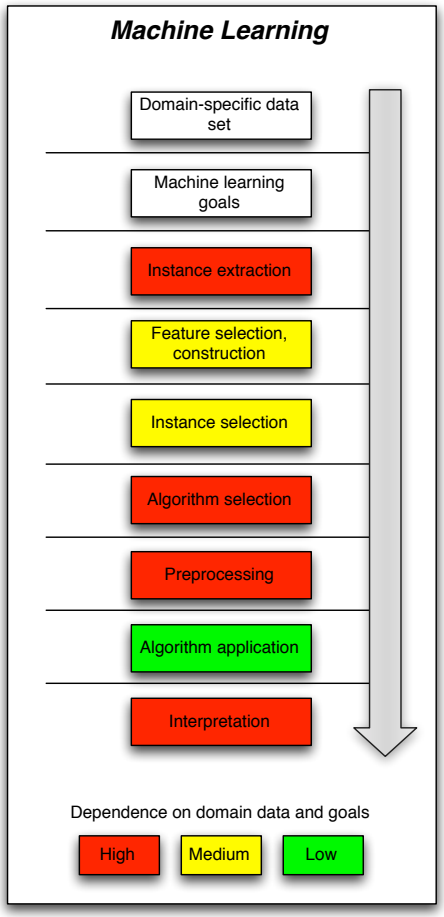

Figure 4.1: The effort required to adapt the processing steps for heterogeneous domains. The complexity of the first two steps is not considered, because they need to be defined manually and describe the basic environment for the complete process.

(medium complexity), and red (high complexity). Thereby, the first two steps – the selection of the domain-specific data set, and the definition of the machine learning goals – are not taken into consideration, because they form the basis for arbitrary knowledge extraction procedures and need to be defined manually per default.

- **Data**: The analyzed data changes when different aspects of the same domain data set need to be analyzed, or when another knowledge discovery domain is investigated.
- **Goals**: Here, the analyzed data remains the same, but the machine learning or knowledge discovery goals are changed. Typically, these changes lead to significant setup adaptations including the selection of a different algorithm family and the associated algorithms, the adaptation of the data preprocessing and transformation steps, and the required transformation of the final knowledge extraction process.
- **Instance Extraction (high)**: In general, the process of defining a relation, which combines the features into instances, is a manual operation that needs to be adapted to the domain data set and the machine learning goals. In some cases the data set is already organized in a way that enables the direct extraction and use of the appropriate instances, but this is not the typical scenario. Furthermore, after defining the relation, a specific instance selection process might be applied that extracts a subset of the instances according to some defined aspect. Examples are the filtering of outliers, the focus on a certain class of instances, or the application of any other filter that is chosen according to the defined machine learning goals. Although certain aspects, such as outlier detection can be automized and probably do not need to be adapted when the setup changes, most of the applied operations still need to be executed manually.
- **Feature selection/construction (medium)**: For the typical application, some of the feature selection processes can be automized due to techniques that automatically identify the relevance of certain features (e.g., the Principal Component Analysis [74]). However, the manual selection still makes sense, especially when a priori domain knowledge is available. The feature construction process can be partly automatized but is largely based on manual operations. However, this process is not always required, which justifies the assignment to the medium complexity category.
- **Algorithm selection (high)**: The selection of a machine learning algorithm strongly depends on the analyzed data and the defined machine learning goals. Therefore, this process needs to be adapted whenever the setup changes.
- **Preprocessing (high)**: Preprocessing and data transformation are also highly domain-specific, because they need to be adapted to the data and the selected algorithm.

- **Algorithm application (low)**: The setup for the application of the algorithm largely consists of the preprocessing steps that need to be applied to the data in order to provide the correct structure for the algorithm. However, the application of the algorithm typically does not require complex manual interactions or setup procedures.
- **Interpretation (high)**: The interpretation of the results strongly depends on the data, the deployed algorithm and the defined machine learning goals and, thus, is considered to require significant manual effort whenever the domain changes.

4.1.2 Reasons for the Adaptation Complexity

This section takes a closer look on the processing steps that require complex adaptations when the setup changes, and identifies **seven** specific **Reasons** for this complexity.

Algorithm Selection

The selection of the algorithm plays a decisive role in the quality of the results gained by the application of a knowledge discovery process. Thereby, two issues need to be taken into consideration during the selection process: **First**, the selected algorithm influences all the preprocessing steps and the final knowledge extraction step, which all need to be adapted according to the capabilities and properties of the selected algorithm. **Second**, the defined knowledge discovery goals and nature of the analyzed data place bounds on the algorithm selection, which must be considered.

The first issue causes the required adaptation of the subsequent processing steps and thus induces a significant part of the setup complexity. The second issue causes the requirement to select different algorithms, even when only the analyzed data changes, but the knowledge discovery goals remain the same. When analyzing these two issues, the first two **Reasons** are identified:

- **Reason 1 – Nature of the data:** The analyzed data is comprised of instances described with numerical and/or categorical features. Thereby, the composition of these features – only numerical features, only categorical features, or a mixture of both types – requires the selection of an adequate algorithm that is capable of handling the available data. Before the algorithm can be applied, the data must be transformed into the common feature vector representation (discussed in *Reason 3*), which is only capable of representing numerical values. Thus, when categorical features are present within the data set, they need to be transformed into a numerical representation. Unfortunately, in reality there is no perfect numerical representation of categorical features within the value-centric feature vector representation. A good discussion on the issues that occur when handling categorical data is presented in [33], where Guha et al. present the unsupervised learning algorithm ROCK, which is capable of

handling categorical data. When an appropriate choice has been made, the analyzed data needs to be processed according to the requirements of the selected algorithm, which is described in *Reason 3*.

- **Reason 2 – Suitability of algorithm model for knowledge discovery**: Machine learning algorithms, even when they belong to the same algorithm family (e.g., supervised learning), are based on different techniques which influence the aspects of the algorithm model. Thereby, the ability to extract certain type of knowledge from a given model might be subject to strong limitations or at least requires the application of very specific extraction processes, whose complexity depends on the nature of the model. Due to this limitation, a careful choice must be made in order to achieve the defined knowledge discovery goals. Details on the algorithm specific knowledge extraction processes are given in *Reason 7*.

Preprocessing – Data Transformation

After selecting an appropriate algorithm, the available data must be transformed into an adequate feature vector representation that can be handled by the algorithm. Thereby, the following two **Reasons** need to be considered:

- **Reason 3 – Representation of categorical and numerical features**: The difference in the capabilities of machine learning algorithms to handle categorical and numerical features has a major influence on the algorithm selection process, as discussed in *Reason 1*. When a certain algorithm has been chosen according to the nature of the data, a feature vector representation adapted to the algorithm must be found for the categorical and numerical features. While in most cases the feature values of numerical features can directly be mapped to the value-centric feature vector representation, data and algorithm-specific transformations need to be applied to the categorical features.

- **Reason 4 – Variability in the number of features describing the instances:** The feature vector representation requires that each instance is described with the same number of features, which must remain at the same position for the complete instance set. However, not all objects represented as instances can be described directly in this way. Some features might only be present within certain instances and absent within others. Furthermore, features might occur more than one time in one instance. This variability could be related to missing values, but in the most cases is caused by the nature of the properties use to describe the instances. Regardless of the cause, an adequate transformation needs to be applied in order to make the description compliant with the feature vector representation.

Preprocessing – Handling of Missing Values

Even when the number of features is constant or an adequate feature vector representation has been found for instances with different number of features, there is still the possibility of missing values.

- **Reason 5 – Handling of missing values:** Missing values must not be confused with values that are not present, because their corresponding features or properties are not needed, or are not valid for the described object. The fact that a country does not export *iron* is not considered as a missing value. However, the non-availability of data due to errors, or not capturing the data during the data extraction process, are considered as missing values in this context. In the first case, the information that a property is not available needs to be addressed by the transformation operations. In the second case, the information is simply not present and some way of representation needs to be found for the feature vectors. The selection of one of the available methods discussed in the previous chapter is a manual process and depends on the environment defined by the data and the machine learning goals.

Preprocessing – Data Normalization

The requirement to normalize the feature values and thereby equalizing the influence of the features contained in an instance is a mandatory step in most of the machine learning setups. The rationale for this requirement is based on the value-centric feature vector representation:

- **Reason 6 – Data normalization:** The value-centric feature vector representation is based on numerical values, which are either directly extracted from numerical features, or are the result of applying transformations to categorical features. Since these feature values are stored in the vectors, their different value ranges have a direct influence on the similarity calculation employed by machine learning algorithms (especially within unsupervised algorithms). In order to equalize this influence, normalization operations need to be applied. Although some default operation could be applied to arbitrary feature vectors, better results typically can be achieved when the strategies are chosen manually according to the analyzed data and the selected algorithms.

Interpretation

In order to extract knowledge after the application of a machine learning algorithm, the algorithm model needs to interpreted. Unfortunately, this interpretation process heavily depends on the selected algorithm and the thereby utilized model.

- **Reason 7 – Model dependent knowledge extraction**: Depending on the model employed by the algorithm, limits are placed on the possibility to extract information after training the model. As example, the

kernels employed by Support Vector Machines [13] are more difficult to interpret than the model used for the Self-Organizing Map [39], which was specifically designed for the visualization of high-dimensional data.

Due to these limits, the interpretation of the results can also be shifted from the models to the actual feature vectors. The application of a machine learning algorithm typically assigns the analyzed feature vectors to different categories or groups. By focusing the analysis on the properties of the feature vectors within these groups, the possible complex processing and interpretation of the algorithm models can be avoided. However, due to the structure of the feature vectors, the extraction of knowledge is not straightforward and multiple issues need to be considered: **First**, due to the transformation and preprocessing steps applied to the feature vectors, some kind of mapping operation needs to be applied that translates the information back to the original contained information. **Second**, only the feature values of the features are contained in the feature vectors. There is no additional information on the relation of these features or values, and thus further processing is required in order to gain such information. **Third**, transformations from categorical into numerical values and vice versa need to be considered when interpreting the results. **Finally**, if further information needs to be extracted via additional processing operations, then again, specific adaptations need to be applied in order to format the input accordingly.

4.1.3 Adaptation Complexity – Summary

These seven **Reasons** can be assigned to five main issues that cause the requirement to adapt the processing steps:

- **Nature of the data – numerical and categorical features**: *Reason 1* in *Algorithm selection* and *Reason 3* in *Preprocessing – Data transformation*
- **Model and data dependent knowledge extraction and interpretation process**: *Reason 2* in *Algorithm selection* and *Reason 7* in *Interpretation*
- **Variability in the number of features describing the instances**: *Reason 4 Preprocessing – Data transformation*
- **Handling of missing values**: *Reason 5* in *Preprocessing – Handling of missing values*
- **Data normalization to cope with different value ranges**: *Reason 6* in *Preprocessing – Data normalization*

When looking at Figure 4.1, there are further three processing steps that were not discussed discussed in this analysis: *Instance extraction*, *Feature selection and construction*, *Instance selection* and *Algorithm application*. Thereby, the

last one – *Algorithm application* is not discussed, because the pure application of the algorithm is for the most part not domain-specific. The important adaptations are already covered by the preceding preprocessing operations. The other processing steps are not relevant for the discussions due to another reason: The *Semantic Pattern Transformation* introduced in this thesis, cannot simplify or remove the adaptations required for these steps.

4.2 The Value-Centric Representation

By analyzing the seven reasons for the adaptation complexity of the processing steps, several core issues can be identified that are based on the value-centric feature vector representation.

- **Independence of the feature values**: The value-centric representation stores the feature values independently of each other. By looking at a feature vector alone, the relations between features and feature values are not revealed. Thus, additional operations such as calculating the correlation between features are required to reveal these relations.

- **Numerical representation**: The value-centric representation can only handle numerical values. Thus, categorical features and their values must be transformed into numerical ones prior to the application of a machine learning algorithm. The simplest transformation process just assigns numerical values to the categorical feature values. However, this representation is problematic since categorical values, which cannot be related via distances, are then simply transformed into a numerical representation for which the distance calculation is still applied within the machine learning algorithms. In order to cope with this issue, typically one dimension is used for every categorical feature value and its number of occurrence or its absence is represented with numerical values. However, this still requires special distance measurements that take the categorical nature into account. Unfortunately, when categorical and numerical features are mixed, then the deployment of distance measures applicable to the categorical parts may not be beneficial for the numerical values or vice versa.

- **Different meaning of the values**: Even when only numerical features need to be represented as feature vectors, and the problem imposed by the transformation of categorical values into numerical ones can be safely ignored, there is still a fundamental problem: The values stored in the vectors are only meaningful when viewed in context with their corresponding feature. Thus, two feature values of different features cannot be compared in theory because they represent different properties. However, despite this incompatibility the values are still combined when calculating the distance between two instances. Thereby, the different value ranges are compensated by equalizing them with normalization operations. These removes the imbalanced influence of different features, and allows the combination

of all feature values within distance calculation operations. However, two issues remain: **First**, this equalization of the weights is still arbitrary since it does not represent the real relevance contributed by the features and their relation to others, and **second**, the values are combined within the distance calculation, although they might represent completely different properties.

- **Meaning of the representation**: When looking at the data within a feature vector, the size of the feature values and their meaning can only be understood within the context of the specific machine learning setup. Similar, the feature value itself is only meaningful when the corresponding feature is known, and in general has a completely different meaning within another feature. In addition, since different numerical values and categorical values are represented within a feature vector, various preprocessing and transformation steps need to be applied to the features. Thus, the interpretation of the feature vectors must always be seen within the context of the whole machine learning setup.

In summary, these four main issues may lead to the requirement for the manual adaptation of many processing steps once the domain changes.

4.3 The Semantic Representation

As a main contribution of this thesis, a new semantic representation is presented that shall overcome the problems of the discussed value-centric feature vectors. This semantic representation is still compatible with the standard feature vector representation and removes the need for many manual operations within the processing steps of a machine learning scheme. In this section, the key properties and benefits of this semantic representation will be discussed without going into the technical details of the required transformation steps, which will be covered in the following two chapters.

4.3.1 Basic Idea and Properties

The principle idea behind the presented semantic representation is based on the information gained by the semantic analysis of the features and feature values contained in the analyzed instances. Thereby, two feature values are considered to be semantically related when they co-occur within an instance. The number of the co-occurrences over the complete set of analyzed instances defines the strength of the semantic relation.

This new representation comes with the following properties that have a significant influence on the processing steps of the machine learning setup.

- **Feature vector representation:** The semantic representation is still compatible to the standard value-centric feature vector representation. This is vital for the application of standard machine learning algorithms

since they all require that the input data is represented as feature vectors. Therefore, in the following discussions the feature vectors of the semantic representation will be referred to as semantic feature vectors.

- **Common representation**: The semantic feature vectors are the result of applying a transformation operation to the value-centric feature vectors. The semantic feature vectors represent all types of features and their combination within the given instances. This new representation is based on the analysis of the semantic relations between the features and their feature values. Thereby, the semantic feature vectors fulfill the following three core properties:

  1. **Numerical values**: All features – even categorical ones – and their values are represented as pure numerical values within the semantic feature vectors after applying the transformation operation. Thereby, the transformation ensures that the numerical values can be related via distance measures, which is not the case for the value-centric representation of categorical values within the standard feature vectors.
  2. **Same meaning**: All elements of the semantic feature vectors are of the same nature and convey the same information. This is in stark contrast to the value-centric feature vector representation.
  3. **Similar value ranges**: As a consequence of the previous property, all elements of the semantic vectors have similar value ranges.

- **Semantic relations:** The semantic vectors contain information that describes the semantic relations between the represented feature values. This is also a significant difference to the value-centric feature vectors where each feature is considered as independent. In the semantic representation, a single feature value is represented with a semantic fingerprint that describes the strength of the relation to other feature values.

The next chapter will give a detailed description of the process that transforms the value-centric feature vectors into the semantic feature vectors.

4.3.2 Influence on the Machine Learning Setup

The described properties of the semantic representation have a significant influence of the machine learning setup, and remove the need for many manual operations, which results in the simplification of the required processing steps. These differences are depicted in Figure 4.2.

- **Preprocessing – Data transformation**: The definition and application of manually defined transformation operations based on the nature of the analyzed features and the selected algorithm is not required. Instead, one generic transformation is applied in order to transform instances described by arbitrary numerical and/or categorical features into generic semantic feature vectors.

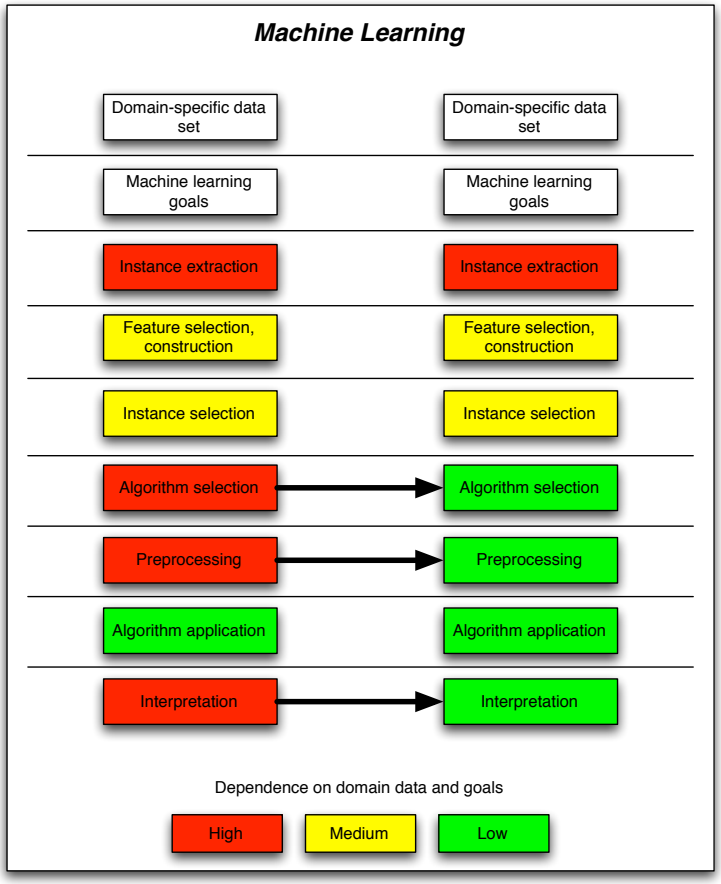

Figure 4.2: Differences between the adaptation complexity required for value-centric feature vectors and semantic feature vectors.

- **Preprocessing – Data normalization**: The semantic feature vectors consist of numerical values only, which have similar value ranges. Thus, normalization operations are not required.
- **Preprocessing – Handling of missing values**: The transformation process analyzes and stores the semantic relations between feature and feature values. Due to the availability of this additional information, missing feature values do not need to be handled, because their information is automatically reconstructed within the transformation process by utilizing the semantic relations between the feature values.
- **Interpretation**: The semantic feature vectors can be directly interpreted without the need to specially adapt the process according to the algorithm model or the represented features. Thus, one common knowledge extrac-

tion method can be deployed which is capable of extracting information regardless of the analyzed features and employed algorithms.

- **Algorithm selection**: Due to the common numerical representation of the semantic feature vectors and their simple interpretation, the selection of the appropriate algorithms is significantly simplified. In general, a single algorithm per algorithm family can be selected and reused for arbitrary knowledge discovery processes.

4.3.3 Further Advantages of the Semantic Representation

In addition to the advantages gained by reducing the complexity of the machine learning setup, this new representation offers high level benefits due to the common model, which is capable of representing all types of data:

- **Extensibility**: The *Semantic Patterns* model is independent of the analyzed domain and therefore can be used for sophisticated analysis processes based on the output of single or multiple ML-KDTs. In contrast to the value-centric feature vector representation, these ML-KDTs do not need to be adapted when the domain changes.

- **Understanding of unknown data**: Especially, when no a priori knowledge regarding the features and their relations is available, the common semantic representation has a key advantage: Regardless of the employed feature and feature values, the information contained in the semantic feature vectors is interpreted in the same way, because it focuses on the relation between feature values and not the values itself. Thus, when unknown data is analyzed, one is able to gain an understanding of the relations between the feature values without the need to understand their meaning.

- **Visualization**: The generic model and the semantic representation contained in the semantic feature vectors enable the creation of simple visualization and analysis procedures that benefit from the common model.

4.4 Semantic Patterns

For the remainder of this thesis the semantic feature vectors will be identified as *Semantic Patterns*, which are the result of applying the *Semantic Pattern Transformation* to arbitrary value-centric feature vectors. While the basic techniques required for this transformation will be explained in detail in the next chapter (*Chapter 5 – Semantic Patterns Techniques*), their arrangement and organization within this procedure will be covered in *Chapter 6 – Semantic Pattern Transformation*.

4.5 Chapter Conclusions

In this chapter, the value-centric feature vector representation is identified as the key reason for many setup steps that need to be adapted whenever the knowledge discovery domain or the associated goals change. In order to solve these problems, a semantic representation is suggested that does not consider the feature values themselves, but the semantic relations between them. This new representation, which is referred to as semantic feature vectors or *Semantic Patterns*, removes the need for many manual steps within a knowledge discovery or machine learning setup, while being compatible with the standard feature vector representation. The simplification and automatization of the processing steps enables the deployment in heterogeneous domains. Furthermore, due to the common model that represents data of arbitrary nature, further advantages in understanding, visualizing and interpreting data, and the extensibility of existing KDPs with sophisticated analysis procedures, are gained.

5 Semantic Pattern Techniques

Before going into the details of the *Semantic Pattern Transformation*, the required techniques will be explained in this chapter. The basic components are associative networks, spreading activation algorithms, and supervised and unsupervised machine learning algorithms. While the associative networks and spreading activation algorithms are required to analyze, store and query the semantic information contained in the value-centric feature vectors, the machine learning algorithms are mainly used for the analysis of the transformed *Semantic Patterns*. They also play a role within the conversion from numerical features into categorical ones. Regarding the machine learning algorithms, the focus in this chapter will be placed on those algorithms that are directly required by the *Semantic Pattern Transformation*. The other algorithms, which are used for the evaluation of the *Semantic Pattern Transformation*, will only be explained briefly and the reader will be referred to other works for more information.

5.1 Overview

The *Semantic Pattern Transformation* converts value-centric feature vectors into *Semantic Patterns* by utilizing three different techniques from the areas of machine learning and artificial intelligence: **First**, associative networks are used for storing the semantic relations between feature values gained by analyzing their co-occurrences within the instances. **Second**, in order to extract information from the associative network and generate the *Semantic Patterns*, spreading activation algorithms are employed. **Finally**, for the evaluation and the analysis of the *Semantic Patterns*, the supervised Support Vector Machine (SVM) algorithm, the supervised J48 algorithm, which is based on decision trees, and

the unsupervised K-Means and Expectation Maximization (EM) algorithm are employed. For storing numerical feature values within the associative network via a discretization procedure, the unsupervised Robust Growing Neural Gas algorithm is utilized. It must be noted, that the analysis of *Semantic Patterns* is not limited to these algorithms. In fact, arbitrary machine learning algorithms can be deployed.

5.2 Semantic and Associative Networks

According to Sowa [75], *"...common to all semantic networks is a graphic notation for representing knowledge in patterns of interconnected nodes and arcs"*. The concept of semantic networks was first presented in a work about semantic memory by Quillain [66], and is used for the representation of knowledge employed for machine learning, artificial intelligence and information retrieval. The concept is also related to other non-IT related areas such as philosophy, psychology or linguistics.

The basic components of a semantic network are nodes that represent information, and links (arcs) between these nodes that model the relations between the information elements. Based on the way the knowledge is organized within a semantic network and how this information is used, the network can be assigned to one of the following categories: *Definitional networks*, *Assertional networks*, *Implicational networks*, *Executable networks*, *Learning networks* and *Hybrid networks* [75]. An example for a definitional network is described in Example 4.

Example IV: Example for a semantic network describing countries

In Figure 5.1 an example for a semantic network that describes countries with various properties is shown. The links between the nodes refer to the different properties of a given country. E.g., Austria has two links of the type *agreement* – one is linked to the node *Eurozone* and the other is linked to the *Kyoto protocol*. By visualizing the information contained in the semantic network, a quick overview on the similarity between countries and the relations within the network is gained. This network could be considered as a definitional network and has a strong similarity to the semantic networks described with the Resource Description Framework (RDF) [59] language in nowadays semantic web. In this description an arbitrary subject (the country) is described via an object (e.g. the *Kyoto protocol*) that has a specific relation (agreement) to the described subject. Thereby, the objects themselves could become subjects that are described via further objects according to a relation.

Especially, learning networks play an important role in the area of machine learning and form the basis for artificial neural networks, which are employed by a wide range of supervised and unsupervised algorithms, such as supervised artificial neural networks [55], Self-Organizing Maps [39] or Neural Gas [57], [29], [65].

Another special type of semantic network is the associative network, which is also a member of the *Learning networks* category. In contrast to the typical semantic network, that employs various link types for modeling different relations, the associative network employs only specific type of link. This type represents the associations between nodes that are defined according to some

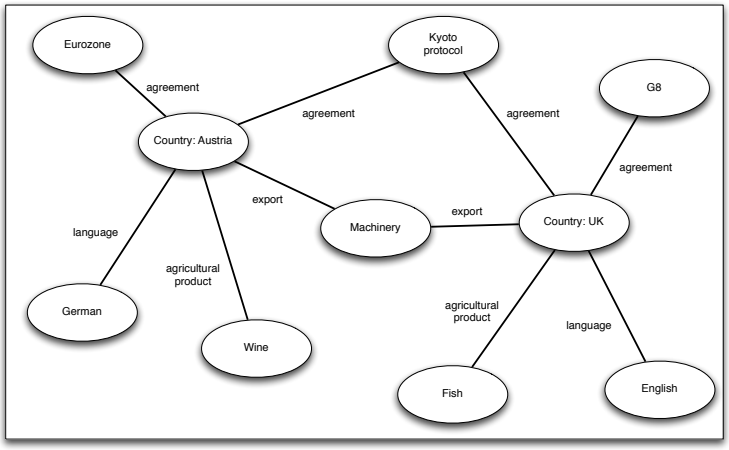

Figure 5.1: This example shows a definitional network that describes countries with different properties. Thereby, the properties are encoded as links and their values are stored within the network nodes.

kind of relation. Their strength is represented by weights that are assigned to the links, and the employed links can either be bi-directional or uni-directional. An example for an associative network is described in Example 5.

Example V: Example for an associative network representing export commodities

Figure 5.2 shows an example for a associative network describing the relations between commodities exported by different countries. The network is based on a subset of the data contained in the *Demo Data Set 1*, and consists of nodes representing export commodities, and weighted links modeling the associations between these commodities. Links or associations between various commodities are created when they are exported by the same country. Their strength is defined by the number of co-occurrences within all country descriptions. By looking at the network, one observes that the export commodity *coffee* is strongly related to *cocoa*, has a moderate association with *bananas* and *petroleum products*, and a weak link to *chemicals*. Furthermore, two clusters of export goods can be identified that have strong relations between their members but rather weak links that connect these clusters. The right cluster has export goods of countries that export *coffee*, *cocoa*, *bananas*, or *petroleum products*, whereas the left cluster is related to industrialized countries that export *machinery*, *chemicals* or *electronics*.

The *Semantic Pattern Transformation* introduced in this thesis employs associative networks – similar to the one in Example 5 – with uni or bi-directional

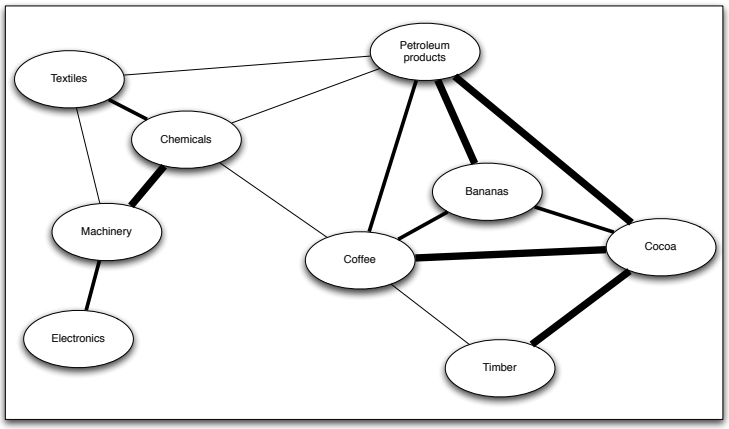

Figure 5.2: An associative network is based on nodes that carry arbitrary information, and weighted links between these nodes that describe the strength of their association or relation.

links for describing the semantic relations between feature values contained in the analyzed instances. Together with the spreading activation algorithm described in the next section, these two methods are the basic components required for the transformation of value-centric feature vectors into *Semantic Patterns*.

5.3 Spreading Activation

The initial idea for spreading activation algorithms is based on psychological studies that tried to find a model for human memory operations. The concept was then introduced in the artificial intelligence field within computer science, and adapted continuously for specific applications such as information retrieval and text analysis. Thereby, the technique is used to retrieve information from a given semantic or associative network. By activating one or more nodes within the network with a certain activation value, and following the links emerging from these nodes, the activation can be propagated to the linked nodes (neighbors). In case of an associative network, the weighted links determine the amount of the activation energy a node is able to spread to a linked neighbor. This process can then be repeated for multiple iterations until some kind of stopping criterion is met.

In order to avoid the flooding of the network, which means that a large number of the network's nodes become activated, various constraints were introduced that limit the amount of activation energy spread from a source node to a target node [6].

In the remainder of this section the focus will be placed on the parts of the spreading activation process that are required for the *Semantic Pattern Transformation*. In order to gain a more detailed understanding of the algorithm, the

reader is referred to the work by Crestani et al. [15], who give an excellent overview on how spreading activation techniques are employed within information retrieval. Further details on applications that employ spreading activation are given in the last part of this section, and additional information regarding the influence of the various spreading activation parameters on the *Semantic Pattern Transformation* is presented in Section 7.3 of *Chapter 7 – Semantic Pattern Analysis*.

5.3.1 The Basic Spreading Activation Algorithm

Although there are many adaptations of the spreading activation algorithm focusing on different aspects of information retrieval tasks, the core process is always based on the same technique:

First, a set of nodes is activated by assigning an activation value that is higher than the threshold needed to activate a node. At this stage the spreading activation algorithm is applied, which spreads the energy of the activated nodes to their neighbors. Also, certain preprocessing steps can be applied in order to influence the impact of various nodes. **Second**, the spreading activation process is repeated for a fixed number of iterations, or until a certain criterion is met. **Third**, information is gained by analyzing the activation energies of the nodes within the network. Here, also certain post-processing steps can be applied in order to focus on specific aspects.

The steps required for the generic spreading activation process are shown in Algorithm 1 and described as follows:

- *Input:*
  - The associative network NET contains a set N with an arbitrary number of nodes n_i. Given two associated nodes n_i and n_j that are connected via a weighted link w_{ij}, then the existence of a link between two nodes indicates an association between these nodes. The assigned weight determines the strength of this association. In order to apply the spreading activation process, the following condition must be met for each weight: $0 < w_{ij} \leq 1.0$.
  - The set of nodes A that are considered as active according to some kind of threshold function.
  - Initial activation value IA: This value is used for the initial activation of the selected nodes. In order to cause the nodes to spread their activation energy, it must be higher than the activation threshold.
  - Set of spreading activation parameters P: These parameters influence the spreading activation algorithm and typically include basic parameters such as the decay factor, the activation threshold, the initial activation, or the number of iterations. More advanced parameters could be related to information retrieval or constrained-spreading, such as fanout factors, special spreading activation techniques etc.

- *Output:* The activation state of the associative network after applying the spreading activation process: *NET*.
- *Line 1 to 3*: The network is reset by setting the activation values of all nodes to 0.0.
- *Line 4 to 6*: The activation values of the given input nodes within the set A are set to the initial activation value IA. Typically, this value is the same for all the activated nodes, but for certain operations the value could depend on other operations that have been applied before the spreading activation process.
- *Line 7*: Preprocessing steps are applied to the initial activation state. For the basic spreading activation algorithm these are not needed, but, depending on the specific retrieval task, could include the application of fanout transformations, more complex activation functions or even the local application of spreading activation techniques.
- *Line 8 to 10*: Here, the spreading activation algorithm is applied for one or multiple iterations. The number of iterations depends on the employed stopping criterion, which could be a given maximum number of iterations, or a function that is fulfilled when a certain network state is achieved. In this thesis, all spreading activation processes are only applied for exactly one iteration.
- *Line 11*: Here, post-processing operations are applied to the activated network. For the basic spreading activation process this is not necessary, but could include the final application of fanout techniques, the application of filter operations that extract nodes according to various criteria, or the application of transformations that replace the activation values according to properties within the local neighborhood.
- *Line 12:* Finally, the desired information is extracted from the activated network via an arbitrary process. The nature of this process strongly depends on the desired information retrieval task.

Thereby, the details of each spreading activation iteration labeled as *spreadingActivationIteration* in Algorithm 1, are described in Algorithm 2 and explained as follows:

- *Input:*
  - The associative network *NET* in an arbitrary activation state.
  - The set of nodes A that are considered as active according to some kind of threshold function.
  - The set of spreading activation parameters P.
- *Output:* The activation state of the associative network after applying the spreading activation iteration: *NET*.

Algorithm 1 Basic spreading activation algorithm

Require:
NET {associative network}
$A = \{a_1, a_2, ..., a_n\}$ {set of nodes to be activated}
IA {initial activation}
P {set of spreading activation parameters}

```
1: for all a_i in NET do
2:    setActivation(NET,a_i,0) {reset network by setting all activation values to 0.0}
3: end for
4: for all a_i in A do
5:    setActivation(NET,a_i,IA) {set initial activation values for input nodes}
6: end for
7: preProcess(NET,P)
8: while stopping criterion not met do
9:    spreadingActivationIteration(NET,P)
10: end while
11: postProcess(NET,P)
12: retrieveInformation(NET)
```

- *Line 1 to 11*: For each node a_i in the set A the activation is spread to the neighboring nodes according to the following procedure.
- *Line 2 to 3*: The links L emanating from node a_i and the activation value S of node a_i are extracted.
- *Line 4-10*: For each emanating link w_{ij} in L, the target node a_j and the activation value A are extracted. Then, the activation energy T, which the target node a_j receives from a_i is calculated by $T = S * w_{ij} * D$. The received activation energy T is then added to the activation value A already contained in node a_j, and finally this new activation value is stored in the target node.

5.3.2 Constrained-Spreading

The basic application of the spreading activation algorithm without placing any constraints on the iterations or the amount of spread energy results in the flooding of the network. A good analysis of this problem is presented in [6], where the authors prove that unconstrained-spreading yields results that are independent of the set of initially activated nodes. In order to solve these problems, various operations that place constraints on the spreading of activation energy have been introduced.

Limited Distance or Iterations

The distance between two nodes within an associative network can be expressed by the minimum number of links that need to be traversed in order to get from

Algorithm 2 Spreading activation iteration

Require:
NET {associative network}
$A = \{a_1, a_2, ..., a_n\}$ {activated nodes}
P {spreading activation parameters.}
1: **for all** a_i in A **do**
2: L = getEmanatingLinks(a_i) {get emanating weighted links w_{ij} from a_i}
3: S = getActivation(a_i) {get activation value of source node a_i}
4: **for all** w_{ij} in L **do**
5: a_j = getTargetNode(w_{ij}) {get target node a_j of weighted link w_{ij}}
6: A = getActivation(a_j) {get activation value of target node}
7: $T = S * w_{ij} * D$ {calculate target activation value}
8: $A = A + T$ {add calculated value to activation value of target node}
9: setActivation(NET,a_j,A) {set new activation value of target node}
10: **end for**
11: **end for**

the first node to the second node. Obviously, nodes that have a direct link are closely related, whereas the degree of association drops with the distance between two nodes. In each spreading activation iteration, the maximum distance of reached nodes is increase by one. As a consequence, the unconstrained-spreading over several iterations reaches nodes that have no significant relations with the initial nodes. The adequate number of iterations depends on the information retrieval scenario but typically should be kept low in order to limit the spreading activation to nodes that have a high degree of association.

In the *Semantic Pattern Transformation*, the maximum number of iterations is set to one. The reason is that the associative networks employed within the transformation are highly connected, and the application of multiple iterations would cause the flooding of the network which disables the useful extraction of information.

Decay Factors and Activation Functions

Similar to limiting the number of iterations, a decay factor that attenuates the spread activation energy, can be applied. As described in Algorithm 2, the activation energy spread from a given source node to a target node is calculated by $T = S{*}w_{ij}{*}D$, where S is the activation value of the source node and w_{ij} is the weighted link between the source node and the target node. D is the mentioned decay factor, where $0 \leq D \leq 1$. In case of $D = 0.0$ activation spreading is completely suppressed, whereas $D = 1.0$ does not attenuate the spread energy at all. The best choice for D depends on the scenario, but typically values below 0.5 offer a good balance between emphasizing the initial nodes and allowing the neighboring nodes to retain some influence. When multiple iterations are used, the decay factor ensures that the spread activation energy drops with each iteration.

Fanout Factors

When a node is connected to a large number of nodes within the network, its activation and the subsequent application of the spreading activation algorithm causes the wide spread distribution of the activation energy. Such nodes have a large number of semantic relations to other concepts within the network and, thus, the knowledge gained by including them in the spreading activation process is low. An example for a highly connected node that does not carry valuable information is given in Example 6.

Example VI: Fanout and text analysis

The necessity for fanout factors can easily be shown by considering an example from text analysis. Assuming an associative network that models the relations between terms within sentences, and further assuming that stop-word-removal has not been applied to the analyzed text, then terms like *and*, *or*, or *is* remain in the text and are represented by nodes within the network. Since such terms are used within sentences together with arbitrary terms, they are connected to a large number of other term-nodes and their activation would cause the activation of large parts of the network. This flooding does not reveal important information, but on the contrary significantly reduces the quality of the extracted information.

In order to attenuate the influence of nodes with a large number of connections, the concept of fanout factors was introduced. Thereby, these factors can be utilized within different stages of the spreading activation process:

- **Before spreading**: Here, the initial activation values are attenuated before spreading activation is applied. In Algorithm 2 this would be introduced after *Line 3*.
- **During spreading**: Here, the activation energy, which is distributed to the neighboring nodes is attenuated. In Algorithm 2 this would be integrated into the calculation of the target node activation energy in *Line 7*.
- **After spreading**: Here, the activation values of the nodes after the spreading process are attenuated according to the fanout factors. In Algorithm 2 this would be executed after applying a spreading activation iteration (directly after *Line 10*.

The choice of the best fanout strategy and the calculation of the fanout values depends on the desired analysis and will be further discussed in Section 7.3.5 of *Chapter 7 – Semantic Pattern Analysis* .

Selection of a Path

This constraint is related to the selection of a preferred spreading activation path through the network. The benefit here is that the spreading of activation follows along a selected path while avoiding the activation of undesired regions. Since this constraint is only applicable when more than one iteration is involved, it is not applied within the *Semantic Pattern Transformation*.

5.3.3 Applications

Associative or semantic networks and the application of spreading activation for information retrieval play in important role in a wide range of systems.

A classic example for the utilization of a semantic network for information storage and retrieval is WordNet [25], which is a lexical database of the English language. It contains synonym sets (synsets) of similar words and relates these synsets via rich semantic relations. WordNet and other similar systems, such as Wiktionary[1] play an important role within text mining applications: [93], [92] and [58].

Further examples for the application of semantic networks and spreading activation include word sense disambiguation [88], the storage and retrieval of information within the semantic web [94], tag recommender systems [86], knowledge retrieval in large databases [32], recommendation systems based on ontologies [30], and analysis processes based on the semantic information stored within social networks such as Facebook, Twitter or Google+ [87], [23], [71].

Another very representative example for the utilization of semantic networks is the nowadays widely used semantic web, which represents the semantic relations between information as machine readable language. Within the semantic web umbrella, which is standardized by the World Wide Web Consortium (W3C), various concepts such as the Resource Description Framework (RDF) [59] or the Web Ontology Language (OWL) [70] for modeling ontologies play an important role. The *Semantic Pattern Transformation* has also been applied to data modeled by the RDF language. More information on the results will be presented in *Chapter 10 – Semantic Patterns – Applications*.

Especially, the reader is referred to two papers by Kozima et. al ([43] and [44]). This work is fundamental in the context of this thesis, because the basic idea for the *Semantic Pattern Transformation* was conceived after studying the author's approach to apply spreading activation techniques for calculating the similarity between english words. Thus, these two papers will be revisited in *Chapter 9 – Related Work*.

5.4 Discretization

The *Semantic Pattern Transformation* stores the analyzed feature values as nodes within an associative network. While categorical feature values can be

[1] http://www.wikitionary.org

directly mapped to network nodes, some kind of discretization operation must be applied to the numerical feature values. There are many discretization algorithms available including very simple methods that put the distance-based values into bins, to more complex methods based on entropy or fuzzy techniques [40], [91], [21]. However, due to the application of pre-spreading techniques, which will be discussed in Section 7.3 of *Chapter 7 – Semantic Pattern Analysis*, none of these methods, but the unsupervised RGNG algorithm was chosen for this task.

5.5 Machine Learning

Within the scope of this thesis, machine learning algorithms are mainly employed for the analysis of the *Semantic Patterns* in order to train supervised classifiers, or to recover previously unknown relations within the analyzed data. Thus, this section gives a short introduction to machine learning and the two main categories of machine learning algorithms – supervised and unsupervised algorithms. Later, the algorithms relevant for the *Semantic Pattern Transformation* will be presented and references to more detailed descriptions will be given.

The basic principles behind machine learning can be described by looking at the properties of the employed algorithms and the data analyzed by them:

- **Noisy data**: Due to the capability of machine learning algorithms to identify general properties of the analyzed data, the trained models are able to cope with noisy data.
- **Fuzzy nature**: Machine learning algorithms are capable to analyze fuzzy data, where the clear separation in categories or classes is not possible. In addition, machine learning algorithms are often applied, when simple decision or classification rules cannot be used due to the high complexity of the relations within the data.
- **Visualization and interpretation**: Certain machine learning algorithms employ models that are capable of visualizing high-dimensional data, which is of special benefit when previously unknown relations within the data need to be extracted.
- **Unknown relations**: Machine learning algorithms can always be applied to data, even when the relations between the features or the meaning of the features and their values are not known. Among the possibility to train classifiers on unknown data without the requirement to understand the data, the following main benefit can be observed: An analyzer is able to apply a machine learning algorithm in order to gain a better understanding of the relations within the analyzed data.
- **Empirical data**: Machine learning algorithms are often applied in order to understand relations within data, for which the complete underlying statistical model is now known. Often, this data is referred to as empirical

data because its features are based on information captured by arbitrary sensors.

The application of a machine learning algorithm is called training. Thereby, the algorithm's model is adapted according to the data within multiple iterations. This training, or adaptation process could also be described as learning, but then many philosophical questions about the nature of learning must be considered.

The two main categories of machine learning algorithms are defined as supervised and unsupervised learning. While unsupervised algorithms do not receive any additional information on the data during the training process, a supervised algorithm requires constant feedback. Supervised algorithms are also known as classifiers, because they are able to assign unknown instances to one of the classes they were trained to differentiate. Unsupervised algorithms are mainly employed when knowledge should be extracted from a data set, for which only limited or even none a priori knowledge is available.

The remainder of this section will introduce the key components and issues of supervised and unsupervised learning algorithms. For a thorough understanding of machine learning, the reader is referred to well established work within this area: [90], [22], [8].

5.5.1 Supervised Learning

A trained supervised algorithm or classifier is capable of analyzing unknown data and assigning it to one of the classes it learned to differentiate. Thereby, during the training phase the employed algorithm receives external information from a supervisor that influences and adapts the creation of the algorithm's model. This supervisor acts as a teacher during the analysis of the data and in most cases provides labels that assign the analyzed data to predefined categories or classes.

Supervised learning is very similar to human learning, where a teacher or another information source is available that guides the learning process. E.g., a child who learns the names of the colors (class labels) will get constant feedback from her parents, whether the spoken name matches the seen color. This feedback is used for learning and storing the appropriate patterns within the brain's neurons.

All human and machine based learning must consider one important aspect that is known as overfitting, which is explained for a human learner in Example 7.

Example VII: Learning multiplication rules

In real life, overfitting is related to the problem when a person commits something to memory without understanding the background of the learned concept. Subsequently, this person will not be able to draw general conclusions and infer high level rules that allow the correct identification of instances that have not been seen before, but are related to the learned ones. A good example is the learning and understanding of the mathematical multiplication rule versus committing the multiplication results of small numbers to memory. The latter can be considered as overfitting, because the learner highly specializes on the results without deriving general rules, and thus will fail to calculate the results for numbers that were not present during learning.

Although, in general, neither machine learning nor artificial intelligence are capable (yet) of inferring complex rules such as mathematical concepts, the problem of overfitting in machine learning has a strong relation to the example from above and must be considered whenever a machine learning algorithm is applied. During the training of a classifier, the machine learning algorithm tries to build a model that has a minimal classification error when applied to the labeled training data. Thus, in this respect an optimal classifier would be able to classify all the examples in the training set correctly. Unfortunately, the performance of the algorithm on the training data alone is very misleading, which is highlighted in Example 8.

Example VIII: A simple lookup table as classifier

In this example, a lookup table is used during the training process to map the analyzed instances to their corresponding class labels. Although, this simple classifier would be capable of classifying all examples within the training data without making any classification errors[a], it would fail in the classification of any new instance which was not present during the training phase, even when it only slightly deviates from the instances in the training set.

[a] Obviously, this only works when the same instance is not assigned to contradicting classes.

The problem of the lookup table is similar to that of the person who does not understand the concept behind the multiplication of numbers: The lookup table specializes on the training data and is not able to generalize. However, generalization is a key requirement of any learning scheme, because it enables the algorithm (and the person) to derive generic rules. These rules allow the correct classification of instances that deviate from those present during training, but are still considered similar according to some kind of relation. The problem of training an algorithm model that is not capable of generalizing, is called overfitting. It means that the algorithm specializes too much on the data during the training phase, or in other words over fits this data. In most cases this causes a drop in performance when applied to the real data. Unfortunately, the problem of overfitting is not only related to the nature of the algorithm model alone, but in most cases to the parameters used for the training of its model. Typically, in order to train an algorithm, multiple iterations are required that adapt the model step by step according to the training data. The adaptation of the model to the training data increases with the number of training iterations and thus the classifier achieves a higher classification performance on the training set. However, the likelihood of overfitting also increases. In order to cope with this problem, various training schemes have been developed (e.g., [9]). Although, there are many different versions of these schemes, in general they avoid overfitting by constantly evaluating the performance of the trained model on evaluation data that is not part of the training data. As soon as the performance drops on this evaluation set, the training scheme reasons that further training would over fit the training data and stops the training process. Further information on that substantial issue within machine learning can be found in Chapter 1 of the book by Witten et al. [90].

Supervised machine learning algorithms are based on a wide range of different techniques. Many of these techniques were inspired by the learning process of our brain and model some parts of its behavior. The most prominent example for such algorithms are artificial neural networks that model the structure of the neurons and their adaptation during learning [55].

Another very important example within the supervised category of machine learning algorithms is the Support Vector Machine. Its training process involves

the transformation of low dimensional data into a high-dimensional representation. The SVM then separates this data into different classes by using high-dimensional planes, which are referred to as hyperplanes [13]. This algorithm is used for the supervised evaluation of the *Semantic Patterns*.

5.5.2 Unsupervised Learning

As the name already suggest, there is no supervisor available during the training phase of an unsupervised learning algorithm. In the real word this corresponds to Example 9 where a botanist categorizes hitherto unknown objects without having a guide or supervisor that supports her in this process.

Example IX: Categorization of plants

A good example would be the categorization of plants by a botanist living in a time when only very limited information or prior work about plants was available. In order to find adequate categories, certain features of the plants such as size, shape, time for blossom, color etc. need to be derived. By using these features, their values and their combinations, the similarity of different plants can be calculated. This similarity is then used to assign the plants to their appropriate categories. A very important question is related to the number of categories or clusters, which are needed to represent the analyzed plants adequately. Obviously, only one cluster as it is represented by the term *plants* is not enough for more detailed considerations, whereas using a category for each single plant corresponds to the detailed identification and cannot be used for analyzing the relations between similar examples. The right number of categories or clusters depends on the problem to be solved, and the analyzed data.

In machine learning, unsupervised learning algorithms also analyze data described by instances and try to arrange this data in categories or clusters according to their similarity. Thereby, in most cases the algorithm model employs some kind of distance measure that enables the calculation of the similarity between arbitrary instances. This similarity is then used to arrange related instances into categories or clusters. The latter term is used more often, thus unsupervised learning is also often referred to as unsupervised clustering. In contrast to supervised learning, there is no evaluation set available that allows the employed training scheme to determine when the algorithm model starts to over fit. Here, the term overfitting refers to the training of a complex model that arranges the data into too many clusters. In order to cope with these problems, generic methods from information theory such as the Minimum Description Length (MDL) [5] are utilized that measure the balance between the model complexity and the error caused by the model. The model complexity is typically expressed by the amount of information that is needed to encode the model, whereas the error is

calculated by summing up the quantization errors (derived from the employed distance measures) of all instances in respect to their assigned cluster. The balance between the model complexity and the quantization error is highlighted in Example 10.

Example X: Model complexity in unsupervised learning

Considering a data set that is analyzed by an unsupervised algorithm, then the trivial model corresponds to representing all the analyzed data by only one cluster. Although, such a model would have a very low complexity – only one cluster needs to be encoded – the caused quantization error is at maximum. On the other hand a model that represents each instance within the data with a separate cluster has a very high complexity, which is equal to the encoding complexity of the raw data, but results in no quantization error at all. Thus, the MDL criterion is employed to find an adequate balance between the model complexity and the quantization error.

Unsupervised learning plays a vital role in analyzing data for which no a priori knowledge is available. By clustering the data according to its similarity, a quick superficial overview can be gained and the principle properties of the classes within the data set can be revealed. Clustering can also be understood as a method that decreases the amount of the analyzed and presented information. Due to this simplification, advantages for visualizing the often high-dimensional data can also be gained. Although there are methods like the MDL that aim to find the adequate number of clusters, in many applications this number still needs to be adapted according to the desired information and the available data. The analyzer can increase the number of clusters (the model complexity) when more detailed information is required or decrease it when a high level overview about the data is sufficient. Typically, the number of clusters found by the MDL criterion can thereby used as a starting point.

Similar to the supervised algorithms, there is a wide range of unsupervised learning algorithms that vary in the capability to handle categorical or numerical data, employ different distance measures and training procedures, differ in their capability to visualize high-dimensional data, and employ different strategies to arrange the data into clusters. Due to the processing steps, which are required for the setup of a machine learning algorithm and need to be adapted according to the desired goals and the nature of the analyzed data, the adequate choice of the appropriate algorithm is vital for the successful extraction of knowledge.

In order to cope with the different nature of the analyzed data, numerous unsupervised learning algorithms have been developed. Some important examples that handle numerical data are algorithms based on hierarchical clustering techniques (e.g. [35]), algorithms from the Neural Gas family [57], or the Self-Organizing Maps (SOMs) [39]. Other algorithms are also capable of handling categorical data and include COOLCAT [4], Kernel K-Means[14] or ROCK [33].

5.6 Adapting the Model Complexity

The Minimum Description Length (MDL) criterion plays an important role in determining the adequate complexity of a model trained by an unsupervised learning algorithm. Although the employment of the MDL criterion is highly beneficial within unsupervised learning, depending on the analysis more detailed models might be required. This is typically the case when a general overview about the data has been gained by analyzing an MDL based model. In order to increase the level of detail, more complex models might be trained on the whole data or a selected subset of the data.

Within the scope of this thesis, a simple method was used to define the model complexity based on the MDL criterion. Thereby, the complexity of an MDL optimal model is continuously increased until the desired complexity is reached. This principle is depicted in Figure 5.3 and explained as follows: Given the MDL value M of the model with the MDL optimal complexity, and an MDL override factor C, then the complexity of the model is increased until the overridden MDL value $M_c = M * C$ is reached.

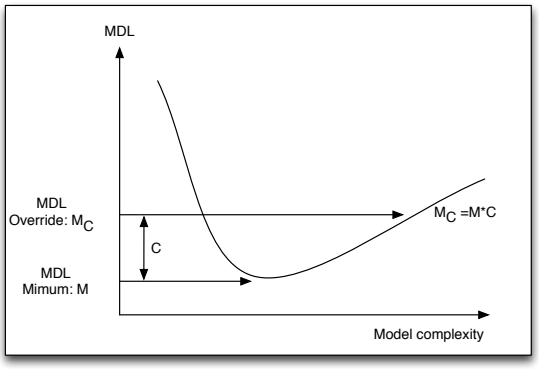

Figure 5.3: The optimal model complexity M is achieved when the MDL criterion is minimal. If a more detailed model is used, the model complexity is increased until the overridden MDL factor M_C is reached.

In this thesis, this procedure is used for two applications: **First**, in unsupervised clustering to specify the required level of detail, and **second**, for defining the complexity of the discretization model used by the *Semantic Pattern Transformation* to represent the analyzed feature values within an associative network.

5.7 Selected Unsupervised Machine Learning Algorithms

This section will focus on selected unsupervised algorithms that play a role within the *Semantic Pattern Transformation*, and the analysis of the thereby

gained *Semantic Patterns*. It must be noted that the *Semantic Pattern Transformation* is not limited to these algorithms and they can be replaced by others, without adapting the representation used within the *Semantic Patterns*.

The remainder of this section will first introduce the Vector Quantization (VQ) technique, which is the basis for the unsupervised Neural Gas algorithms employed in this thesis. Subsequently, the evolution from Neural Gas over Growing Neural Gas to Robust Growing Neural Gas will be explained.

5.7.1 Vector Quantization

Vector Quantization (VQ) [31] is technique initially used for data compression within signal processing tasks, however later was also utilized for training models within machine learning algorithms. The main idea of VQ is that a model comprised of a codebook of multiple prototype vectors describes the feature vectors of a given data set. Thereby, each prototype represents or compresses multiple instances or feature vectors. In the trivial case, when only one prototype is used, all feature vectors of the data set are represented by only this prototype. Each prototype covers an area that is called Voronoi space in which all feature vectors are mapped to the same prototype. This mapping is based on a distance measure that describes the similarity between prototypes and feature vectors. Typically, the Euclidean distance is used, but depending on the analyzed data other measures such as the Cosine similarity might also be employed. An example for a two dimensional data set, a VQ model trained on the data and the Voronoi spaces of the model codebook is presented in Figure 5.4. The mapping of multiple feature vectors to one prototype corresponds to a lossy compression function that induces quantization errors. For a given VQ model, the overall quantization error on a data set is calculated by summing up the distances of each feature vector to its corresponding prototype. This error is an indication for the quality of the VQ distribution.

Typically, there are two different problems that need to be solved by an algorithm that finds a VQ model for a given data set. **First**, given the number of codebook vectors, they must be placed in a way that minimizes the overall quantization error. **Second**, the model complexity, which is the number of codebook vectors, needs to be chosen. This can either be done manually by using a parameter that defines the number of codebook vectors, or automatically by employing a criterion like the MDL.

Vector quantization is the basis for many machine learning algorithms such as K-Means, Self-Organizing Maps and the family of Neural Gas based algorithms. An algorithm from the latter category is used for the *Semantic Pattern Transformation* and will be explained in detail in the following sections. While all of these algorithms solve the problem of minimizing the quantization error for a given model complexity, only a few sophisticated algorithms, such as the Robust Growing Neural Gas, employ measures like the MDL for finding an adequate balance between the model complexity and the quantization error.

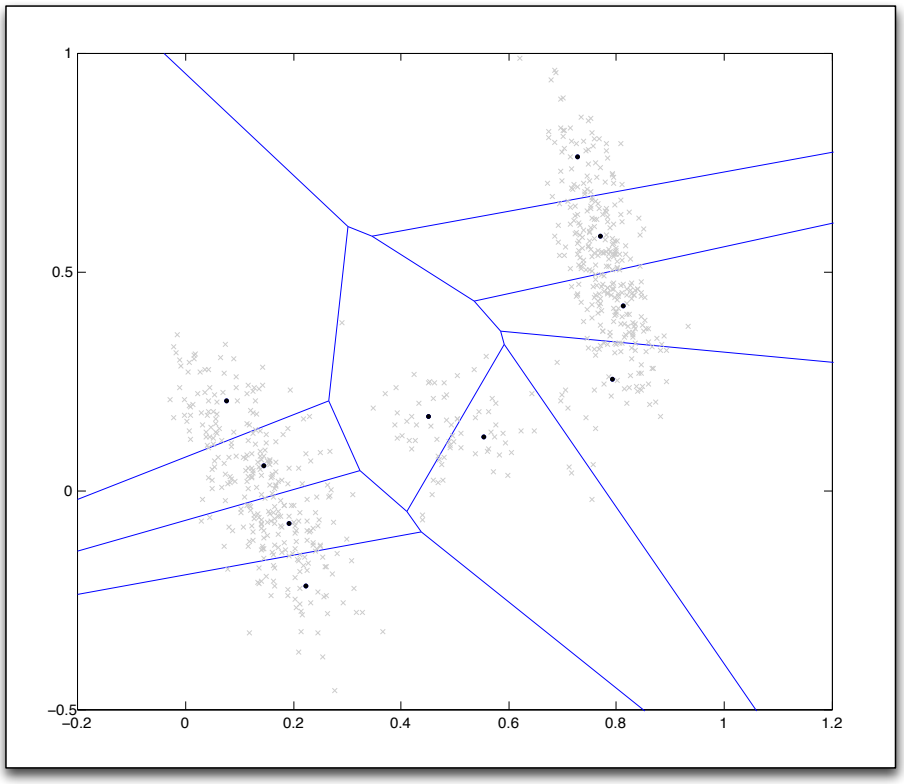

Figure 5.4: In this example, a Neural Gas algorithm with 10 prototypes was trained on a two-dimensional data set. For each of the prototypes (black dots) its corresponding Voronoi region (limited by the blue lines) is shown. Thereby, all of the feature vectors within such a region are represented (compressed) by the corresponding prototype.

5.7.2 K-Means and Self-Organizing Maps

The K-Means algorithm was originally introduced in 1967 by MacQueen [54] and is a widely employed unsupervised machine learning algorithm. Given the number of prototypes (the k in K-Means) and a data set arranged in feature vectors, then the algorithm minimizes[2] the overall quantization error by finding an adequate distribution of the k prototypes. The basic version of the K-Means algorithm is not capable of finding an adequate model complexity, and, therefore, was not used for the discretization operation within the *Semantic Pattern Transformation*. There have been many extensions to K-Means including the adaptation to handle categorical features [14], a fuzzy version capable of assigning instances to multiple clusters [7], the Expectation Maximization (EM) algorithm [19], which models the clusters as Gaussian mixture models, and fur-

[2] In general, only a local minimization is possible.

ther variants that automatically detect the correct number of cluster prototypes (e.g. [34]).

The Self-Organizing Map (SOM) is another unsupervised clustering algorithm based on VQ and was introduced in 1991 by Kohonen [39]. In addition to K-Means it also defines a strong neighborhood topology based on links between the prototype units. Due to the strength of this topology, the SOM enables the visualization of high-dimensional data sets on a two-dimensional map.

In Section 7.4.1 of *Chapter 7 – Semantic Pattern Analysis*, a simple example will be given where the SOM has been applied to an artificial data set. Also, the reader is referred to *Chapter 10 – Applications*, where the paper *Event Correlation on the Basis of Activation Patterns* is discussed [83]. Here, the SOM algorithm has been used for the unsupervised analysis of event correlation data within network intrusion detection.

5.7.3 Neural Gas Family

The basic Neural Gas algorithm was introduced by Martinetz [57] in 1991. Similar to K-Means and the Self-Organizing Map, all Neural Gas algorithms employ Vector Quantization in order to represent the analyzed data with prototypes. In the following sections three algorithms from the Neural Gas family will be described: the basic Neural Gas (NG) algorithm, its growing version the Growing Neural Gas (GNG) algorithm, and finally the Robust Growing Neural Gas (RGNG) algorithm that is employed by the *Semantic Pattern Transformation*. While the simple Neural Gas algorithm has a fixed number of clusters that needs to be passed as an algorithm parameter, GNG and RGNG start with two prototypes and grow the Neural Gas network according to the underlying data. In comparison to GNG, the main benefit of RGNG is the employment of the MDL criterion that determines the required number of clusters automatically.

Neural Gas

The Neural Gas algorithm was introduced by Martinetz et al. [57] in 1991 and is based on Vector Quantization. The learning process is similar to the Self-Organizing Map, however the link topology is dynamic as opposed to the strict topology of the SOM. This loose topology removes the capability to visualize high-dimensional data sets, but simplifies the training process and improves its quality due to the dynamic links.

- *Input:*
  - The data set D contains the instances described by feature vectors, for which the neural gas model is to be trained.
  - The number of iterations I determines how often the neural gas training process is applied to the data.
  - The learning rate α determines how strong the adaptation of a prototype is influenced by a feature vector.

  - The maximum link age T determines when a link between two prototypes is removed.
  - N is the fixed number of neural gas prototypes that should be trained for the given data.
  - LA is an N-dimensional matrix that contains the link age of each link.

- *Output:*
  - M represents the trained neural gas model and is a matrix that stores the neural gas prototypes. Thereby, each row represents the multi dimensional vector of a prototype. Therefore, the matrix has N rows and the number of columns is equal to the number of features contained in the feature vectors.
  - L is an N-dimensional matrix that contains the topological information of the neural gas model – meaning it keeps the state of the links between the various prototypes.

- *Line 1*: The prototype matrix M is initialized with N random prototypes stored within the rows. The dimension of the prototypes is equal to the dimensionality of the feature vectors within the input data D. Each dimension represents a feature that is used for the description of the data.

- *Line 2*: The link matrix L is a squared matrix with dimension N. In the initial state, there are no links between the prototypes.

- *Line 4 to 22*: For the given number of iterations I the following training procedure is repeated:

- *Lines 5 to 21*: In each training iterations all feature vectors of the data set D are used to update the neural gas prototypes according to the following procedure:

- *Line 6:* A feature vector d is randomly selected from the data set D and its distance to each neural gas prototype is calculated. The prototypes are then sorted according to their distance in P.

- *Lines 7 to 11*: The feature vector d is presented to the prototypes in P and the prototypes are updated according to their topological distance k to d. For the closest prototype p_0, the topological distance $k = 0$, for the second closest prototype p_1, $k = 1$ and so forth. The update rule moves the prototype p closer to d by calculating the vector $d - p$ and adding a fraction of this vector $\alpha * e^{\frac{-k_i}{\lambda}}$ to p. Thereby, the learning rate α and the topological distance k influence the size of the adaptation.

- *Lines 12 to 14*: These steps are used to update the link structure of the prototypes. First, a link is created between the two closes prototypes p_0 and p_1 if it does not yet exist. Second, the age of this link is set to zero and finally, the link age of all links is increased by 1.

Algorithm 3 Basic Neural Gas algorithm

Require:
D {data set consisting of feature vectors}
I {number of iterations}
α {learn rate}
T {maximum link age}
N {number of prototypes}
M {Neural Gas prototype codebook}
L {link matrix}
LA {link age matrix}

1: M = createRandomPrototypes(n) {create n number of prototypes with the same dimension as the input feature vectors and initialize the vectors with random values}
2: L = initLinkMatrix(n) {create squared link matrix of dimension n}
3: LA = initLinkAgeMatrix(n) {create squared link age matrix of dimension n and init all link ages with 0}
4: **for** $i = 1 \rightarrow I$ **do**
5: **for all** d in D **do**
6: P = calcDistance(d,M) {calculate distances to Neural Gas prototypes and sort them according to similarity}
7: $k = 0$ {Init topological distance $k = 0$. For the closest prototype p_0, $k = 0$, for the second closest prototype p_1, $k = 1$ and so forth.}
8: **for all** p in P **do**
9: $p = p + \alpha * e^{\frac{-k_i}{\lambda}}(d - p)$ {adapt all units according to their topological distance from d}
10: $k = k + 1$ {increase topological distance by 1}
11: **end for**
12: setLink(L,p_0,p_1) {take the two prototypes p_0 and p_1 with the smallest distance to d and create a link if it does not exist yet}
13: setLinkAge(LA,p_0,p_1,0) {set link age of link between p_0 and p_1 to 0}
14: increaseLinkAge(LA,1) {increase the link age of each link by 1}
15: **for all** p_i in P,p_j in P **do**
16: la = getLinkAge(LA,p_i,p_j)
17: **if** {$la > T$} **then**
18: removeLink(L,p_i,p_j) {Remove link between p_i and p_j if the link age is older than the given threshold}
19: **end if**
20: **end for**
21: **end for**
22: **end for**

- *Lines 15 to 20*: As the link structure changes over the training iterations, certain links might become invalid. This is indicated by an old link age. Therefore, in each iteration all links with an age larger than the given threshold T are removed.

5.7.4 Growing Neural Gas

The Growing Neural Gas (GNG) algorithm was invented by Fritzke [29] to overcome several shortcomings of the basic Neural Gas (NG) algorithm. During the NG training process the algorithm adapts the size of the topological neighborhood of a prototype, which defines an area in which neighboring prototypes are also included in the update process. The size of the neighborhood decreases over the iterations and gets limited to the immediate neighbors of the adapted prototype at the end of the training process. Here, the rationale is that during the early stages of the training process the influence of a feature vector on the prototypes updates should be larger, because the untrained prototypes are still located at non-adapted locations with large quantization errors. During the training process, the prototypes get trained and are relocated to locations where the feature vectors are represented more accurately, which is indicated by lower quantization errors. Since these trained positions get closer to the local quantization error minima during each iteration, only the feature vectors in the immediate neighborhood should be considered in order to optimize the final prototype locations. While this adaptable training process is reasonable, it comes with a price: In order to decrease the size of the adapted neighborhood the number of iterations needs to be given as input parameter. Thus, the training process is adapted to this number of iterations and during the training process changes cannot be made to the data set. Therefore, NG can only be applied in static environments where the data within the analyzed data set does not change.

The GNG algorithm addresses this problem by implementing an iteration independent adaptation process that is capable of following dynamic data. The "growing" nature of the GNG model also has another advantage that is not used by GNG directly, but by other more sophisticated algorithms based on GNG. The complexity of the GNG model increases over the training process. Thereby, the models, which are trained during the initial stages of the training process, describe the superficial aspects of the analyzed data, whereas the more detailed ones trained in later iterations cover more and more details of the data set. The information about the changing model complexity and its influence on the quantization errors can be utilized by criteria like the MDL to automatically find a model that keeps the balance between complexity and quantization errors. However, since the complexity of the standard NG algorithm is defined as parameter – as in K-Means – and thus does not change over the training iterations, criteria like the MDL cannot be deployed directly within the NG training process.

- *Input:*
  - The data set D contains the instances described by feature vectors, for which the neural gas model is to be trained.
  - The stopping criterion SC determines when the training process stops. This could be a simple criterion like the maximum number of prototypes, or a sophisticated one such as the Minimum Description Length.

  - The learning rates α_1 and α_2 determine how strong the adaptation of a prototype is influenced by a feature vector.
  - The maximum link age T determines when a link between two prototypes is removed.
  - The error constants E_1 and E_2 are used to decrease the overall errors when a new unit is inserted and at the end of each iteration.
  - The link matrix L and the link age matrix LA contain the information about links and their ages.
  - The array *error* contains the accumulated quantization errors of each codebook prototype.

- *Output:*
  - M represents the trained neural gas model and is a matrix that stores the neural gas prototypes. Thereby, each row represents the multidimensional vector of a prototype. Therefore, the matrix has N rows and the number of columns is equal to the number of features contained in the feature vectors.
  - L is an N-dimensional matrix that contains the topological information of the neural gas model – meaning it keeps the state of the links between the various prototypes.

- *Line 1*: The initial codebook M containing two prototypes at random locations is created.

- *Line 2*: The structures required by the algorithms are created. These are the link matrix L, the link age matrix LA and the array *error* that contains the accumulated errors of each prototype. Thereby, in the initial state links do not exist, and the link ages and the quantization errors are set to zero.

- *Line 3*: The iteration counter c is initialized. This counter is needed for determining when a new unit should be inserted and ensures that there is an adequate time interval between two insertions.

- *Lines 4 to 45*: The complete Growing Neural Gas process consists of training the existing prototypes and inserting new ones at locations with high quantization errors. The training and insertion processes continue until some stopping criterion SC is met.

- *Line 5*: The iteration counter c is increased.

- *Line 6*: The prototypes p_0 and p_1 with the smallest and the second smallest distance to d are extracted. In algorithms based on VQ, the closest prototype is also called the best matching unit (BMU). The distance calculation depends on the employed distance measure such as the Euclidian distance or the Cosine similarity.

- *Line 7*: The quantization error for unit p_0 is calculated (based on the employed distance measure) and added to the accumulated *error* of p_0. These error values are required for determining where a new unit needs to be inserted.
- *Line 8*: A link is created between p_0 and p_1 if it does not yet exist.
- *Lines 9 to 12*: The link age of all links emanating from p_0 is increased by 1.
- *Lines 13 to 16*: The update rule moves the prototype p_0 closer to d by calculating the vector $d-p_0$ and adding a fraction of this vector $\alpha_1 \cdot (d-p_0)$ to p. The learning rate α_1 ensures that the size of the update step is limited. The same procedure is applied to the topological neighbors of p_0, but here a lower learning rate α_2 is used.
- *Lines 17 to 23*: After setting the age of the link between p_0 and p_1 to zero, the age of all links is checked. If the link is older than the age threshold T, it is removed.
- *Lines 24 to 30*: Units that are not linked to other units are considered as dead units, because they do not represent any feature vector. Since they only increase the complexity of the model, and do not reduce the quantization errors they are removed from the model.
- *Lines 31 to 43*: After the training process, the GNG algorithm checks whether a new prototype should be inserted. If the input-stimuli counter c is an integer multiple of the parameter λ, a new unit is inserted. Thereby, the insertion process **first** extracts the prototype *pe* that has the maximum accumulated quantization error (stored in *error*), **second** extracts the topological neighbor *pn* that has the maximum quantization error of all topological neighbors PN, **third** inserts a new unit at the half distance between *pe* and *pn*, and **finally** updates the link and link age matrices, and the quantization errors. The accumulated error of *pn* and *pe* is decreased by a multiplication with the parameter E_1 and the error of the newly inserted unit is set to the just decreased error of *pe*. In addition, the now obsolete link between *pe* and *pn* is replaced with two links between *pe* and *pi* and *pn* and *pi*.
- *Line 44*: The quantization errors of all units are decrease by multiplying them with the parameter E_2.

5.7.5 Robust Growing Neural Gas Algorithm

The Robust Growing Neural Gas (RGNG) algorithm presented by Qin et al. [65] is a member of the robust neural gas algorithms (e.g., [1]) and implements various measures to improve the robustness of the neural gas learning process.

Thereby, the authors base their definition of robustness on the principles defined by Huber in [38] for statistics and used as fundamental properties for robust clustering algorithms by Dave et al. in [16]. A robust procedure has the following properties:

- The efficiency or accuracy should be good for the assumed model.
- The negative impact in the performance caused by small deviations from the model assumption should be small.
- The impact of larger deviations from the model assumption should not have a catastrophic impact.

While Neural Gas algorithms fulfill the first property, they typically have issues related to the second and third property. These are the sensitivity to the initialization of the prototypes and the drop in performance due to the presence of a large number of outliers. Another issue which is not directly related to robustness, is the determination of the appropriate model complexity. Since clustering is an unsupervised learning process, class labels are not available and the number of clusters must either be given as parameter or automatically determined by the algorithm. For most of the NG and GNG variants, the number of clusters cannot be determined automatically. This is a major disadvantage when data needs to be analyzed for which only limited or none a priori knowledge is available. In order to overcome the robustness issues, and to automatically determine the adequate model complexity, RGNG implements several extensions to the standard GNG algorithms:

- Employment of the Minimum Description Length (MDL) for choosing an appropriate model complexity.
- Integration of an outlier resistance strategy that detects and attenuates the influence of outliers, even when a large number of them is present within the input data.
- Support for adaptive learning rates that are set according to the age of the model prototypes. Thereby, the learning rate is lower for older prototypes within the model, which are already trained on the cluster they represent, and higher for newly inserted prototypes that still need to be adapted to the represented data.
- In order to avoid the prototype coincident problem, which occurs when two prototypes represent the same cluster, RGNG employs a new learning rule that introduces a repulsion force (initially presented in [20]) that moves closely situated prototypes apart.

Although all of these sophisticated procedures help to improve the performance over the standard NG and GNG based algorithms, the most important aspect for the *Semantic Pattern Transformation* was the inclusion of the MDL

criterion by RGNG. The reason here is that the *Semantic Pattern Transformation* is often used in scenarios where none or only limited a priori knowledge is available about the analyzed data. Thus, the manual definition of the model complexity is not feasible, and the model complexity determined by the MDL within RGNG is a good starting point for further analysis.

5.7.6 Expectation Maximization Algorithm (EM)

This algorithm is used in *Chapter 8 – Evaluation* for the evaluation of the *Semantic Patterns* within unsupervised learning. The *EM* algorithm is based on Gaussian mixture models for representing the analyzed data. When compared to the simple centroid models used by *K-Means* and the *Neural Gas* algorithm family, the deployment of these Gaussian kernel allows for a more accurate modeling of the underlying data.

Since this algorithm is not directly employed by the *Semantic Pattern Algorithms*, further details will not be discussed here, and the reader is referred to this explanation [18] for further information.

5.8 Selected Supervised Machine Learning Algorithms

The evaluation of the *Semantic Patterns* presented in *Chapter 8 – Evaluation* is based on two supervised learning algorithms: The Support Vector Machine algorithm, and the J48 algorithm. The latter is an implementation of the well-known C4.5 decision tree algorithm.

Both algorithms are not employed directly by the *Semantic Pattern Transformation*. Therefore, the reader is referred to external sources for further information: SVM algorithm [13], C4.5 algorithm [67]. General information about supervised classification techniques can be found in the work by Kotsiantis [41].

5.9 Chapter Conclusions

The *Semantic Pattern Transformation* transforms arbitrary value-centric feature vectors into a common semantic representation – the *Semantic Patterns*. This transformation process is based on associative networks, the application of spreading activation, and the application of supervised and unsupervised machine learning algorithms.

While this chapter focused on the details of these methods, the following chapters will discuss how they are arranged and applied within the whole transformation process.

Algorithm 4 Growing Neural Gas (GNG) algorithm

Require:
D {data set consisting of feature vectors}, SC {stopping criterion}, α_1, α_2 {learn rates}, T, E_1, E_2 {Maximum link age T and error constants E_1, E_2}, L, LA, *error* {link matrix, link age matrix, quantization error array}, M {Neural Gas prototype codebook}

M = createPrototypes(2) {Initialize random codebook}
{L, LA, *error*} = initStructures(2) {Initialize link matrices and error array}
$c = 0$ {Input-stimuli counter}
while *SC() != true* **do**
  $c = c + 1$ {Increase input-stimuli counter}
  $\{p_0, p_1\}$ = getBMUs(d,M,2) {Get the two best matching units p_0 and p_1}
  error(p_0) = *error*(p_0) + $||d - p_0||^2$ {Calculate and update quantization error}
  setLink(L,p_0,p_1) {create a link between p_0 and p_1 if it does not exist yet}
  EL = getEmanatingLinks(L,p_0) {Get emanating links from p_0}
  for all *el* in EL **do**
    increaseLinkAge(LA,*el*,1) {Increase the link age of each link *el* by 1}
  end for
  $p_0 = p_0 + \alpha_1 \cdot (d - p_0)$ {Move p_0 into the direction of d}
  PN = getNeighbors(L,p_0) {Get all topological neighbors from p_0}
  for all *pn* in PN **do**
    $pn = pn + \alpha_2 \cdot (d - pn)$ {Move each neighbor *pn* into the direction of d}
  end for
  setLinkAge(LA,p_0,p_1,0) {set link age of link between p_0 and p_1 to 0}
  for all {p_i in M},{p_j in M} **do**
    la = getLinkAge(LA,p_i,p_j) {Get and check link age of each link}
    if {*la*> T} **then**
      removeLink(L,p_i,p_j) {Remove link between p_i and p_j}
    end if
  end for
  for all p in M **do**
    EL = getEmanatingLinks(L,p) {Get emanating links of p}
    if {$EL = \{\}$} **then**
      removePrototype(p) {Prototype with no links are removed}
      updateStructures(L,LA,*error*) {Adapt structures}
    end if
  end for
  if {mod(c,$\lambda = 0$} **then**
    pe = getUnitMaxError(*error*) {Get prototype with the maximum error}
    PN = getNeighbors(L,*pe*) {Get the topological neighbors of *pe*}
    pn = getUnitMaxError(*error*,PN) {Get neighbor with the maximum error}
    pi = $0.5 \cdot$ (*pe* + *pn*) {Create a new prototype between *pe* and *pn*}
    updateStructures(L,LA,*error*) {Adapt structures}
    removeLink(L,*pe*,*pn*) {Remove link between *pe* and *pn*}
    createLink(L,*pe*,*pi*) {Create link between *pe* and *pi*}
    createLink(L,*pn*,*pi*) {Create link between *pn* and *pi*}
    error(pe) = *error(pe)* $\cdot E_1$ {Reduce error or *pe*}
    error(pn) = *error(pn)* $\cdot E_1$ {Reduce error or *pn*}
    error(pi) = *error(pe)* {Set initial error value of *pi* to value of *pe*}
  end if
  error = *error* $\cdot E_2$ {Decrease all errors by factor E_2}
end while

6 Semantic Pattern Transformation

This chapter describes the details of the *Semantic Pattern Transformation*. The process is organized in five layers that extract the feature values and their semantic relations from the analyzed instances, train an associative network and apply the spreading activation algorithm to generate the *Semantic Patterns*. Each of these processes is explained in detail and relevant examples are given.

6.1 Semantic Pattern Transformation

The process of generating and analyzing the *Semantic Patterns* is separated into five processing layers depicted in Figure 6.1. The general idea is to extract the co-occurrence information of different features (*Layer 1*), to store this information in an associative network (*Layer 2 and Layer 3*) and to generate *Semantic Patterns* by applying spreading activation (SA) strategies (*Layer 4*). Various analysis techniques can then be applied to the generated patterns (*Layer 5*). The following sections define the required concepts and describe the five processing layers.

6.1.1 Layer 1 - Feature Extraction

The first processing layer extracts all features and their corresponding values from the instance set generated via an arbitrary relation from a domain-specific data set. Depending on the nature of the information carried by the features, they can be assigned to two categories – *symbolic features* and *distance-based features*. This distinction plays an important role for the representation of data

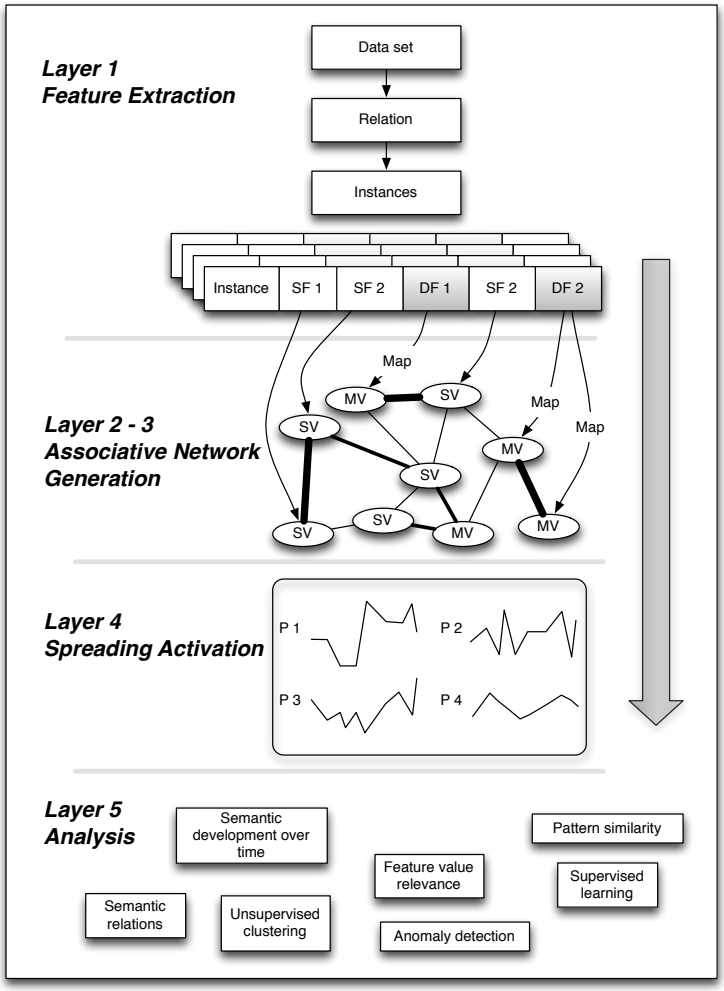

Figure 6.1: Overview of the five processing layers of the *Semantic Pattern Transformation*.

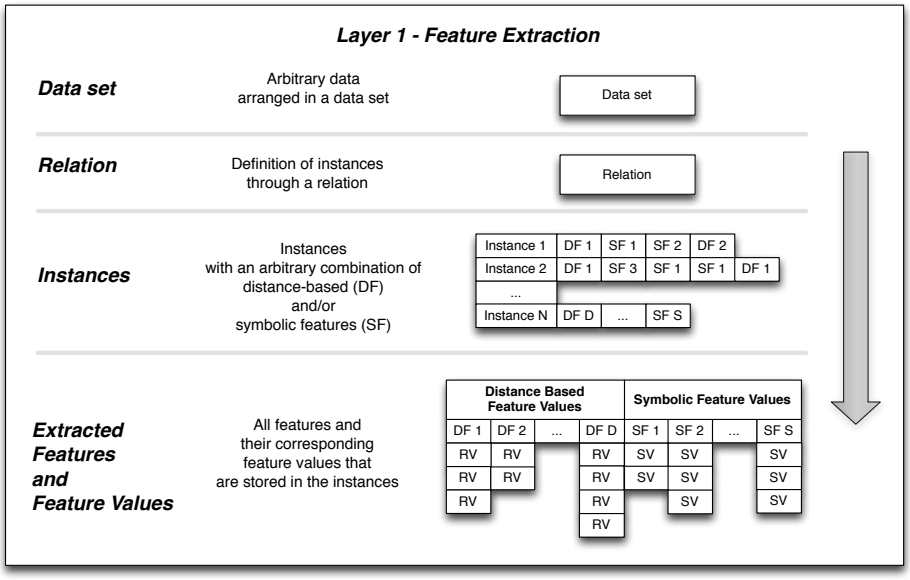

Figure 6.2: *Layer 1*: The features and their unique feature values are extracted from the instances and the feature type is determined for each feature. This information is required for all subsequent processing layers.

within the associative network generated in *Layer 2* and *Layer 3*. After extracting all features, their unique feature values, and determining their category, this information is stored in a table that is used within all subsequent processing layers.

Properties of Instances, Features and Feature Values

A set of multiple instances is extracted from an arbitrary data set via a defined relation. These instances are described via different features and their respective feature values. Thereby, the following properties regarding the *feature type*, the *context between feature and feature value*, and the *number of features used within an instance* need to be considered:

Each feature and all its associated features values can either be assigned to the symbolic or the distance-based feature type. This distinction plays an important role for the generation of the associative network in *Layer 2* and *Layer 3*. Although the categorization could be partly automatized, for certain distance-based features the overall performance can be improved by interpreting them as symbolic features. Typically, this is the case when the number of possible feature values is low, or single or multiple feature values carry information that might be lost when using the distance information as basis for the mapping operation in *Layer 2*. These issues are highlighted in Example 11.

Example XI: Symbolic vs. distance-based features?

Given a data set, which contains information about a wide range of airplanes, and a numerical feature that describes the number of engines an airplane carries: Then, the possible feature values would range from one engine for small aircraft, to six engines used to power the world's heaviest airplane[a]. Assuming, this number is interpreted as a distance-based feature, then, the following information is gained by applying the Euclidean distance measure: The distance between aircraft with four engines is equal to those with two or six engines.
Now, when taking other information into account, one would come to the conclusion that typical civilian passenger airplanes are powered by either two or four engines, whereas six engines are mainly used by special purpose aircraft. Hence, in terms of usage, aircraft with four engines are closer related to those with two than to those with six engines. This leads to the conclusion that in this case the information carried by the specific values is more important than that carried by their distances. Therefore, the feature should be assigned to the symbolic category.

[a] Antanov 225, "the six-engined dream": http://en.wikipedia.org/wiki/Antonov_An-225)

Regardless of the feature type, each feature consists of one or more feature values. Thereby, it is important to note that the feature value only conveys information when viewed in context with its associated feature. This is especially important when multiple features share the same feature values or different distance-based features are used. The issue is highlighted in Example 12.

Example XII: Context of features and feature values?

The features *Import* (import commodities) and *Export* (export commodities) of a country are described by the same feature values such as *iron*, *chemicals* or *coffee*. Although the same values are used for both features, the information whether a country imports or exports *coffee* has a significant influence on other features. Furthermore, by looking at the feature values without knowing the associated feature, it is not clear for which description they are used. The value *coffee* could either belong to the stated *Import* or *Export* features, or to a completely different feature that describes the items on the menu of a restaurant.
In case of distance-based features this is even more important, since a value alone does not convey information without knowing its associated feature. Therefore, a feature value always must be viewed in context with its associated feature.

Each instance is described with one or multiple features. The number of features used for the description may vary from instance to instance. When

looking at all the input instances, neither the number of features per instance nor the used features need to be constant. Furthermore, the same feature can be used multiple times within one instance. In machine learning, these properties and the necessity to handle missing values need to be considered when choosing the desired algorithm and the appropriate preprocessing operations. This is highlighted by Example 13. For further information the reader is referred to the detailed description of the processing steps required for the application of machine learning algorithms in *Chapter – 3 – Knowledge Discovery and Machine Learning*.

Example XIII: Features for the description of an instance?

Demo Data Set 1 contains various features and their values for the description of the world's countries. Although, a large number of features are shared by all countries, some of them are only applicable to certain countries, such as the size of the claimed maritime area or features describing nuclear power plants[a]. Furthermore a feature might occur multiple times within the same instance, which is highlighted by the export commodities of a country. This feature is available for all country instances, but its number of occurrences varies from country to country. Due to these issues, an adequate representation must be found that can be handled by the employed machine learning algorithm.

[a] In some cases the absence of features might be indicated via a "null" value like *none* or *0.0*, but in general non appropriate features are simply not present.

Grouping of Distance-Based Features

Distance-based features might also be arranged into groups. This grouping has a positive influence on the performance of the required mapping operation applied in the next layer. However, this operation is not specifically required by the transformation process, and special care must be taken that only compatible features are grouped. Otherwise, preprocessing operations such as normalization would be needed. This would defeat the purpose of the *Semantic Pattern Transformation*, which aims to eliminate these operations. One possible application of grouping is highlighted in Example 14.

> ***Example XIV: Grouping distance-based features?***
>
> Within *Demo Data Set 1*, each country has three distance-based features describing the percentage of its gross domestic sectors – agriculture, industry and services. The values are expressed as percentage values and the sum over these three features is equal to 100%. When comparing the feature values, they have the same value range (0% to 100%), and the typical distances between the feature values of all three features can be considered as similar. Therefore, these features could be grouped without applying additional preprocessing operations.
> In contrast, the following example highlights that grouping might also lead to the requirement of further preprocessing operations. The feature birth-rate, which is expressed as births per 1000 people, and the feature Gross Domestic Product (GDP) per capita (in U.S. dollars) are based on different value ranges. In order to eliminate these differences, normalization procedures are required. However, as the *Semantic Pattern Transformation* aims to remove these operations, the grouping of such features is considered as counterproductive.

Layer 1 - Processing Steps

The processing steps executed in *Layer 1* are defined via the pseudo algorithm shown in Algorithm 5 and described as follows:

- *Input:* The instance set (I) containing and arbitrary number of instances.
- *Output:* The network's meta-information structure NET_I containing all features F, their corresponding unique feature values FV and their feature types C.
- *Lines 1 to 7*: The desired features F and their values FV are extracted from the instances within the instance set I, which are generated from an arbitrary data set via a relation.
- *Lines 8 to 11*: Depending on the information carried by the features, they are assigned to two possible feature types C – *distance-based or symbolic*. The features, their defined categories and their values are then stored

Algorithm 5 *Layer 1*: Feature Extraction

Require:
$I = \{i_1, i_2, ..., i_n\}$ {set of instances}
NET_I {network meta-information structure}
1: **for all** i_i in I **do**
2: F = extractFeatures(i_i) {extract all features from the given instance}
3: **for all** F_i in F **do**
4: FV = extractFeatureValues(F_i,i_i)
5: storeUniqueValues(*NET_I*,F_i,FV) {add features and feature values to the network's meta-information structure, only unique values are added}
6: **end for**
7: **end for**
8: **for all** F_i in *NET_I* **do**
9: C = determineFeatureType(F_i) {determine feature type manually or automatically by inspecting the feature values associated with the given feature}
10: storeFeatureType(*NET_I*,F_i,C) {store feature type in the network's meta-information structure}
11: **end for**
12: DF = getDistanceBasedFeatures(*NET_I*) {optional: get distance based features}
13: *groups* = createGroups(DF) {optional: manually create groups}
14: storeGroups(*NET_I*,*groups*) {optional: store the group information within the network's meta-information structure}

for the subsequent *Layer 2* processing in the network's meta-information structure *NET_I*.

- *Line 12 to 14:* Optionally, multiple distance-based features that have similar value ranges could be arranged into *groups*. This group information is then stored within the network's meta-information structure *NET_I*.

6.1.2 Layer 2 - Node Generation

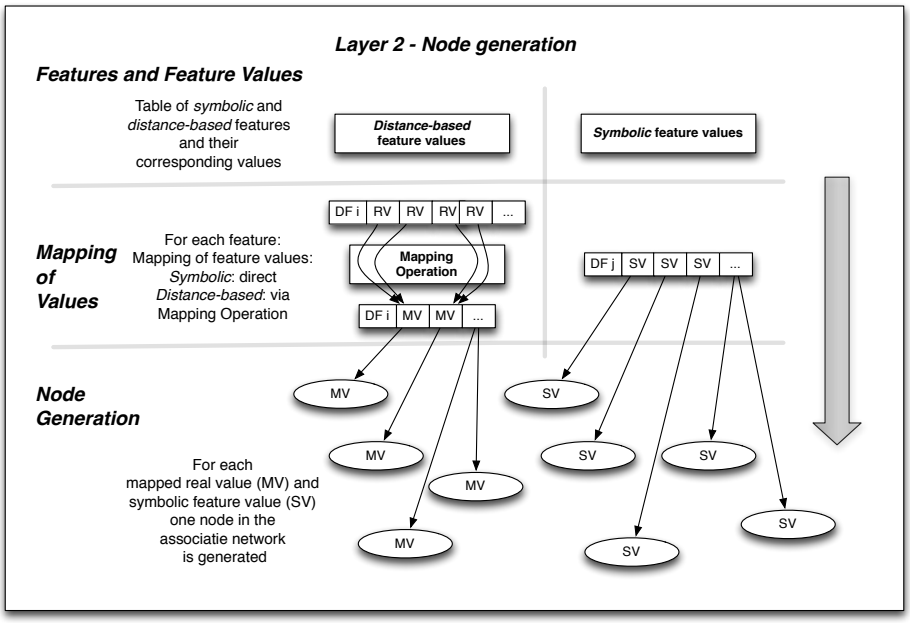

Figure 6.3: *Layer 2*: The mapping of symbolic and distance-based feature values to network nodes based on the feature type.

Within an associative network, information and relations are modeled as nodes and links connecting these nodes. This processing layer – depicted in Figure 6.3 – creates the nodes of the network by mapping the previously extracted feature values to network nodes. Thereby, the feature type – symbolic or distance-based – plays an important role and determines how this mapping is realized.

Symbolic and Distance-Based Features

For *symbolic* features the possible feature values are directly mapped to distinct nodes by creating one node for each feature value. Thereby, a node representing a symbolic feature value is uniquely defined by three properties: The name of the corresponding feature, the feature value and the feature type.

For *distance-based* features, the process must be altered due to the following reasons: **First**, the associative network would grow too large if each real feature value occurring in the analyzed instances would be mapped to a distinct node. **Second**, and even more important, the input range covered by *distance-based* values might not be countable and therefore cannot be directly translated into nodes. The latter would result in the inability to map deviating feature values of instances that where not available during training. Therefore, some kind of

operation that maps multiple input values or a value range to a corresponding node within the network, is needed.

Mapping Distance-Based Feature Values

The mapping operation can be implemented via a discretization algorithm. Although there is a broad range of such techniques available, the decision was made to use a different approach taken from the area of machine learning – the unsupervised clustering algorithm RGNG. Although, the application of an unsupervised learning algorithm typically comes with a higher complexity than the utilization of discretization algorithms, it offers several advantages in the area of anomaly detection and the application of spreading activation (SA) techniques. In other more specific scenarios, the RGNG algorithm could be replaced with an adequate discretization method. The unsupervised mapping operation is obviously not limited to the RGNG algorithm – basically any unsupervised clustering algorithm could be used for the discretization process. However, the RGNG algorithm was selected, because it includes several robust learning techniques and employs the Minimum Description Length (MDL) [68] to automatically determine the model complexity. Since the performance of RGNG and similar algorithms has already been evaluated on a wide range of data sets, one can safely assume that these algorithms will produce good results for the one-dimensional data represented by single features.

For the application of the RGNG algorithm, the following processing steps are required: For each distance-based feature, the algorithm is applied to the corresponding unique feature values and a model is trained. In case of RGNG, this model contains cluster prototypes that are directly mapped to nodes within the associative network. Thus, for each of these prototypes a node is generated and the feature values within the analyzed data are assigned according to the minimum distance to these prototypes. The values of the prototypes, and therefore the covered value range depends on the location of the agglomeration of input values (clusters) (Example 15). If a number n of distance-based features were grouped in the previous step, the RGNG algorithm is applied to the n-dimensional feature vectors consisting of the grouped feature values.

Example XV: Mapping of a distance-based feature

Figure 6.4 shows an example for mapping the feature values ranging from 0 to 100 of a distance-based feature to associative network nodes. Here, the application of the RGNG algorithm as mapping operation generates a model with four prototypes (clusters).

Using Distance Information

When training an RGNG map for a distance-based feature or a group of such features, additional information is gained that is not directly used for building

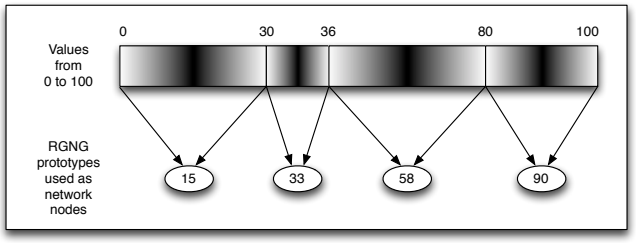

Figure 6.4: Mapping of values ranging from 0 to 100 to prototypes for a *distance-based* feature. The density of values is indicated by the color ranging from white (low density) to black (high density). The input value clusters are represented by RGNG prototypes, which are used as nodes within the network.

the associative network, but plays an important role for spreading activation techniques applied in *Layer 4*. This is discussed in detail in Section 7.3.3 of the next chapter.

First, when feature values are used to train an RGNG map, the distances between these values and the best matching prototypes can be determined. When analyzing the distances between the feature values and their corresponding prototypes, the mean distance and the variance of these values can be extracted. This information can be used later for anomaly detection by attenuating the influence of feature values that are not typical for a given node.

Second, the distance between the trained prototypes themselves can be calculated. This information is used within the spreading activation process in *Layer 4* to cause the co-activation of nodes representing feature values that are close to the input values.

Further details on how this information is structured and how it is used will be given in the next chapter.

Layer 2 - Processing Steps

The processing steps executed in *Layer 2* are defined in Algorithm 6 and described as follows:

- *Input:* The network's meta-information structure NET_I currently containing all features F, their corresponding unique feature values FV and their feature types C.
- *Output:*
  - The initial associative network NET containing the nodes, which are created by applying a mapping operation to the feature values within NET_I.
  - Additional meta-information is stored within the structure NET_I.

- *Line 2 to 3*: The feature values FV and the feature type C are extracted for each feature F contained in NET_I and extracted in *Layer 1*.
- *Line 5 to 8*: For each symbolic feature F, its feature values FV are extracted, and for each unique feature value FV_j a node a within the associative network NET is created. Thereby, a node is described by three components: The feature F, feature value FV and the feature type C.
- *Line 10 to 11*: For each distance-based feature F, a discretization operation is applied to its unique feature values FV. Here, the operation is based on the training of an RGNG map (*model*). The prototypes PR defining this map are extracted.
- *Line 12 to 15*: For each of the map's prototypes PR, a node consisting of the feature F, the value of the prototype pr_i and the feature type C is created within the network (NET).
- *Line 16 to 17:* Information related to *distances* within the RGNG map and between the feature values, and their best matching prototypes is stored within the network's meta-information structure NET_I. This information is required for certain subsequent analysis processes.

Algorithm 6 *Layer 2* - Node generation

Require:
NET {associative network}
NET_I {network meta-information structure}
1: **for all** F_i in NET_I **do**
2: FV = getUniqueFeatureValues(NET_I,F_i) {get all unique feature values from the network's meta-information structure}
3: C = getFeatureType(NET_I,F_i) {get type of feature: symbolic or distance-based}
4: **if** {C = *"symbolic"*} **then**
5: **for all** FV_j in FV **do**
6: a = createNode(F_i,FV_j,*"symbolic"*) {symbolic feature values are directly mapped to network nodes, a node is identified by the represented feature, its feature value and its feature type}
7: addNode(NET,a) {the node is added to the network}
8: **end for**
9: **else if** {C = *"distance-based"*} **then**
10: $model$ = trainDiscretizationModel(FV) {train discretization model on feature values for the given distance-based feature}
11: PR = getPrototypes(model) {extract prototypes from discretization model}
12: **for all** pr_i in PR **do**
13: a = createNode(F_i,pr_i,*"distance-based"*) {a node for a discretization prototype is identified by the represented feature, the value of the prototype and the feature type}
14: addNode(NET,a) {the node is added to the network}
15: **end for**
16: $distances$ = getDistances($model$) {extract distance information from trained model}
17: storeDistanceInformation(NET_I,$distances$) {store distance information in the network's meta-information structure}
18: **end if**
19: **end for**

6.1.3 Layer 3 - Network Generation

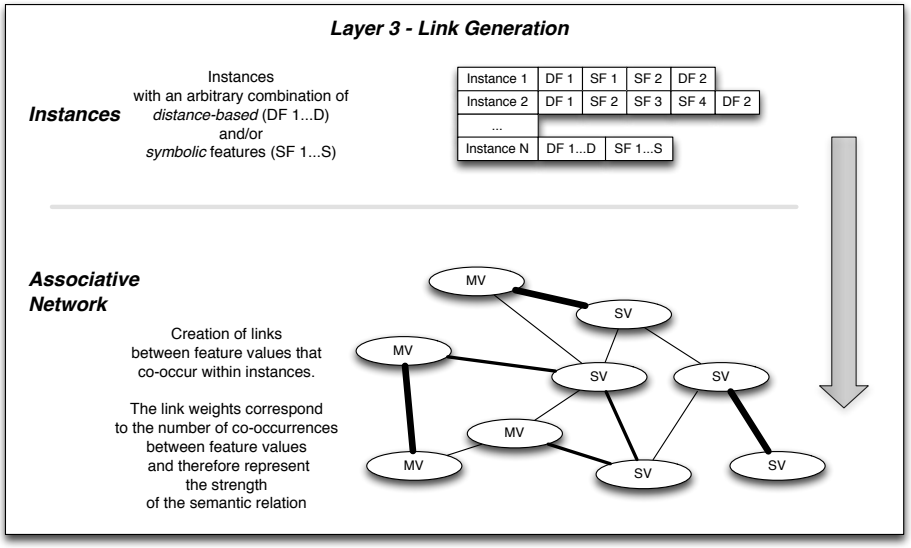

Figure 6.5: *Layer 3*: Generation of the associative network that represents the feature values and their relations.

In this layer, which is depicted in Figure 6.5, the relations between the feature values are extracted and links representing these relations are created within the associative network. Thereby, two feature values of arbitrary features are considered to be related when they co-occur within the same instance. The strength of the relation is defined by the number of co-occurrences within the complete instance set. The relations and their strength are modeled as weighted links between the network's nodes corresponding to the feature values.

Extracting Relations From Instances

For each instance, its corresponding feature values, their features and feature types are extracted. Each feature value is then mapped to the corresponding node within the network. Thereby, symbolic feature values are directly mapped to network nodes. For distance-based features the best matching RGNG prototype needs to be found, which can then be mapped to a network node. After applying the mapping operation, a list of all nodes corresponding to the feature values within the analyzed instance is gained. For each possible combination of two nodes within this list, a link is created between these nodes, or if it already exists the weight of this link is increased.

Normalization

After processing all instances, the weights of the links are equal to the absolute number of co-occurrences between feature values within the analyzed instances. Since these absolute values cannot be used for the SA process applied in *Layer 4*, a normalization technique must be applied. For most of the analysis processes a local max norm yields the best results: Thereby, the outgoing links of a given node are normed according to the maximum link weight of these links. This local norm emphasizes feature values that sparsely occur within the data set, which allows for a better analysis of their relations.

Fanout Values

By taking a closer look at the nodes and their weighted links, knowledge about the information carried by a node can be extracted. This knowledge can then be used to attenuate the influence of certain nodes when applying the spreading activation techniques in the next layer. The rationale behind the attenuation is to avoid noise that is introduced by nodes that are connected to a large number of other nodes within the network. The issues is highlighted by Example 16. For a detailed discussion on how fanout factors are calculated and how they influence the spreading activation results, the reader is referred to Section 7.3.5 of the next chapter.

Example XVI: Attenuating the influence of a redundant node

Given an associative network that models the information contained in instances, which describe countries with various features such as export commodities, unemployment rates, languages or continents, and assuming, the analysis is focused on details of European countries only: Then, the feature *continent* becomes redundant, because each country within the analyzed instances is on the same continent. Thus, all other nodes within the network representing the possible feature values are linked to the node representing the feature value *Europe*. When applying the spreading activation techniques in the next layer, this feature value would therefore cause the activation of every other node which would add noise to each generated pattern. Therefore, the influence of such nodes must be attenuated by employing so called fanout factors.

Layer 3 - Processing Steps

The processing steps executed in *Layer 3* are defined via the pseudo algorithm shown in Algorithm 7 and described as follows:

- *Input*:
  - The instance set (I) containing and arbitrary number of instances.

  - The network's meta-information structure *NET_I* now containing the feature information extracted in *Layer 1* and the distance-information extracted in *Layer 2*.
  - The initial associative network *NET* containing the nodes.

- *Output*:
  - The final associative network *NET* containing the nodes and the normalized weighted links.
  - The fanout values stored within the network's meta-information structure *NET_I*.

- *Lines 2 to 3*: For a given instance i_i all features F are extracted, and an empty array N reserved for the network nodes corresponding to the feature values within the instance i_i is created.

- *Lines 4 to 18*: The feature values FV of the extracted features F are mapped according to their feature type C to nodes within the associative network *NET*. The mapped nodes are stored within the node array N.

- *Lines 20 to 30*: For each possible combination of two nodes a_i and a_j within the array N, the strength of the weighted link w_{ij} between a_i and a_j is increased by 1.0. If the link does not exist, it is created.

- *Lines 31*: A normalization algorithm is applied to all links within the network *NET* in order enable the application of spreading activation techniques in the next layer.

- *Lines 32 to 33*: The fanout values *fanouts* are calculated for each node and stored within the network's meta-information structure *NET_I*.

Algorithm 7 *Layer 3* - Network links generation

Require:
$I = \{i_1, i_2, ..., i_n\}$ {set of instances}
NET {associative network with nodes generated in the previous layer}
NET_I {network meta-information structure}

1: **for all** i_i in I **do**
2: $N = \{\}$ {initialize node list, which will contain all nodes representing the feature values of the given instance}
3: F = extractFeatures(i_i) {extract all features from the given instance}
4: **for all** F_j in F **do**
5: C = getFeatureType(NET_I,F_j) {get feature type: symbolic or distance-based}
6: FV = extractFeatureValues(F_j) {extract all features values of the given feature contained in the instance}
7: **for all** FV_k in FV **do**
8: **if** {C = *"symbolic"*} **then**
9: a = getNode(NET,F_j,FV_k,*"symbolic"*) {retrieve node directly from network}
10: addNode(N,a) {store node in array}
11: **else if** {C = *"distance-based"*} **then**
12: *model* = getModel(NET_I,F_j) {get model for distance based feature}
13: pr = mapFeatureValue(model,FV_k) {get the best-matching prototype for the given feature value}
14: a = getNode(NET_I,F_j,pr,*"distance-based"*) {retrieve corresponding node from the network}
15: addNode(N,a) {store node in array}
16: **end if**
17: **end for**
18: **end for**
19: {loop over all network nodes contained in the array}
20: **for all** a_i in N, a_j in N **do**
21: **if** {a_i != a_j} **then**
22: w_{ij} = getWeightedLink(NET,a_i,a_j) {retrieve the link for the given nodes from the network}
23: **if** {$w_{ij} = NULL$} **then**
24: w_{ij} = createLink(NET,a_i,a_j) {if link does not exist, it is created}
25: $w_{ij} = 0.0$ {initialize new link with weight 0.0}
26: **end if**
27: $w_{ij} = w_{ij} + 1.0$ {increase the weight of the link by 1.0}
28: **end if**
29: **end for**
30: **end for**
31: applyNormalization(NET) {apply normalization algorithm to network links}
32: *fanouts* = calcFanoutValues(NET,NET_I) {calculate the fanout values for the network nodes}
33: storeFanoutValues(NET_I,*fanouts*) {store the fanout values in the network's meta-information structure}

6.1.4 Layer 4 - *Semantic Pattern* Generation

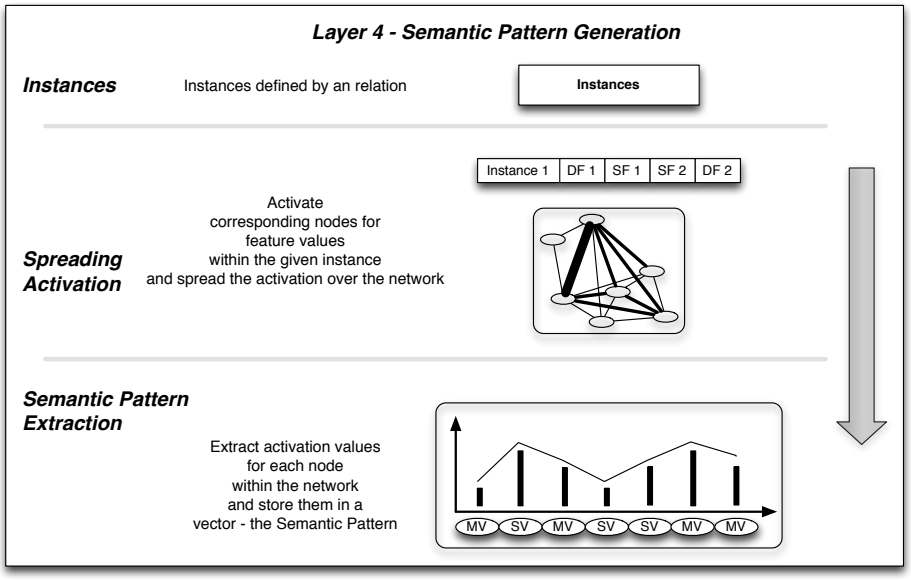

Figure 6.6: *Layer 4*: Generation of the *Semantic Patterns*.

In *Layer 4* the network representing the information and relations contained within the input instances is used to generate the actual *Semantic Patterns*, which form the basis for further analysis processes. Thereby, depending on the analysis single or multiple feature values or complete instances can be transformed into *Semantic Patterns*. Typically, for the initial analysis the patterns for all instances are generated and then used as a basis for further processing steps.

After selecting the input feature values, the corresponding nodes within the associative network need to be retrieved. Thereby, as in the previous layers, symbolic feature values are mapped directly and distance-based feature values are mapped according to the discretization model trained in *Layer 2*. The mapped nodes are then activated and their activation is spread via the application of spreading activation algorithms. Thereby, different parameters concerning the application of the spreading activation algorithm need to be tuned according to the desired analysis. A detailed explanation of these parameters and their influence will be given in the next chapter.

After spreading the activation of all activated nodes, the activation values of all associative network nodes are extracted and stored in a vector. This vector is called *Semantic Pattern* and forms the basis for all further analysis processes.

The structure of a *Semantic Pattern* is highlighted in Figure6.7.

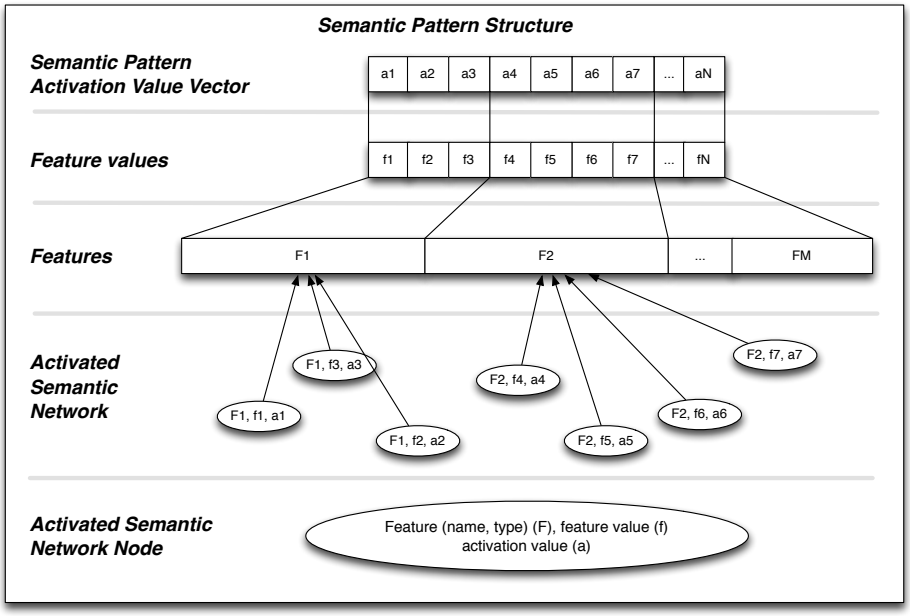

Figure 6.7: Structure of a *Semantic Pattern*. After activating the nodes in the associative network and spreading their activation, the activation values of the network nodes are extracted and stored in a vector, which represents the *Semantic Pattern*.

Layer 4 - Processing Steps

The processing steps executed in *Layer 4* are defined via the pseudo algorithm shown in Algorithm 8 and described as follows:

- *Input*:
  - The instance set (I) containing and arbitrary number of instances.
  - The network's meta-information structure *NET_I*.
  - The final associative network NET containing the nodes and the normalized weighted links.
- *Output*: The generated *Semantic Patterns* are stored within the set P.
- *Line 1*: Initialize set P which will contain the transformed *Semantic Patterns*.
- *Line 2 to 7*: All features F, their feature type C and their feature values FV are extracted from the given instance i_i.
- *Line 8 to 19*: Depending on the feature type C the corresponding nodes are extracted from the network NET and stored in the set A.

Algorithm 8 *Layer 4* - Pattern Generation

Require:
NET {trained associative network}
NET_I {network meta-information structure}
$I = \{i_1, i_2, ..., i_n\}$ {set of instances} P {transformed Semantic Patterns representing the input instances}

1: $P = \{\}$ {initialize set that will contain the transformed patterns}
2: **for all** i_i in I **do**
3: F = extractFeatures(i_i) {extract features from the given instance}
4: $A = \{\}$ {initialize node list, which will contain all nodes representing the feature values of the given instance}
5: **for all** F_j in F **do**
6: FV = extractFeatureValues(F_i) {extract feature values from the instance for the given feature}
7: C = getFeatureType(NET_I,F_j) {get feature type: symbolic or distance-based}
8: **for all** FV_k in FV **do**
9: **if** {C = *"symbolic"*} **then**
10: a = getNode(NET,F_j,FV_k, *"symbolic"*) {retrieve corresponding node from the network}
11: addNode(A,a) {add node to set}
12: **else if** {C = *"distance-based"*} **then**
13: *model* = getModel(NET_I,F_j) {get model for distance based feature}
14: *pr* = mapFeatureValue(model,FV_k) {get the best-matching prototype for the given feature value}
15: a = getNode(NET,F_j,*pr*, *"distance-based"*) {retrieve corresponding node from the network}
16: addNode(A,a) {add node to set}
17: **end if**
18: **end for**
19: **end for**
20: NET_A = spreadActivation(NET,NET_I,A) {activate selected nodes and apply spreading activation techniques}
21: p = extractSemanticPattern(NET_A) {extract all activation values from activated network and store them in a vector – the *Semantic Pattern*}
22: addPattern(P,p) {add generated pattern to set}
23: **end for**

- *Line 20*: The nodes contained in the set A are activated within the associative network and their activation is spread to the neighboring nodes by applying spreading activation techniques. The resulting activation state of the network is stored in NET_A.
- *Line 21 to 22*: The activation values of all nodes of the activated network NET_A are extracted and stored in a vector that is called the *Semantic Pattern* p. This pattern is added to the result pattern set P, which forms the basis for further analysis processes.

6.1.5 Layer 5 - Analysis

The *Semantic Patterns* form the basis for arbitrary analysis processes, which will be described in the next chapter.

6.2 Chapter Conclusions

The *Semantic Pattern Transformation* is organized in five processing layers that transform the raw feature vectors into *Semantic Patterns*. Thereby, associative networks, unsupervised machine learning and spreading activation are the core building blocks of the whole transformation process.

7 Semantic Pattern Analysis

The *Semantic Pattern Transformation* transforms arbitrary feature vectors into *Semantic Patterns*. Thereby, the response of the underlying associative network to an input-stimulus – consisting of one or multiple feature values – is gained by activating the corresponding network nodes, and spreading their activation values to the neighboring nodes. The activation state of the complete network is then extracted and stored in a vector – the *Semantic Pattern*. These patterns form the basis for all further knowledge extraction procedures, which include simple filtering and sorting algorithms for pattern interpretation, the application of similarity measures for the implementation of semantic search algorithms, or highly sophisticated machine learning algorithms used for unsupervised or supervised learning scenarios.

This chapter focuses on the interpretation of the *Semantic Patterns*, then goes into various details of the transformation process that were omitted in the last chapter, and finally presents various knowledge extraction procedures. The most important of these processes will then be evaluated in the next chapter.

7.1 Interpretation

The underlying vector of a *Semantic Pattern* represents the activation state of an activated associative network. Therefore, information about the network and its response to the input-stimuli can be extracted directly from this vector representation without the need to query the network. The remainder of this section will focus on the interpretation of *Semantic Patterns* by discussing basic operations for the extraction and processing of the represented semantic information. The given examples are taken from the two demonstrations data sets described

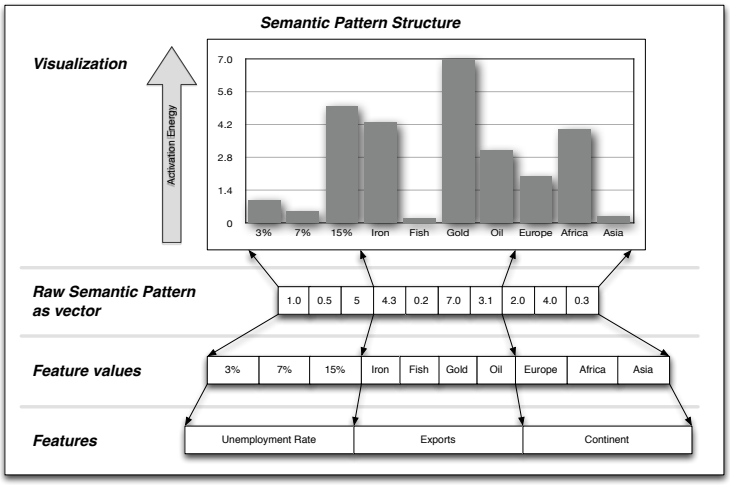

Figure 7.1: An example for a *Semantic Pattern* representing a country. Different sections of the pattern correspond to different features, and each feature is represented by one or multiple feature values.

in the appendix of this thesis, where the first contains instances representing the world's countries, and the second one is a collection of Tweets that were published on the social network Twitter during the Egyptian revolution in early 2011.

7.1.1 Interpreting the Activation Values

The basic structure of a *Semantic Pattern* is depicted in Figure 7.1, where the value-centric feature vector of an African country has been transformed into a *Semantic Pattern*. For visualization purposes, only a small number of feature values contained in the features *Unemployment rate*, *Exports* and *Continent* are shown. Thereby, the raw *Semantic Pattern* depicted in the middle of the figure contains the activation values that where extracted from the associative network after applying the spreading activation algorithm. Each value corresponds to the activation value of a network node, which represents a certain feature value, and each feature value is associated with a certain feature. The *Semantic Pattern* can easily be visualized as shown in the upper part of the figure.

However, in contrast to the simple example in Figure 7.1, most *Semantic Patterns* are represented by high-dimensional vectors. Thus, for the interpretation of such patterns the focus is typically placed on certain sub-parts (e.g., features), which are extracted by applying simple filters and other operations.

An example for such an operation is the sorting of the activation values and the subsequent extraction of the most active ones. By visualizing an adequate number of activation values, a quick overview about the most important properties of the analyzed pattern can be gained. This is highlighted by two examples

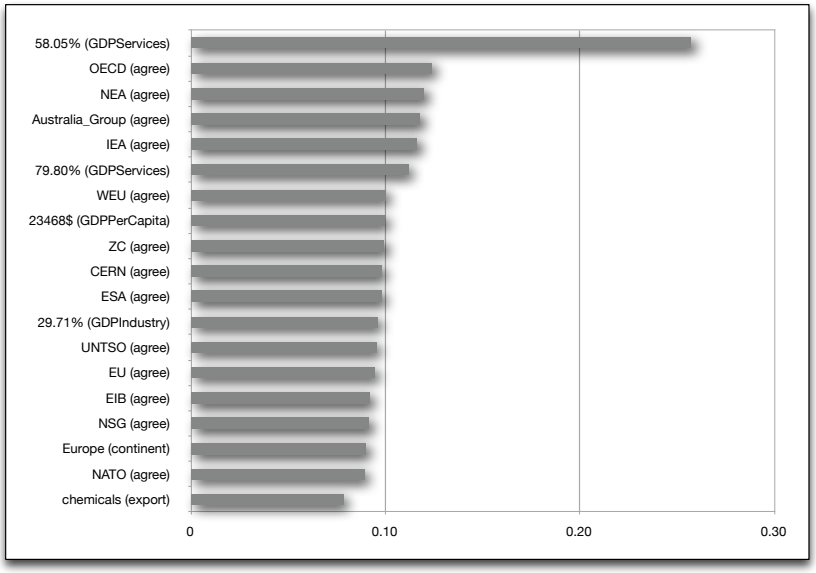

Figure 7.2: *Semantic Pattern* for the feature value *70%* of the feature *GDP-Service*. The 19 most active and thus most semantically related feature values are visualized.

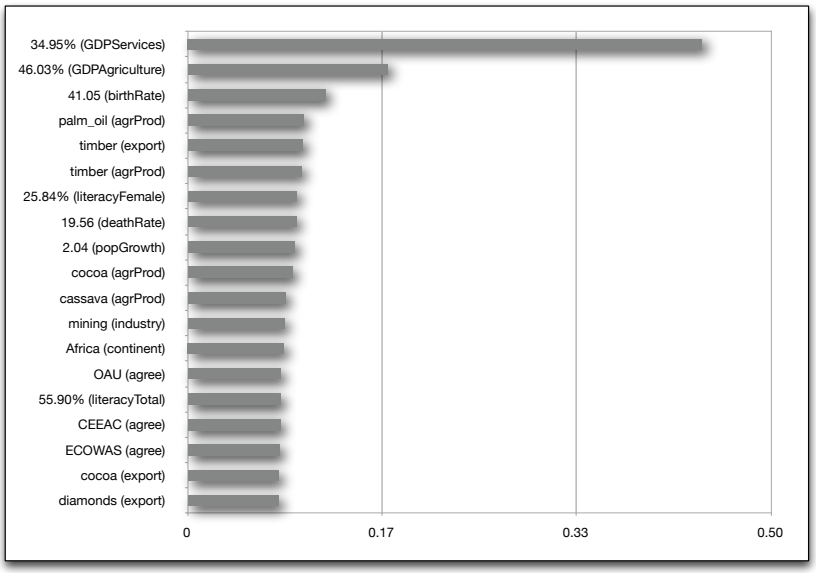

Figure 7.3: *Semantic Pattern* for the feature value *20%* of the feature *GDP-Service*. The 19 most active and thus most semantically related feature values are visualized.

shown in Figure 7.2 and Figure 7.3. In these examples the semantic relations of the feature *GDP-Service* are extracted. This feature describes the contribution of the service sector to the Gross Domestic Product (GDP) in relation to the agricultural and industrial sector. Thereby, the first pattern represents the semantic fingerprint of a highly developed country with a large service sector at *70%*, while the second example shows the pattern of a country with a smaller service sector at *20%*. In both cases the 19 most active feature values of the corresponding *Semantic Patterns* have been extracted and sorted. By looking at these values, a quick understanding on the semantic relations within the generated pattern can be gained. For the two examples it is easy to observe that the feature values *70%* and *20%* occur in a completely different semantic context.

Since the *GDP-Service* feature is a distance-based feature, the values *70%* and *20%* cannot be directly mapped to nodes within the associative network. Thus, a discretization operation needs to be applied. Here, the best matching node within the network for the value *70%* is a node representing *67.3%* and for *20%* there is one with the value *27.51%*. These best matching nodes would be the most activate ones, but are not shown in the bar charts due to scaling issues. However, the represented values also have a small distance to other nodes of the *GDP-Service* feature. These nodes are co-activated as seen by the most active feature values *58.05%* in the first example and *34.95%* in the second example. The reason for this co-activation is the application of the pre-spreading technique described in Section 7.3.3, which is applied to nodes representing distance-based feature values.

By looking at the bar charts one can quickly observe that a large service sector (*70%*) is closely related to international agreements such as the European Union (*EU*), the European Organization for Nuclear Research (*CERN*), the European Space Agency (*ESA*), a rather high *GDP per capita* (*23468$*) and *chemicals* as export good. In contrast, a small service sector (*20%*) is strongly associated with a large agricultural sector (*46.03%*), raw materials as export goods such as *palm oil*, *timber*, *cocoa*, or *diamonds*, a rather high death rate (*19.56*), the continent *Africa* and a low literacy rate (*55.9%*).

When the extraction of information should be limited to certain features, then simple filter operations can be applied. This is shown by the two examples in Figure 7.4 and Figure 7.5. Thereby, the first one is the result of activating the feature value *Europe* of the feature *Continent*, while the second one uses the feature value *Africa* of the same feature. The resulting patterns reveal the semantic relations of the feature values *Europe* and *Africa*. In order to focus on the feature *Exports* only, the appropriate feature values are extracted and sorted according to their activation values. By looking at the bar charts, one can easily observe the differences between *Europe* and *Africa*. While in *Europe* the most relevant exports are related to industrial products, such as *machinery and equipment*, *chemicals* or *manufactured goods*, the most active values in *Africa* are raw materials such as *cotton*, *coffee*, *diamonds* or *timber*. Another representation is shown in Table 7.1, where the three most active feature values for each feature of the two *Semantic Patterns* are extracted.

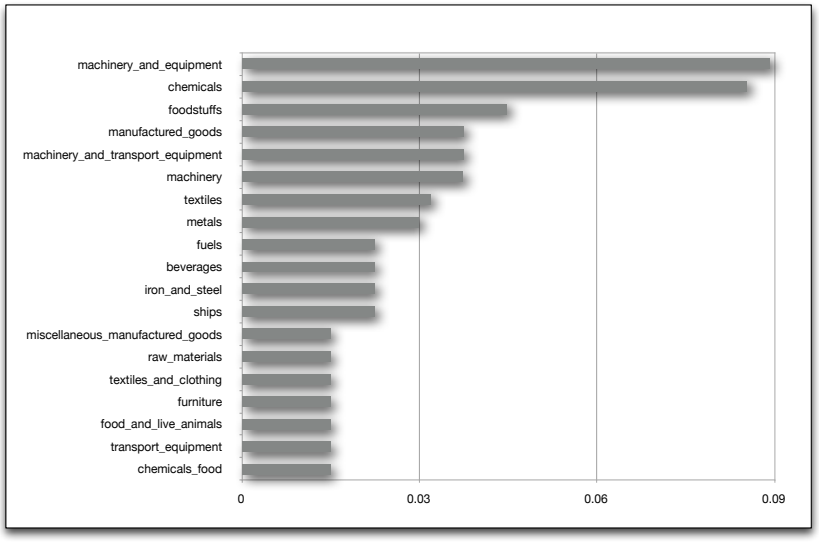

Figure 7.4: *Semantic Pattern* for the feature value *Europe* of the feature *Continent*. Only the activation values related to the feature *Exports* have been extracted.

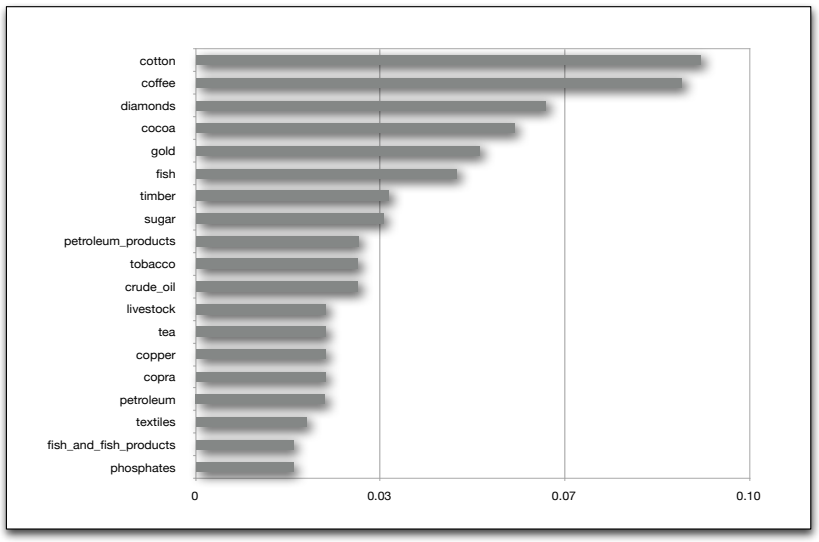

Figure 7.5: *Semantic Pattern* for the feature value *Africa* of the feature *Continent*. Only the activation values related to the feature *Exports* have been extracted.

Activated Nodes	GDP-Services 70%	GDP-Services 20%
Feature	Feature Value	
GDP-Agriculture	11.93%	46.03%
	4.37%	36.41%
	19.62%	19.62%
GDPPerCapita	$23,468	$1,587
	$16,146	$5,666
	$7,625	$3,283
MilitaryGDP	2.02%	2.02%
	1.17%	5.08%
	3.07%	7.61%
PopGrowth	0.32	2.04
	1.04	2.97
	1.57	1.04
Unemployment	3.63%	26.75%
	10.60%	7.35%
	7.35%	42.76%
AgrProd	sugar beets	palm oil
	dairy products	timber
	barley	cocoa
Export	chemicals	timber
	machinery and equipment	cocoa
	fish	diamonds
Import	fuels	petroleum products
	transport equipment	grain
	machinery	fuels and lubricants
Industry	machinery	mining
	clothing	metallurgy
	shipbuilding	machine building

Table 7.1: This table shows a textual representation of the semantic relations for the values *70%* and *20%* of the feature *GDP-Services*.

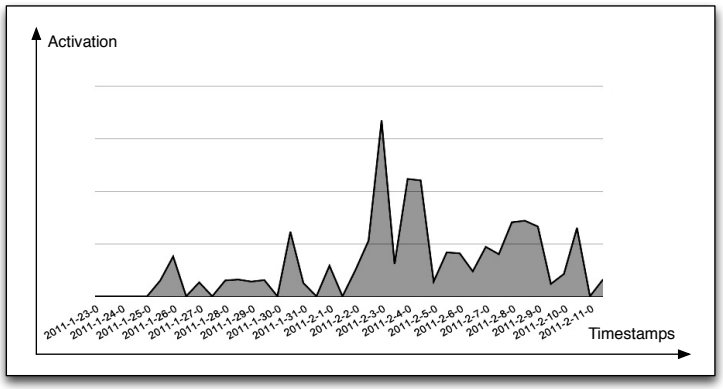

Figure 7.6: Semantic activity of the feature value *Tahrir* over time.

In other scenarios, information can be extracted by sorting the activation values of a *Semantic Pattern* according to the contained feature values. This representation can be used when time-based feature values are included within the analyzed instances. In this case, when a *Semantic Pattern* is generated for an arbitrary concept, the activation values of the time-related nodes represent the semantic development of this concept over time. This is shown for a *Semantic Pattern* that was generated for a term contained in the second demonstration data set, which covers the Egyptian revolution in early 2011. During this revolution the most important demonstrations against Hosni Mubarak occurred on the *Tahrir* place in Cairo. The semantic activity over time contained within the pattern for *Tahrir* is visualized in Figure 7.6.

The presented examples highlight that *Semantic Patterns* can easily be interpreted by applying simple filters and sorting operations. The application of such and other operations to the patterns does not depend on the nature of the analyzed data, the defined knowledge discovery process, nor the applied analysis method. This is a huge advantage for the definition and application of arbitrary knowledge discovery processes, because it removes the need for the implementation of domain-specific interpretation methods.

7.2 Pattern Arithmetic

The vector representation of the *Semantic Patterns* enables the application of a wide range of simple vector-based operations for the transformation and processing of the patterns in order to extract additional information. In this section the most common operations, such as calculating the similarity of two patterns, or combining different patterns, and their main purposes will be discussed.

7.2.1 Similarity

The similarity of two *Semantic Patterns* plays an important role for the application of further analysis processes, such as supervised or unsupervised learning, or the deployment of semantic search algorithms. The differences between two patterns, as depicted in Figure 7.7, represent the different active regions of the underlying associative network. This can also be expressed via a distance measure applied to two vectors, where the calculated distance reflects the semantic similarity. Thereby, two patterns are considered as similar when their corresponding input-stimuli are semantically related and, therefore, cause the activation of similar regions within the network. On the contrary, when two patterns activate distinct regions, then their input-stimuli have no semantic relation at all. In order to model this behavior, the Cosine similarity is the best choice, because it is based on the angle between two *Semantic Patterns*. This means, when two patterns activate distinct regions within the network, they are orthogonal which is reflected by the maximum distance at $\frac{\pi}{2}$. While the Cosine similarity is an obvious choice for determining the similarity of *Semantic Patterns*, the Euclidean distance can also be used, which is an important aspect

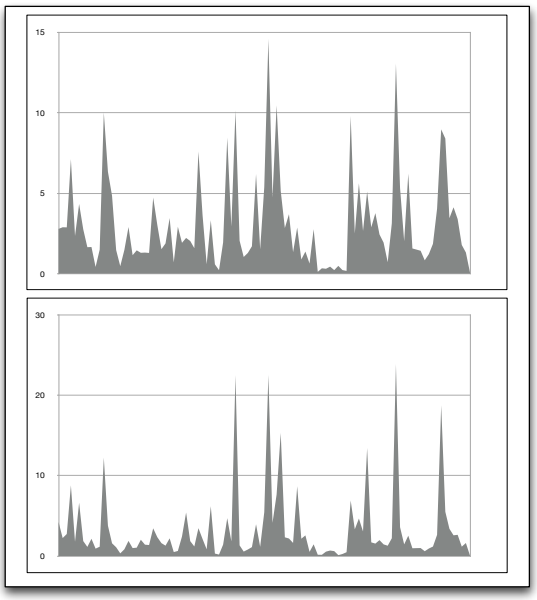

Figure 7.7: These two example patterns represent different regions that are activated within the associative network. The differences between the patterns can be used to calculate their similarity.

when applying standard machine learning algorithms for unsupervised or supervised learning. The calculation of the similarity also plays an important role when applying semantic search algorithms, which will be explained in Section 7.4.3. The differences between the two distance measures will then be revisited in the next chapter, when the semantic search algorithm will be evaluated on real data.

7.2.2 Adding and Subtracting *Semantic Patterns*

The application of simple vector-based addition and subtraction are basic operations that can be applied to *Semantic Patterns* based on the same underlying associative network. While the first operation allows the combination of the semantic concepts represented by different patterns, the second operation is applied when the main differences between two concepts need to be identified. The topic of combining two *Semantic Patterns* is also strongly related to the combination function employed in the spreading activation process, which will be discussed in Section 7.3.4.

The subtraction of two patterns can be used to gain a quick overview of the differences in their semantic activations. This is highlighted by the two examples given in Figure 7.8 and Figure 7.9. In the first case, two *Semantic Patterns* have been generated by activating the feature values *Europe* and *Africa* of the feature

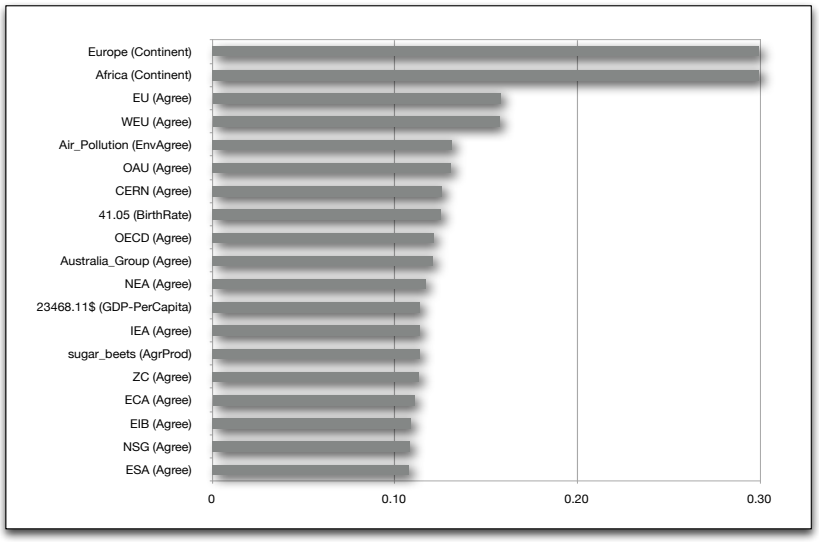

Figure 7.8: Difference of the *Semantic Patterns* for *Europe* and *Africa*, sorted according to the absolute activation values.

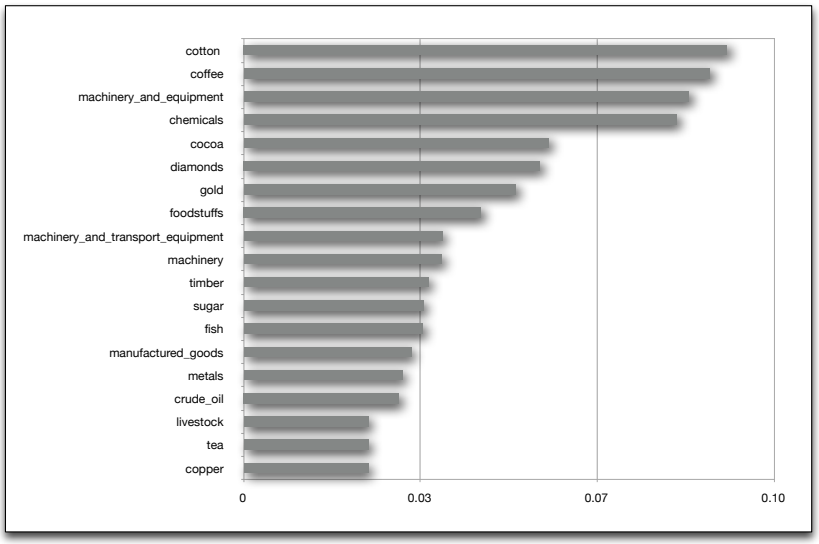

Figure 7.9: Differences of the export commodities between the *Semantic Patterns* for *Europe* and *Africa*, sorted according to the absolute activation values.

Continent. The generated patterns have been subtracted and the absolute values of their elements have been calculated. By sorting the activation values of the gained vector, the main deviations can be identified. As easily observed in the figure, the feature values for *Africa* and *Europe* dominate these differences and are then followed by differences related to the agreements that are signed by European and African countries. In the second case, the same patterns have been subtracted. Here, only the activation values of the feature values related to the feature *Exports* have been extracted.

7.2.3 Mean Value and Variance

By calculating the mean pattern of multiple input patterns, one is able to gain a quick overview of the typical feature values contained in the input patters. An example for the application of this method is the analysis of pattern clusters as found by unsupervised clustering algorithms.

In a similar way, the variance of multiple *Semantic Patterns* shows how the feature values vary within an analyzed pattern group. This is highlighted by an example that analyzes the countries of the Eurozone: Here, the feature *currency* and the corresponding feature value *Euro* will have a much lower variance than the feature *language* and the corresponding feature values.

7.2.4 Activation Energy and Other Operations

By summing up the activation values of a pattern, its total activation energy can be calculated. This energy is a measure on how strong the network response is to a given input-stimulus. This information can be used for detecting anomalies.

Also, many other simple or more sophisticated operations can be applied to the patterns for extracting further information. An example would be the application of methods like the Principal Component Analysis (PCA), which is used to reduce the pattern dimensionality by removing non-relevant features.

7.3 Spreading Activation Techniques

The five principle layers of the *Semantic Pattern* transformation have been described in the previous chapter. In this section, the focus is placed on further details regarding the spreading activation process. Although the principle idea behind this process is fairly simple, several issues need to be considered during its application. These include the appropriate handling of nodes representing distance-based features, the parameters regarding the spreading activation process itself, and the application of fanout procedures to enable constrained-spreading. In general, there is no need to fine-tune these aspects when applying the *Semantic Pattern Transformation* to an arbitrary data set. However, it is important to understand the influence of the different parameters and methods in order to find the best generic strategies used within the default configuration.

7.3.1 Types of *Semantic Patterns*

Before further discussions on the details of the spreading activation process are presented, the different *Semantic Patterns* types need to be discussed. Given an arbitrary knowledge discovery process that involves the *Semantic Pattern Transformation*: Then, according to Section 3.2 in *Chapter 3 – Knowledge Discovery and Machine Learning*, instances described by features with different feature values can be extracted by applying some kind of relation to the data set. Typically, these instances are then transformed into *Semantic Patterns*, which are subject to further analysis processes. However, the transformation process is not limited to complete instances. In fact, in can be applied to any combination of the feature values contained within the data set. The thereby gained pattern types are summarized in the following list:

- **Instances**: An instance refers to the combination of multiple feature values, but is special in the sense that this combination is defined via the relation applied to the raw data contained within the analyzed data set. For most of the standard applications of the *Semantic Pattern Transformation*, these instances are transformed into *Semantic Patterns*, which are then subject to further interpretation and analysis processes.

- **Single feature value**: *Semantic Patterns* for single feature values are generated in order to gain more information about the semantic relations contained within the associative network. This procedure was already demonstrated by multiple examples in Section 7.1.1 of this chapter.

- **Multiple feature values**: *Semantic Patterns* that are generated for multiple feature values reveal information about the common semantic information shared by the input feature values. Here, the application of adequate nonlinear combination functions is required for the spreading activation process. The rationale behind these combination functions is explained in Section 7.3.4 of this chapter. The generation of *Semantic Patterns* for single or multiple feature values also plays an important role within semantic search algorithms. Here, the generated patterns are compared to an existing data set of *Semantic Patterns* and the most similar ones are retrieved. Such a data set could be based on the instances that were used during the training of the associative network or any other *Semantic Patterns* based on single or multiple feature values that were generated based on the trained network.

7.3.2 Generating a *Semantic Pattern*

Assuming that Layers 1 to 3 of the *Semantic Pattern Transformation* have already been applied to an arbitrary data set, then the *Semantic Patterns* can be generated by applying spreading activation on the trained associative network. While the general process was already explained in the previous chapter, here the focus will be placed on the details of the spreading activation algorithm.

The detailed procedures are implemented in Algorithms 9 and 10, which are described as follows:

Based on the trained associative network, *Semantic Patterns* can be generated for either the complete instances, or any other combination of feature values, or even a single feature value.

Algorithm 9 Generating a *Semantic Pattern*

Require:
NET {trained associative network}
NET_I {network meta-information structure}
$F = \{F_1, F_2, ..., F_n\}$ {set of input feature values}
1: A_d = getDistanceBasedNodes(F,*NET*) {map distance-based feature values to nodes A_d}
2: A_s = getSymbolicNodes(F,*NET*) {map symbolic feature values to nodes A_s}
3: A_p = applyPreSpreading(A_d,*NET*,*NET_I*) {apply pre-spreading to distance-based nodes A_d}
4: $A = \{A_s, A_p\}$ {combine symbolic and pre-activated distance-based nodes}
5: P=applySpreadingActivation(A,*NET*,*NET_I*) {apply Algorithm 10 and extract the *Semantic Pattern* P}

- *Input*:
  - The trained associative network *NET*.
  - The network's meta-information structure *NET_I*.
  - A set F of symbolic and/or distance-based features and their corresponding feature values.
- *Output*: The generated *Semantic Pattern* P.
- *Line 1-2*: Depending on the feature type, the symbolic feature values are mapped to the symbolic or distance-based node sets A_d and A_s. While the symbolic feature values are directly mapped to the network nodes, a discretization model is required for the distance-based feature values.
- *Line 3*: For the distance-based feature values, a pre-spreading procedure is required. This will be explained in detail in Section 7.3.3.
- *Line 4-5*: The symbolic and distance-based node sets are combined, spreading activation (Algorithm 10) is applied, and finally the *Semantic Pattern* P is extracted.

After extracting the node set A by mapping the feature values F to the network nodes, the actual spreading activation algorithm is applied in Algorithm 10, which is described as follows:

- *Input*:

Algorithm 10 Basic spreading activation algorithm

Require:
NET {trained associative network}
NET_I {network meta-information structure}
$A = \{a_1, a_2, ..., a_n\}$ {set of nodes to be activated}
IA {initial activation}
D {decay factor}
fanoutBefore: true or *false* {fanout before spreading}
fanoutDuring: true or *false* {fanout during spreading}
fanoutAfter: true or *false* {fanout after spreading}

1: **for all** a_i in A **do**
2: **if** (*fanoutBefore*) **then**
3: FO = getFanoutValue(*NET_I*,a_i) {get the fanout value FO for node a_i}
4: setActivation(a_i,IA∗FO) {set initial activation IA multiplied by FO}
5: **else**
6: setActivation(a_i,IA) {set initial activation values for input nodes}
7: **end if**
8: **end for**
9: **for all** a_i in A **do**
10: L = getEmanatingLinks(a_i) {get emanating weighted links w_{ij} from a_i}
11: S = getActivation(a_i) {get activation value of source node a_i}
12: **for all** w_{ij} in L **do**
13: a_j = getTargetNode(w_{ij}) {get target node a_j of weighted link w_{ij}}
14: A = getActivation(a_j) {get activation value of target node}
15: $T = S * w_{ij} * D$ {calculate target activation value}
16: **if** (*fanoutDuring*) **then**
17: $T = T$∗getFanoutValue(*NET_I*,a_i) {adapt activation by fanout value}
18: **end if**
19: A = *combineActivation*(A, T) {apply combination function to A and T}
20: setActivation(a_j,A) {set new activation value A of target node a_j}
21: **end for**
22: **end for**
23: **if** (*fanoutAfter*) **then**
24: **for all** a_i in *NET* **do**
25: FO = getFanoutValue(*NET_I*,a_i) {get fanout value for node a_i}
26: setActivation(a_i,getActivation(a_i)∗FO) {set fanout activation of node a_i}
27: **end for**
28: **end if**
29: P = extractPattern(*NET*)

- The trained associative network *NET*.
- The network's meta-information structure *NET_I*.
- The set $A = \{a_1, a_2, ..., a_n\}$ of nodes to be activated.
- The initial activation value *IA*, which is typically set to 1.0.
- The decay factor *D*, which limits the amount of energy spread to neighboring nodes.

  - The parameters *fanoutBefore*, *fanoutDuring* and *fanoutAfter*, which describe when and if fanout procedures are applied.

- *Output*: The result is the generated *Semantic Pattern* P.
- *Line 1 to 8*: Set the initial activation value IA for the nodes contained in the set A. If the parameter *fanoutBefore* is activated, the value IA is multiplied with the corresponding fanout value FO.
- *Line 9 to 22*: The spreading activation algorithm is applied for each activated node a_i in A.
- *Line 10*: The links emanating from node a_i are extracted and stored in L.
- *Line 11*: The source activation value S of node a_i is extracted.
- *Line 12 to 21*: For each emanating weighted link w_{ij}, the activation is spread to the respective target node.
- *Line 13 to 14*: The target node a_j of link w_{ij} is extracted, and its activation value A is retrieved.
- *Line 15*: The core spreading activation technique is applied here: The target activation value T is calculated by multiplying the source activation value S with the decay factor D and the weight of the link w_{ij}.
- *Line 16 to 18*: If *fanoutDuring* was set to *true*, then the target activation value T is multiplied with the fanout value of the source node a_i.
- *Line 19 to 20*: The calculated target activation value T, which corresponds to the energy spread from the source node a_i, is combined with the activation value A of the target node a_j. The combination function could be based on a simple addition or more advanced schemes[1]. Finally, the calculated activation value A is set for node a_j.
- *Line 23 to 28*: If *fanoutAfter* was set to *true* then the activation values of all networks nodes are multiplied with their corresponding fanout values.
- *Line 29:* Finally, the activation values of all nodes are extracted and stored as *Semantic Pattern* in the vector P.

The two algorithms utilize several new concepts that have not been explained before. These include:

- **Pre-spreading for distance based-features**: A pre-spreading process for distance-based feature values is applied in order to cope with problems related to very detailed discretization models and to loosing the distance information contained in the original feature values.

[1]The combination function could also be based on a strategy that requires all the received activation values before they can be combined. In this case, the activation values would be stored here and then later combined after the spreading activation process has been completed.

- **Fanout before, during and after spreading**: In order to remove the influence of highly connected nodes, fanout procedures are applied. Thereby, the fanout values need to be calculated for each node based on the network structure.
- **Combining the activation**: In order to combine different activation values within a target node, different combination functions can be used – including simple addition and advanced schemes.

These concepts will be described in the following sections and the influence of their different parameters will be evaluated in the next chapter.

7.3.3 Spreading Activation for Distance-Based Features

The *Semantic Pattern Transformation* represents arbitrary feature values as nodes within an associative network and their semantic relations via weighted links. While symbolic feature values can be directly mapped to network nodes, some kind of discretization operation is required for representing the distance-based values within the associative network. Although the discretization operation solves the problem of mapping feature values to nodes, it comes along with two core problems: **First**, the discretization operation must find the right model complexity, which corresponds to the number of nodes used to represent the distance-based feature values. **Second**, only the semantic relations between feature values are maintained, but not the distance-based information that is contained in the feature values.

Problem 1: Discretization Model Complexity

Obviously, an adequate discretization model complexity is required for accurately modeling the distance-based values contained in the analyzed feature. The problem is strongly related to finding an adequate model complexity within unsupervised learning, which was discussed in Section 5.6 of *Chapter 5 – Techniques*. In fact, the *Semantic Pattern Transformation* employs the unsupervised RGNG algorithm for the discretization of distance-based features. This algorithm uses the MDL criterion in order to find an adequate model.

However, depending on the analysis, more detailed models might be required. In this cases, the MDL override procedure discussed in Section 5.6 of *Chapter 5 – Techniques* can be applied to the discretization models, in order to gain a model with the desired level of detail. Unfortunately, while the increased model complexity helps to highlight fine details within the analyzed data, it also comes along with a significant problem, which is highlighted in Figure 7.10. Assuming, the *Semantic Pattern Tranformation* is applied to instances, which contain a distance-based feature *F1* and a categorical feature *F2*, and further assuming, feature *F1* consists of several values that have a small distance to each other: Then, the associative network is built according to the feature value co-occurrences extracted from the instances. While the feature values C and D of *F2* can be directly mapped to the network nodes, a discretization operation

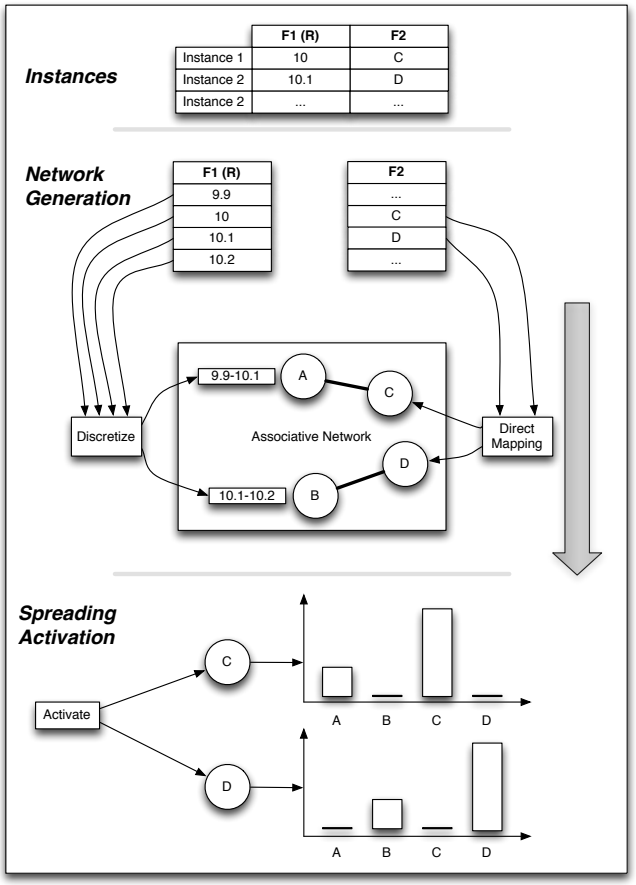

Figure 7.10: Problem 1: Discretization model complexity.

must be applied to the values of *F1*. Assuming, that the discretization algorithm finds a detailed model that maps values from 9.9 to 10.0 to node A and values from 10.0 to 10.1 to another node B, then the process yields the associative network depicted in the middle of the figure. The nodes A and C, and B and D are linked according to the co-occurrences of their corresponding feature values within the analyzed instances. Now, two *Semantic Patterns*, which are depicted in the lower part of the figure, are generated for the feature values represented by the nodes C and D. In the first case, the energy is spread from C to A and in the second case from D to B. Unfortunately, the two gained patterns are not related at all, which is in stark contrast to the closeness of the feature values used for their generation.

As a consequence, while increasing the model complexity enables the extraction of more details, it might also have a negative impact when comparing closely related distance-based feature values.

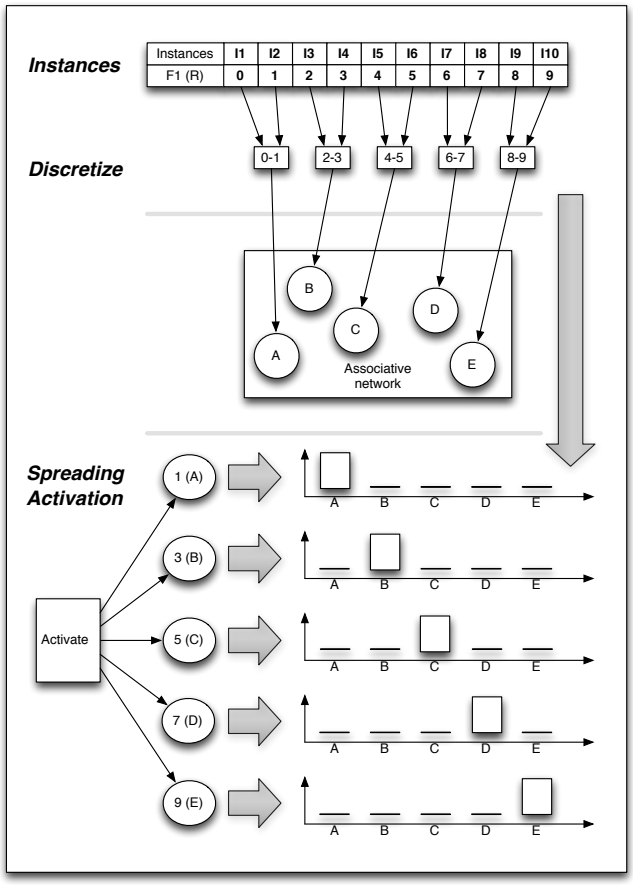

Figure 7.11: Problem 2: Loosing the distance information of distance-based features.

Problem 2: Loosing the Distance Information of Distance-Based Features

When distance-based feature values are mapped to nodes within the associative network via a discretization operation, the distance information of the original feature values is lost. An example of a trivial setup[2] is depicted in Figure 7.11, where instances are described by only one distance-based feature *F1*, which contains integer values from 0 to 9. Assuming, that the feature values are mapped to network nodes via a discretization operation that generates 5 nodes, then the *Semantic Patterns* depicted in the lower part of the figure are gained by applying the *Semantic Pattern Transformation*. In this case the *Semantic Pattern* representation is equal to a simple binary representation, where the presence of

[2]In oder to emphasize the problems, the instances used for this example only contain a single feature value. This means that no semantic relations in terms of co-occurrences can be derived when analyzing the instances.

a feature value is indicated by the value 1.0 and its absence by 0.0. Thereby, two issues need to be considered: **First**, the problem, which was described in Figure 7.10 re-emerges due to the non-similarity of patterns even when they represent closely related feature values. **Second**, the distance-information represented by the original feature values is lost. In fact, within the *Semantic Patterns* the distance-based values behave like symbolic values: In the example it is not possible to say whether the pattern for value 3 is closer to the pattern representing value 5, or the pattern representing the value 7. As a consequence, the correct order of the distance-based feature values cannot be derived for the *Semantic Patterns* as depicted in Example 1 of Figure 7.14. In summary, the advantage of modeling the semantic relations between feature values comes along with the problem of loosing the distance information.

Solution: Pre-spreading for Distance-Based features

Both problems can be solved by applying a pre-spreading strategy for distance-based features. This procedure is applied to nodes representing such feature values prior to the actual spreading activation process. The basic idea here is to co-activate other distance-based nodes of the same feature that represent closely related feature values.

Before this pre-spreading procedure can be applied, the required distance information must be extracted within the discretization procedure that is applied within *Layer 2* of the *Semantic Pattern Transformation* and stored for the later application of spreading activation algorithms. The extraction of the distance information in *Layer 2* is implemented according to Algorithm 11 and described as follows:

Algorithm 11 Extracting information for pre-spreading in *Layer 2*

Require:
NET_I {network meta-information structure}
C {MDL complexity override factor}
1: **for all** distance-based features F_j in F **do**
2: FV = extractFeatureValues(F_j) {extract all features values FV of the given feature F_j}
3: D = trainDiscretizationModel(C,FV) {generate the discretization model D}
4: DM = calculateDistanceMatrix(D) {calculation of the distance between each node of the model}
5: DM = maxNorm(DM) {norm the matrix with the maximum distance}
6: storeDistanceInformation(*NET_I*,F_j,DM) {store the distance information in the network's meta-information structure}
7: **end for**

- *Input*:
  - The network's meta-information structure *NET_I*.

  - The MDL override factor C that influences that complexity of the trained discretization model.

- *Output*: The information required by the pre-spreading process is stored within the network's meta-information structure *NET_I*.

- *Line 1 to 7*: For each distance-based feature F_j within the analyzed instances, the following procedure is applied:

- *Line 2 to 3*: The feature values FV of the feature F_j are extracted and used for training the discretization model D. The complexity of this model is influenced by the MDL override factor C. In this thesis, the unsupervised clustering algorithm RGNG is employed for the generation of the discretization models.

- *Line 4 to 5*: The distance values for each discretization prototype to each other prototype are calculated and stored in a symmetric matrix DM. Then, the matrix is normed by the maximal distance contained in the matrix.

- *Line 6*: The matrix DM for feature F_j is stored within the network's meta-information structure *NET_I*.

Algorithm 12 Pre-spreading

Require:
NET_I {network's meta-information structure}
NET {associative network}
a_d {the distance-based node a_d}
σ_p {the parameter σ_p that models the influence on the related nodes}
1: DM = getDistanceMatrix(a_d,NET_I) {get distance matrix for given node}
2: D = getDistances(a_d,DM) {get distances to other nodes D for node a_d}
3: **for all** d_i in D **do**
4: $d_i = 1 - d_i$ {adaptation of the normed distance}
5: $A = e^{\frac{-(d_i^2-2)^2}{2\sigma_p^2}}$ {application of the Gaussian function dependent on parameter σ_p}
6: setActivation(NET,a_i,A) {set the gained activation value for node d_i}
7: **end for**

When applying the spreading activation process for generating the *Semantic Patterns*, the gained distance information is utilized for the distance-based feature values according to Algorithm 12, which is described as follows:

- *Input*:

  - The network's meta-information structure *NET_I*.
  - The associative network *NET*.

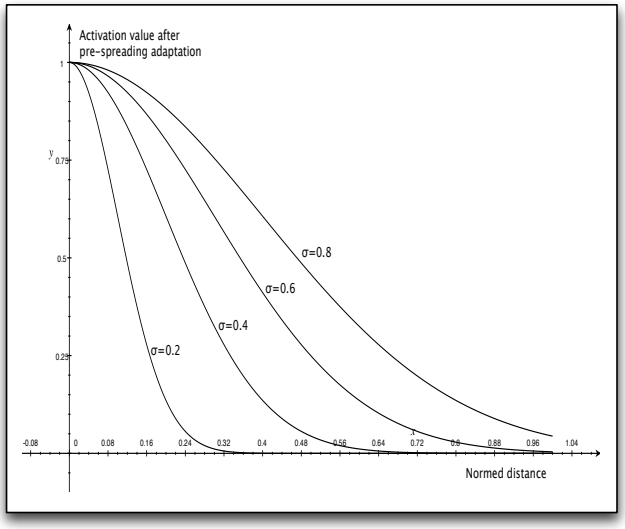

Figure 7.12: The relation between the normed distance to the activated node (x-axis), and the activation value calculated by applying the Gaussian function (y-axis). The influence of the factor σ_p is shown for the values $0.2, 0.4, 0.6$, and 0.8.

  - The distance-based node a_d for which the pre-spreading process needs to be applied.

- *Output*: The activation values of the nodes that are closely related to the input node a_d. These values are determined by applying the pre-spreading process.

- *Line 1*: Get the distance matrix DM for the node a from the network's information structure. This matrix has been determined during the generation of the discretization model as described in Algorithm 11.

- *Line 2*: The distances D from node a_d to the other nodes are extracted from the distance matrix DM.

- *Line 3-6*: For the distances d_i of all other nodes the activation values are calculated.

- *Line 4*: The distance value d_i is adapted by $1 - d_i$. Thus, the maximum distance is represented by the value 0, whereas the minimum distance is represented by the value 1.

- *Line 5*: The value d_i is transformed via the Gaussian function. The parameter σ_p influences the shape of this function, which is depicted in Figure 7.12.

- *Line 6*: The activation values calculated by applying the pre-spreading activation process are stored within the network and are used for the application of the spreading activation algorithm that analyzes the semantic relations to other nodes.

Algorithm 12 introduces the parameter σ_p, which is used to model the size of the neighborhood of the activated node, which still receives a majority of the initial energy. The influence of this parameter is shown in Figure 7.12, where the x-axis describes the normed distance from the activated discretization prototype to the other prototypes, and the y-axis describes the size of the activation values after applying the Gaussian function in *Line 5* of Algorithm 12. The influence of different values for σ_p is also shown.

Example XVII: Pre-spreading

The pre-spreading process is further explained by a simple example given in Figure 7.13. Here, the discretization operation is applied to a distance-based feature that contains values from 0 to 100 and yields 4 nodes representing the values *15*, *32*, *36* and *82*. These nodes are found by applying the unsupervised learning algorithm RGNG and are located at position were the density of the input values is high. Then, for these nodes, a distance matrix containing all distances is calculated and normed according to the maximum distance.
Now, when a *Semantic Pattern* for the distance-based value *31* needs to be generated, the closest node representing the value *32* is selected and the appropriate row from the matrix is selected. The distance values D are then adjusted by calculating $1 - D$ so that the minimum distance, which is equal to 0, is now represented with the maximum value 1, and the maximum distance, which is equal to 1, is represented by the value 0. These adjusted values are then used as the input for a Gaussian function that is influenced by the value σ_p. While smaller values of σ_p keep the pre-activation in the close neighborhood of the activated nodes, larger values for σ_p influence a wider neighborhood. These pre-activation values are then used to activated the network prior to the actual spreading activation process.

Due to the integration of the information gained by applying pre-spreading, the advantages of the semantic analysis are gained while keeping the important distance information between the raw feature values. This comes with two key benefits: **First**, the information represented by the distances between feature values is kept in the *Semantic Patterns*, which enables the possibility to sort the transformed *Semantic Patterns*. This is highlighted by Examples 2 and 3 in Figure 7.14. **Second**, the problems created by complex discretization models are eliminated: This is due the integration of the Gaussian function, which ensures that very close nodes of too complex models receive the right amount of activation energy.

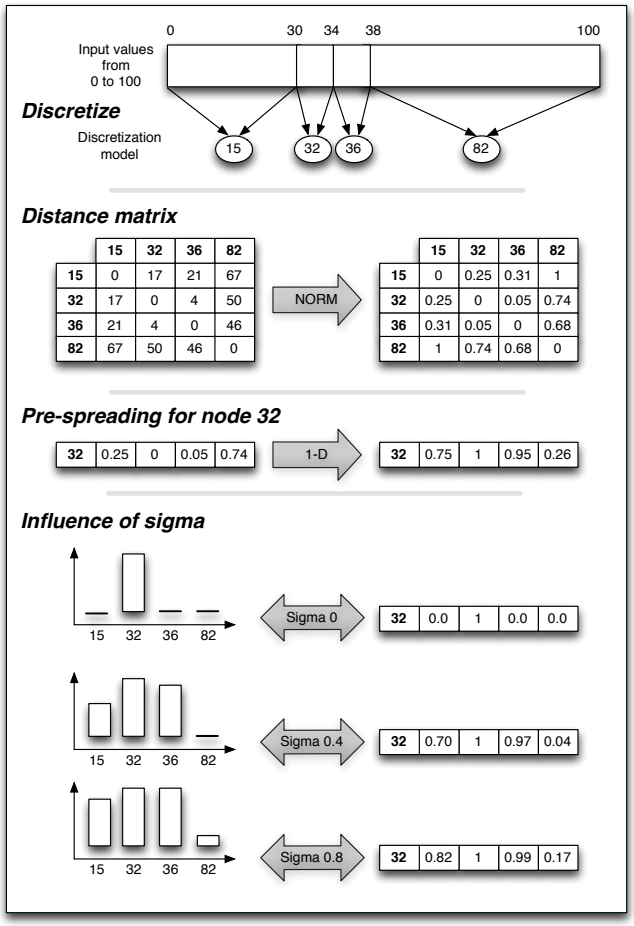

Figure 7.13: Pre-spreading example.

7.3.4 Activation Combination Function

In a typical scenario, where multiple nodes are activated within the network and their activation is spread to their neighboring nodes, the target nodes will receive more than one activation value from the source nodes. In order to combine these activation values, some function needs to be applied, which was already indicated in Algorithm 10. In this thesis, two of such combination functions are implemented: **First**, a function that simply sums up all incoming activation values and adds them to the activation value of the target node, and **second**, a combination function that applies the exponential function to the incoming values before summing them up.

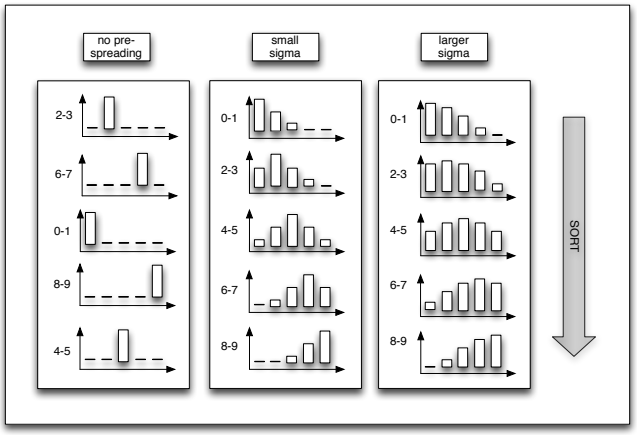

Figure 7.14: Sorting *Semantic Patterns*: **Example 1**: The *Semantic Patterns* of the distance-based feature values cannot be sorted, because they are not related. **Example 2**: Pre-spreading is applied with a smaller σ_p value. The influence of the neighborhood is lower than in **Example 3**, where a larger σ_p value is used for the pre-spreading process. For both examples it is possible to sort the *Semantic Patterns* by applying the Cosine similarity.

SumCombinationFunction

This combination function was selected due to its simplicity and was used solely during the initial applications of the *Semantic Pattern Transformation*. The detailed procedure is explained in Algorithm 13, which sums up all incoming values and adds the result to the activation energy of the target node. The disadvantage of this simple method is the linear model, which does not emphasize nodes that receive activation values from multiple sources.

Algorithm 13 SumCombinationFunction

Require:
A {set of activation values A the target node t with the activation value T receives via the incoming links}
t {target node receiving the activation values}
T {activation value T of target node t}
NET {Associative network NET}

1: **for all** a_i in A **do**
2: $T = T + a_i$ {add activation value a_i to target activation value}
3: **end for**
4: setActivation(NET,t,T) {set activation value T for target node t within the network NET}

ExpCombinationFunction

The rationale behind the exponential combination function described in Algorithm 14 is to emphasize nodes that receive activation values from multiple sources. While the previously described *SumCombinationFunction* has a linear response, this function implements a nonlinear behavior by integrating the exponential function. As will be shown in the subsequent chapter, the utilization of this function yields the better results.

Algorithm 14 ExpCombinationFunction

Require:
A {set of activation values A the target node t with the activation value T receives via the incoming links}
t {target node receiving the activation values}
T {activation value T of target node t}
NET {associative network NET}
1: $T = e^T$
2: **for all** a_i in A **do**
3: $T = T + e^{a_i}$ {add e^{a_i} to the target activation value}
4: **end for**
5: setActivation(NET,t,T) {set activation value T for target node t within the network NET}

Others

During the evaluation of the *Semantic Pattern Tranformation* various other combination functions such as sigmoid ones, or functions that incorporate more sophisticated strategies for combining the activation values have been tested. However, none of these strategies could achieve any significant gain when compared to the *ExpCombinationFunction*. Therefore, further details about these strategies will not be explained.

7.3.5 Fanout

When integrating fanout strategies into the spreading activation algorithm, then the process is considered as constrained-spreading. The main purpose is to avoid the flooding of the network with too much activation energy. Apart from the fanout values, typical measures are related to limiting the spreading activation iterations or enabling path constraints. These two methods were already discussed in Section 5.3.2 of *Chapter 5 – Techniques*, but cannot be utilized for the *Semantic Pattern Transformation*, because only one spreading iteration is executed. The requirement for one iteration is necessary due to the large number of connections within the associative network. Empirical results have shown that the application of a second spreading activation iteration introduces too much additional energy in the network, which severely limits the capability to extract important information.

However, the *Semantic Pattern Transformation* still can implement constrained spreading by using a decay factor, which is already integrated into the standard spreading activation algorithm, and the utilization of fanout strategies. The basic idea of these strategies is to attenuate the influence of nodes that are connected to a large number of other nodes. This process can be applied during three different stages of the spreading activation process, which is explained in the following list. Thereby, a simple example for a highly interconnected node is given in order to show the influence of the different strategies. For this example, it is assumed that the countries of the Eurozone are analyzed, which include the feature *Currency*. Obviously, all of these countries have the *Euro* as currency and therefore the node is linked to all other possible feature values. In order to avoid the noise generated by spreading its activation, a fanout strategy needs to be applied:

- **Before spreading**: Here, the fanout procedures are applied directly after activating the nodes within the network, and thus *before* the application of the spreading activation algorithm. This ensures that the appropriate nodes are already attenuated so that they cannot spread – or only in a limited way – their activation energy during the application of the spreading activation algorithm. In the context of the *Semantic Patterns* and the *Euro* example, this means that the *Euro* node would be immediately attenuated after it is initially activated. Then, it cannot spread its activation energy to neighboring nodes during the spreading activation process. However, the attenuation at this stage also means, that the resulting pattern will only contain a very weak or non-existent activation of the *Euro* node and thus the interpretation of this pattern would not reveal the *Euro* feature value. Depending on the analysis, this might not be the desired outcome.

- **During spreading**: The fanout procedure can also be applied *during* the spreading activation process, when the activation energy is received by the the connected nodes. This ensures that non-relevant nodes are limited in their ability to spread the activation but still remain relevant within the resulting *Semantic Pattern*. For the *Euro* example this means that the corresponding node is activated but does not spread its activation energy to neighboring nodes. As a result, the interpretation of the *Semantic Pattern* will still reveal the feature value *Euro*, but due to strongly attenuating the spread energy, the amount of noise is reduced in the resulting pattern.

- **After spreading**: The same strategy can also be applied *after* the spreading activation process. Here, the activation values of the resulting *Semantic Pattern* would be influenced by the fanout values. For the *Euro* example this means: The activation value for the *Euro* feature value will not be shown in the interpretation of the resulting pattern, but its energy is still spread during the spreading activation process. Therefore, this strategy should not be used as a stand-alone fanout strategy, but in combination with one or both of the previously described methods. The reason is that, even when applying the *before* and *during* strategies, the *Euro* node will

still receive activation energy from other nodes causing a large activation of this node within the resulting pattern. The activation values of such nodes can then be reduced by applying the fanout strategy to the final *Semantic Pattern*.

The deployment of one or a combination of these different strategies depends on the specific analysis, but in general the application of fanout values during the spreading activation process is considered to reveal the best results. The reason is that the nodes subject to attenuation will still be visible in the final *Semantic Pattern*, but do not add noise to the network.

The questions regarding the calculation of these fanout values still remain: **First**, how is it decided whether a node is highly interconnected? There is no absolute measure for a fanout value, because it depends on the size and the structure of the network. **Second**, is the fanout value based on the number of connections, or the weights of the links? In the second case, are the incoming links or the outgoing links of a node considered? This is important, because the link weights are not symmetrical when local network normalization strategies like the ones that will be described in Section 7.3.6 are used.

Since the nodes of the associative networks employed by the *Semantic Pattern Transformation* are highly interconnected, the number of connections does not yield enough information for the determination of adequate fanout values. Thus, the focus needs to be placed on the total activation energy a node can spread or receive via its weighted links. This energy is simply calculated by summing up the weights of all connected links. Whether this calculation is based on the incoming or outgoing links can be defined as a parameter, but typically the outgoing links are used.

Based on these thoughts, Algorithm 15 is employed by the *Semantic Pattern Transformation* for calculating the fanout values of the network nodes. The basic idea behind this algorithm is to calculate the total activation energy values of all nodes and penalize those that strongly exceed the mean total activation energy value of the network:

- *Input*:
  - The network's meta-information structure *NET_I*.
  - The associative network *NET*.
  - The parameter σ_f that influences the shape of the attenuation function.
- *Output*: The fanout values for all nodes within the network are stored within the network's meta-information structure *NET_I*.
- *Line 1*: Initialization of the list E_T that contains the total activation energy values of each node within the network. These values are then used to calculate the mean value and the standard deviation which are the basis for determining whether a node is connected to an exceptionally high number of other nodes.

Algorithm 15 Fanout value calculation

Require:
NET_I {network meta-information structure}
NET {trained associative network NET}
σ_f {factor that influences the shape of the fanout attenuation}

```
1: E_t = {} {list that contains the total activation energy values of all nodes }
2: for all a_i in NET do
3:    L = getLinks(a_i) {get links of node a_i}
4:    E = 0 {init total activation energy}
5:    for all l_i in L do
6:       w_i = getWeight(l_i) {get weight of link l_i}
7:       E = E + w_i {add link weight to total activation energy of node a_i}
8:    end for
9:    storeTotalActivationEnergy(E_t,E) {store total activation energy within the list}
10: end for
11: M = mean(E_t) {mean value of all activation energy values}
12: D = deviation(E_t) {standard deviation of all activation energy values}
13: for all a_i in NET do
14:    E = getTotalActivationEnergy(a_i,E_t) {get total activation energy of a_i}
15:    d = E − M {calculate deviation from mean value}
16:    d = d/D {norm d by standard deviation}
17:    FO = e^(−(d^2−2)^2 / (2σ_f^2)) {calculated fanout value}
18:    if d < 0 then
19:       FO = 2 − FO {emphasize nodes with a small total activation energy}
20:    end if
21:    storeFanoutValue(NET_I,a_i,FO) {store the calculated fanout value for node a_i}
22: end for
```

- *Line 2 to 10*: The total activation energy values for all nodes are calculated according to the following procedure:
- *Line 3*: The links L of the node a_i are extracted. Whether these are the incoming or outgoing links depends on the selected strategy.
- *Line 4 to 8*: The total activation energy E is determined by summing all the weights of the links contained within L.
- *Line 9*: The total activation energy E of node a_i is stored in the list E_t.
- *Line 11 to 12*: The mean value M and the standard deviation D for all total activation energy values are calculated.
- *Line 13 to 22*: The following procedure calculates the fanout values for all nodes within the network based on M, D and the factor σ_f, which is used to shape the attenuation function.
- *Line 14*: The total activation energy E for the node a_i is extracted from E_T.

- *Line 15 to 16*: Now, the deviation d between the mean value M and the total activation energy E is calculated, and then normed with the standard deviation D.

- *Line 17*: The value d is transformed via the Gaussian function in order to calculate the fanout value FO for node a_i. The parameter σ_f influences the shape of this function, which is depicted in Figure 7.15.

- *Line 18 to 20*: When the deviation d is smaller than 0, which means that the total activation energy of the node is below the mean value M, the node is emphasized. This is based on the inverted Gaussian function as depicted in Figure 7.15.

- *Line 21*: Finally, the fanout value FO for node a_i is stored within the network's meta-information structure *NET_I*.

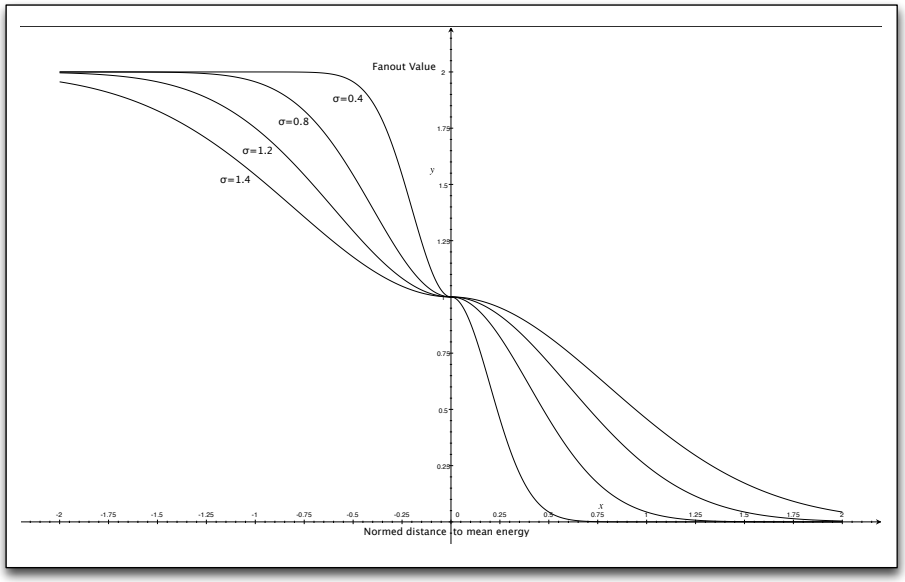

Figure 7.15: The figure shows the normed distance between the total activation energy of the given node and the mean network value on the x-axis. The calculated fanout value is represented by the y-axis and the influence of the factor σ_f is shown for the values $0.4, 0.8, 1.2$, and 1.6.

The fanout values calculated by the described algorithm are then applied either *before*, *during*, or *after* the spreading process. This was already highlighted in Algorithm 10, which describes the spreading activation process employed by the *Semantic Pattern Transformation*. An analysis on the impact of these fanout strategies will be given in the next chapter.

7.3.6 Associative Network Construction

The associative network used within the *Semantic Pattern Transformation* is constructed within *Layers* 2 and 3, which were described in *Chapter 6 – Semantic Pattern Transformation*: **First**, the network nodes are generated by applying a mapping operation to categorical and distance-based feature values. **Second**, the weighted network links are generated by analyzing the co-occurrences of the feature values within the analyzed instances. The strength of the links is determined by the number of co-occurrences within the instances. For the initial weight calculation the counted values are directly used. However, for the application of the spreading activation process in the subsequent layers, the weights need to be normed so that they do not exceed the maximal value of 1.0. The *Semantic Pattern Transformation* currently implements three different normalization strategies which are described as follows:

- **GlobalMaxNorm**: Here, the maximum co-occurrence count of the network's links is taken and all links weights are normed by this value. This strategy suffers through the following problem: If very strong links exists that greatly exceed the mean link weight of the network, then the nodes connected by these links will receive significant higher activation values than the other nodes. Since, the *Semantic Patterns* are compared by employing standard distances measures like the Euclidean distance or the Cosine similarity, this large activation values have much more influence on the distance calculation than the smaller ones received by nodes connected with weaker links. Thus, it might not be possible to utilize nodes that do not occur often, but therefore provide vital information for the *Semantic Patterns*. This effect could be partly compensated by including fanout strategies but the problem of possible large weight differences will still remain. In fact, the initial application of this strategy within the *Semantic Pattern Transformation* did not yield acceptable results and thus was not further investigated.

- **LocalMaxNorm:** Here, a local normalization strategy is applied for each node. The weights of the links outgoing from a node are normalized by the maximum weight of these links. This procedure is applied to each node, which then has at least one outgoing link with a weight equal to 1.0. While this strategy ensures that nodes are emphasized that represent feature values, which do not occur often within the instances, one must chose an appropriate decay factor in order to avoid network flooding.

- **LocalSumNorm:** Similar to the *LocalMaxNorm*, this function is based on a local strategy that considers the weighted links outgoing from a node. Here, the weights of these links are normed by the sum of the weights of all outgoing links. The biggest advantage of this strategy is its intrinsic inclusion of a fanout factor. When a node is connected to a large number of other nodes, then the weights of the individual links automatically get penalized due to the distribution of the total weight over many links.

7.4 Analysis

The remainder of this chapter describes the main analysis processes that were used for the different applications of the *Semantic Pattern Transformation*.

7.4.1 Unsupervised Clustering

Standard unsupervised clustering algorithms can be directly applied to the *Semantic Patterns* that were generated by applying the *Semantic Pattern Transformation* to feature vectors from arbitrary data sets. By varying the model complexity of the employed unsupervised learning algorithm, very coarse grained models for gaining a quick overview, or more complex ones for the detailed representation of the analyzed data, can be trained. The extracted clusters can then easily be analyzed and interpreted by the techniques described in the previous sections. The *Semantic Patterns* remove the need to adapt this interpretation to the employed algorithms and the nature of the data.

1. **Generating the initial *Semantic Patterns***: The *Semantic Pattern Transformation* is applied to the instances and the associative network for the subsequent pattern generation is trained. Depending on the analysis, the patterns can be generated for the complete instances, for single feature values or arbitrary combinations or feature values. The clustering of *Semantic Patterns* generated for single feature values can be utilized to categorize these values according to their semantic fingerprints.

2. **Application of unsupervised clustering**: Without any further preprocessing, the selected unsupervised clustering algorithm is applied to the *Semantic Patterns*.

3. **Interpretation**: The gained clusters and the contained patterns can then be analyzed according to the methods described in the previous chapters. Especially, determining the mean and variance values of the patterns contained in a cluster helps to gain a quick overview on the most important feature values.

In order to highlight the unsupervised clustering process and the influence of the pre-spreading parameter σ_p, a simple example based on four distinct clusters is presented. This example was originally presented by the author of this thesis in [50], and adapted for the following presentation. Thereby, the analyzed two-dimensional data is visualized in Figure 7.16 and consists of two distance-based features. The two-dimensional feature vectors are transformed into *Semantic Patterns* by executing the four 4 layers of the *Semantic Pattern Transformation*:

- ***Layer 1*:** The data set consists of 2D data-vectors representing two different features. In this case, there are only *distance-based features* meaning that the values of these features can be related with a distance measure.

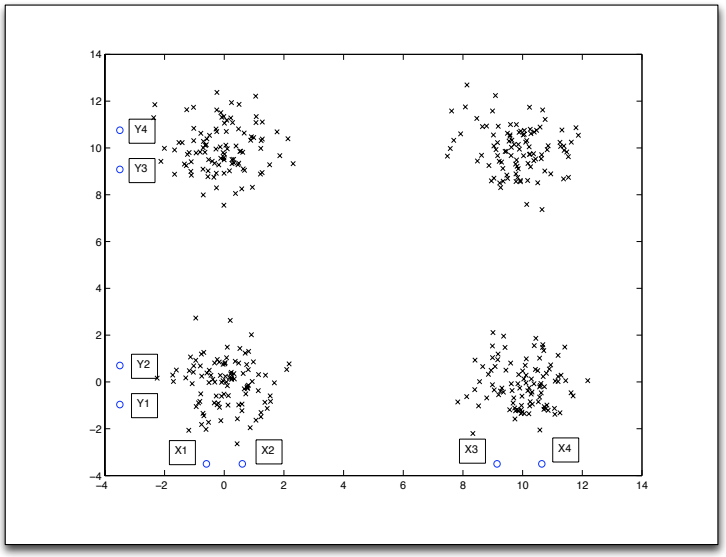

Figure 7.16: Artificial dataset consisting of four distinct clusters. There are four RGNG prototypes for each dimension (X_1 to X_4 for the first feature and Y_1 to Y_4 for the second feature).

- ***Layer 2*:** In this layer the nodes for the associative network are created. Since both features belong to the category of *distance-based features*, a discretization operation needs to be applied. Therefore, the RGNG algorithm is applied to the values of both features. The resulting prototypes for each model are depicted in Figure 7.16[3] (X_1 to X_4, and Y_1 to Y_4). These eight prototypes of both trained models represent the eight nodes of the associative network that will be trained in the next layer. In order to show the influence of pre-spreading and the associated parameter σ_p, a complex model consisting of four prototypes per features is generated. Here, a model with two prototypes per feature would be sufficient, and would also be found by the MDL criterion.

- ***Layer 3*:** Now, the links of the associative network are created and their strength is determined by analyzing the co-occurrence information of the two features within the training data.

- ***Layer 4*:** Now, the spreading activation algorithm is applied to generate the *Semantic Patterns* for each instance within the training set. In order to show the influence of the pre-spreading procedure, two sets of *Semantic Patterns* are generated – one for $\sigma_p = 0.8$ and one for $\sigma_p = 0.1$.

[3]The depicted prototypes are 1D prototypes. When the data is projected on the x-axis, or on the y-axis one can see that the four prototypes of each model are at the center of gravity of the projected data vectors.

- ***Layer 5:*** Now, an unsupervised clustering algorithm is applied to the generated *Semantic Patterns*. Here, the Self-Organizing Map (SOM) was chosen due to its data visualization capabilities.

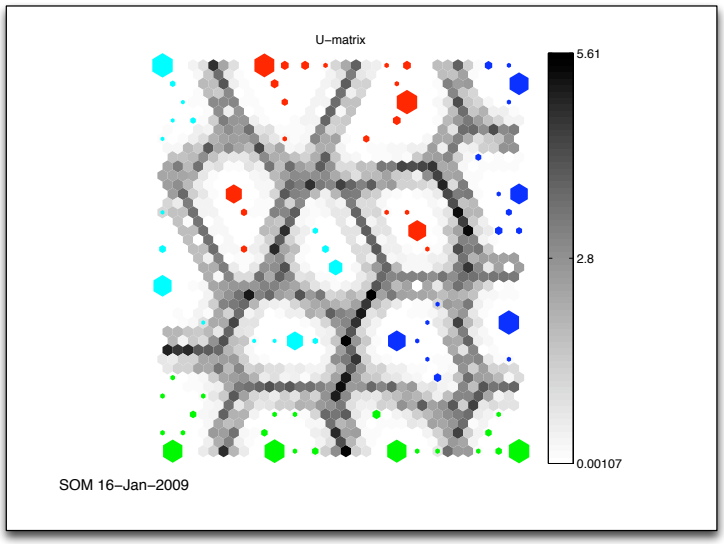

Figure 7.17: Visualization of the map trained on the *Semantic Patterns* with pre-spreading parameter $\sigma_p = 0.1$.

The results of applying the SOM algorithm to the two pattern sets are shown in Figures 7.17 for $\sigma_p = 0.1$ and Figure 7.18 for $\sigma_p = 0.8$. The instances of the four clusters are marked by four different colors. Within the map, the dark areas indicate large distances between the analyzed instances and can be used for the identification of clusters. In contrast, the distances between the instances within a bright area are fairly small. Thus, the first map indicates that 16 clusters are found, while the correct number of four clusters is observed within the second map: This difference is explained by the pre-spreading factor σ_p: According to the explanations given in Section 7.3.3, the larger σ_p value compensates the problems associated with a complex discretization model by increasing the influence on the neighboring nodes within the pre-spreading process.

7.4.2 Supervised Learning

When class labels are available for the instances, supervised learning algorithms can be applied to the *Semantic Patterns* in order to train a classifier. Similar to unsupervised learning, the *Semantic Pattern Transformation* process eliminates the need for preprocessing operations required by many supervised learning algorithms. As for unsupervised learning, the resulting classes of patterns can easily be interpreted without the need to analyze the model of the algorithm.

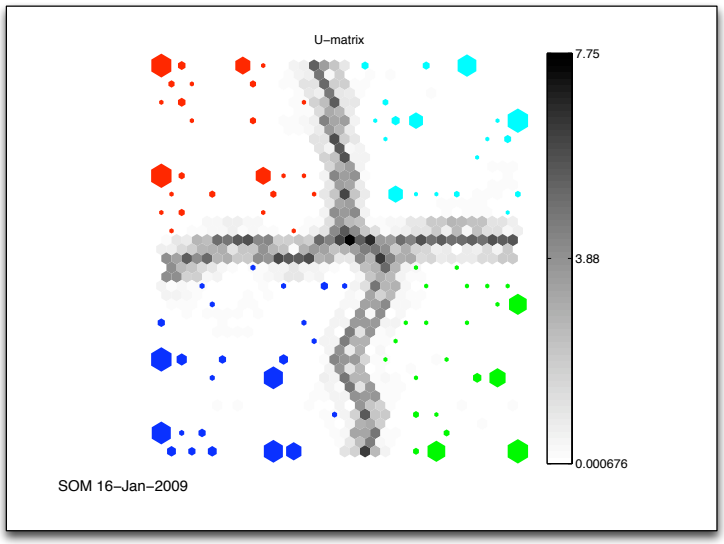

Figure 7.18: Visualization of the map trained on the *Semantic Patterns* with pre-spreading parameter $\sigma_p = 0.8$.

7.4.3 Semantic Search

The semantic similarity between patterns is the basis for the application of semantic-aware search algorithms. By activating the nodes of one or more feature values contained in the search query, and applying spreading activation, a search pattern is generated. Similar semantic similar patterns can then be retrieved from an existing pattern data base according to their similarity with the search pattern. The whole process is highlighted by Example 18 and described as follows:

1. **Generating the initial *Semantic Patterns***: Before one is able to apply semantic search queries, the *Semantic Patterns* of the search data base need to be generated. This could either be the patterns generated by applying the *Semantic Pattern Transformation* to complete instances (e.g documents), single feature values (e.g., terms) or an arbitrary mixture of multiple features values (e.g., sentences)[4].

2. **Definition of the semantic search query**: The semantic search query needs to be defined either by specifying a feature value, multiple feature values, or a complete instance. Obviously, when symbolic feature values are involved, only those nodes of the associative network can be retrieved that perfectly match the feature values within the search query. This is different when distance-based feature values are used within the search

[4]The generation of patterns for single feature values only works when these feature values have been put into some semantic relation before – e.g. via the analyzed instances.

query. Due to the possibility to calculate a distance to the nodes representing the distance-based feature values, a best matching node always can be retrieved. However, one needs to consider that the network nodes can only cover those feature value ranges that were present during the training. Therefore, feature values that are larger than the node representing the largest value within the network, will always be mapped to the same node regardless of their value.

3. **Generation of the *Semantic Pattern* for the semantic search query**: The *Semantic Pattern Transformation* is applied to the input feature values and a *Semantic Patterns* representing the search query is generated.

4. **Retrieval of the search results**: The Cosine similarity between this "search pattern" and all of the existing *Semantic Patterns* is then calculated and the results are sorted according to their similarity.

Example XVIII: Examples for semantic search queries

Assuming that *Semantic Patterns* have been generated for various news articles covering stories about US related politics of the time when this thesis was written[a]: Then, executing a search query for the term *president* would retrieved all news articles that contain the term *president*, or any other semantically related term such as *Obama*, *Barack* or *health reform*. This concept can be applied to arbitrary *Semantic Patterns* generated by single terms, complete sentences or even documents.

[a]End of 2011

7.4.4 Anomaly Detection

Anomaly detection aims to find outliers within a data set or instances that were not seen during training and deviate from the trained model. Since the response of the network – represented as *Semantic Patterns* – depends on the input-stimuli given with feature values, and the links between the network nodes, non-typical combinations of input feature values can be identified by analyzing the total activation energy of the pattern. Depending on the analysis, too high or too low energies can be considered as anomaly.

7.4.5 Semantic Relations

The associative network describes arbitrary relations between feature values. By activating one or more nodes (corresponding to feature values) within the associative network, and spreading their activation via the links to the neighbors,

details about the semantic relations between various feature values can be extracted. Many examples for this analysis process have already been shown in Section 7.1.1.

7.4.6 Feature Relevance

The relevance of a node or its represented feature value depends on its relations to other feature values. Thereby, a node that has a large number of connections is considered as non-relevant. When taking a closer look at this problem, one observes that the requirements for determining the relevance values are strongly related to the determination of the fanout values, which was discussed in Section 7.3.5. Nodes that are linked to a large number of other nodes within the network should be limited in their capability to spread activation energy. In a similar way these nodes are not considered as relevant within the network. Therefore, in order to determine the feature value relevance, the fanout values, which are calculated after generating the associative network, can directly be used. By sorting these values, a quick overview about the relevancy of the feature values can be gained.

It is important to note that here, the relevance of a node refers to its capability to provide useful information within the spreading activation process that distributes its activation energy to the neighboring nodes. While in this sense the feature value *Europe* is not relevant for participating in the spreading activation process, the information carried by the node itself might still be relevant. The removal of the feature value *Europe* would be beneficial for the various analysis processes, but would also remove the capability of the analyst to identify the instances as European countries. Thus, a non-relevant feature value should not be removed from the instances but only its influence to other nodes.

7.4.7 Time-Based Analysis Processes

When timestamps are available, this information can be integrated in the same way as for other features into the associative network. The integration of such information enables the analysis of semantic development over time.

7.5 Chapter Conclusions

This chapter describes various methods for interpreting and analyzing *Semantic Patterns*. Due to the employed vector representation and the semantic information contained in the patterns, simple operations like filtering, sorting, adding, subtracting, or distance measures can be applied to the *Semantic Patterns*. Furthermore, the *Semantic Patterns* use a common representation, which enables the application of these and more sophisticated methods like supervised or unsupervised learning in the same way, regardless of the underlying data and the deployed knowledge discovery process.

8 Semantic Patterns - Evaluation

In this chapter, the *Semantic Pattern Transformation* is evaluated by applying supervised and unsupervised machine learning algorithms, and a semantic search algorithm. Thereby, the main focus is placed on the various parameters that influence the transformation process. All of the evaluations are based on data sets extracted from the UCI Machine Learning Repository [28], and include nine data sets based on categorical features, ten data sets based on numerical features, and seven data sets that contain both feature types. Before presenting the evaluation results, the required assumptions and the evaluation environment will be discussed, and a short overview of the conducted evaluations will be given.

8.1 Evaluation Environment

This section gives a quick overview of the evaluation scenarios and the data sets covered in this chapter. Furthermore, the employed quality measure and the evaluation environment are described.

8.1.1 Weka

According to the project webpage Weka is described as *"Weka is a collection of machine learning algorithms for data mining tasks. The algorithms can either be applied directly to a dataset or called from your own Java code. Weka contains tools for data pre-processing, classification, regression, clustering, association rules, and visualization. It is also well-suited for developing new machine learning schemes."*

Weka and the Weka API, which can be directly used from a Java program

were heavily used during the evaluation of the supervised and unsupervised learning algorithms. Thereby, the Weka implementations of the *K-Means*, *EM*, *Support Vector Machine (SVM)* (*SMO* in Weka), and *C4.5* (*J48* in Weka) algorithms were deployed.

For a detailed description of Weka and knowledge discovery, or data mining in general, the reader is referred to the book of Witten et. al [90].

8.1.2 Evaluation – At a Glance

In order to systematically evaluate the *Semantic Pattern Transformation*, and the various parameters that influence the transformation process, the following procedures are carried out by utilizing Weka [90]:

- **Unsupervised learning**: The performance of the unsupervised learning algorithms *EM* and *K-Means* are evaluated by applying the algorithms to categorical, numerical and mixed data sets from the UCI Machine Learning Repository. Thereby, different parameter sets are used to configure the *Semantic Pattern Transformation*, which is then applied to the raw data sets. The performance is determined by employing the V-quality-measure.

- **Supervised learning**: For the evaluation of the performance of supervised classification algorithms the supervised *Support Vector Machine* algorithm, and the decision tree based *J48* algorithm are applied to the raw data and the *Semantic Patterns*. In order to limit the large number of possible parameter combinations for the *Semantic Pattern Transformation*, the best results from the unsupervised learning evaluation are taken and used as a basis for the supervised evaluation.

- **Semantic search**: In order to evaluate the usability of the *Semantic Pattern Transformation* for the implementation of a semantic search algorithm, the same UCI Machine Learning Repository data sets are utilized. Thereby, the usability of the *Semantic Patterns* as basis for a semantic-aware search algorithm is determined by calculating the distances between the instances of a given data set. Based on these distances, which describe the similarity between the patterns, and the available class labels, the error rates can be determined and compared for the raw data and the *Semantic Patterns*. The process is conducted for the Euclidean distance and the Cosine similarity.

- **Influence of missing values**: Finally, the influence of missing values on the raw data and the *Semantic Patterns* is evaluated. For this procedure the best results from the other evaluations have been taken and different amounts of feature values have been removed from the instances.

- **Fanout strategies**: Also, the evaluation of unsupervised and supervised machine learning algorithms has been extended by employing the fanout strategies discussed in Section 7.3.5 of *Chapter 7 – Semantic Pattern Analysis*. In order to limit the possible combinations of input parameters, the

fanout values have only been evaluated for the parameters that achieved the best results in the standard evaluation. However, none of the possible fanout parameter combination could increase the quality of the results. Thus, the application of these strategies will only be discussed in terms of the employed parameters, and within the conclusions of this chapter.

- **Other evaluations**: Other evaluations of advanced techniques like anomaly detection, feature relevance and the applicability of methods that reduce the dimensionality of the *Semantic Patterns* are beyond the scope of this thesis and subject of future work. For the empirical evaluation of the *Semantic Pattern Transformation*, the reader is referred to the descriptions of the published works in *Chapter 10 – Applications*, which describe the deployment of the *Semantic Patterns* in a wide range of domains.

8.1.3 Quality Measure: V-Measure

For the evaluation of the supervised and unsupervised results, the V-Measure is used for determining the quality of the results. This measure was chosen due to the advantages described in [69], when compared to the other common measures such as F-Measure or the External Entropy.

8.1.4 *Semantic Pattern* Parameters

In summary, the different methods employed within the *Semantic Pattern Transformation* are configured by various parameters that influence the generation of the associative network and the spreading activation process. In order to get an overview of the different parameters that are used within the evaluation processes, the following summary is given. Thereby, the abbreviations in parentheses will later be used for the discussion of the gained results.

- **Associative network**: The following parameters are related to the construction of the associative network:
  - **Link normalization (Norm)**: Two link normalization strategies are available for the generation of the network: The **LocalMaxNorm** (**Norm=L**) strategy, which applies a local maximum norm to the links of each node, and the **SumLocalNorm** (**Norm=S**) strategy, which norms the links of each node according to the sum of all link weights. For details regarding these methods the reader is referred to Section 7.3.4 of the previous chapter.
  - **Discretization model complexity MDL override for distance-based features (*C*)**: This is the MDL override factor that directly influences the complexity of the discretization model. For further information the reader is referred to Section 5.6 of *Chapter 5 – Techniques*, and the description of the pre-spreading procedure for distance-based features described in Section 7.3.3 of the previous chapter.

- **Spreading activation**: The following parameters are related to the spreading activation process (discussed in Section 7.3 of the previous chapter), which is applied to the trained network:
  - **Decay factor *D***: The decay value D influences the amount of energy spread to the neighboring nodes and plays an important role for constrained-spreading, which avoids the flooding of the network.
  - **Combination function (Comb)**: This function is used to combine multiple activation values that a network node receives. Here, the linear version **SumCombinationFunction** (**Comb=S**) and the nonlinear variant **ExpCombinationFunction** (**Comb=E**) are evaluated. For details regarding these methods the reader is referred to Section 7.3.4 of the previous chapter.
  - **Pre-spreading (σ_p)**: The influence of the pre-spreading process on the neighborhood is configured by the factor σ_p. Details on this process are covered in Section 7.3.3 of the previous chapter.
- **Fanout strategy**: The calculation of the fanout values employed in constrained-spreading and their application are influenced by the following parameters (as discussed in Section 7.3.5 of the previous chapter):
  - **Links for fanout calculation (FanLinks)**: This strategy determines whether the incoming or outgoing links of a node are considered for the calculation of its fanout value.
  - **Fanout shaping (σ_f)**: This factor shapes the function used for the calculation of the fanout values.
  - **Fanout application (FanApp)**: The fanout values can either be applied **before**, **during**, or **after** applying spreading activation.

8.1.5 Data Sets

The UCI Machine Learning Repository data sets used for the evaluation procedures covered in this chapter are listed in Table 8.1. The data set names are covered in the first column (**Data set**), and will further be identified by the labels listed in the second column (**Label**). Thereby, the data sets can be grouped into three main categories according to the contained features – **Categorical**, **Mixed** and **Numerical**. Additional information about the data sets is presented in columns three to six, which include the number of instances (**Inst**), the number of distance-based features (**DF**), the number of symbolic features (**SF**), and the number of classes within the respective data set (**Classes**). Furthermore, the best results for the supervised *Support Vector Machine* (**SVM**) algorithm, and the unsupervised algorithms *K-Means* (**KM**) and *Expectation Maximization* (**EM**) are given for each data set. Thereby, the labels within the parentheses represent the type of data, which is analyzed by the respective algorithm: **NN** indicates the raw data that is not normed, **N** represents the normed raw data, and **P** means that the data is transformed into *Semantic Patterns* by

applying the *Semantic Pattern Transformation*. For each of these categories, the V-Measure results of the respective algorithms are listed, where the bold results indicate the best results for the respective algorithm (*SVM*, *K-Means* or *EM*). Also, a total result (**Total**) is gained by calculating the mean value for the individual results and presented for each algorithm and data set category. In case of the *Semantic Patterns* (**P**), the results for the *Semantic Patterns* generated via the parameter-set described in the third row of the table are listed.

Data set	Label	Inst	DF	SF	Classes	SVM (N)	SVM (NN)	SVM (P)	KM (N)	KM (NN)	KM (P)	EM (NN)	EM (P)
						SVM			K-Means			EM	
						SP-Parameters: D=0.5, Comb=E, Norm=L, MDL=1.5, σ = 0.2							
		Categorical											
Breast Cancer	BC	286		9	2	0.03	**0.04**	**0.04**	0.01	0.01	**0.06**	0.00	**0.08**
Dermatology	DE	366	1	33	6	0.93	0.92	**0.95**	0.58	0.09	**0.86**	**0.87**	**0.87**
KR vs. KP	KR	3196		36	2	**0.75**	**0.75**	0.72	0.00	**0.01**	0.00	**0.04**	0.00
Lymph	LY	148		18	4	**0.53**	0.51	0.48	0.13	0.18	**0.25**	0.26	**0.27**
Mushroom	MU	8124		22	2	**1.00**	**1.00**	**1.00**	**0.48**	0.47	0.45	**0.61**	0.59
Soybean	SO	683		35	19	0.92	0.92	**0.93**	0.59	0.62	**0.73**	**0.79**	**0.79**
Splice	SP	3190		60	3	0.71	0.72	**0.80**	0.03	0.03	**0.44**	**0.41**	0.31
Vote	VO	435		16	2	**0.76**	0.74	0.67	0.47	**0.48**	0.47	**0.49**	0.45
Zoo	ZO	101		17	7	0.94	0.94	**0.97**	0.78	0.78	**0.82**	0.82	**0.85**
		Total				**0.73**	**0.73**	**0.73**	0.34	0.30	**0.45**	**0.48**	0.47
		Mixed											
Anneal	AN	898	6	32	6	0.86	0.86	**0.92**	0.23	0.03	**0.30**	0.31	**0.32**
Colic	CO	368	7	15	2	0.31	**0.32**	0.31	**0.13**	0.03	0.05	0.10	**0.12**
Credit-A	CA	689	6	9	2	**0.41**	**0.41**	0.39	0.16	0.02	**0.25**	0.17	**0.21**
Credit-G	CG	1000	7	13	2	0.11	0.10	**0.12**	**0.01**	**0.01**	0.00	0.01	**0.02**
Heart-C	HC	303	6	7	5	**0.36**	**0.36**	0.29	0.24	0.01	**0.36**	**0.31**	0.28
Heart-H	HH	294	6	7	5	0.32	0.31	**0.33**	0.27	0.01	**0.32**	**0.28**	0.25
Hepatitis	HE	155	5	14	2	0.25	**0.28**	0.21	0.13	0.00	**0.21**	0.22	**0.24**
		Total				0.37	**0.38**	0.37	0.17	0.02	**0.21**	0.20	**0.20**
		Numerical											
Breast-w	BW	699	9		2	**0.78**	**0.78**	0.77	0.73	0.74	**0.82**	**0.72**	0.58
Diabetes	DI	768	8		2	**0.18**	**0.18**	0.15	0.05	0.03	**0.10**	**0.10**	0.08
Glass	GL	214	9		7	0.30	0.30	**0.50**	0.34	**0.39**	0.33	**0.37**	0.36
Heart-Statlog	HS	270	13		2	**0.36**	**0.36**	0.37	0.25	0.02	**0.39**	**0.29**	0.27
Ionosphere	IO	351	34		2	0.48	0.48	**0.50**	0.12	0.12	**0.16**	**0.25**	**0.25**
Iris	IR	150	4		3	0.87	0.87	**0.87**	0.71	0.71	**0.75**	**0.81**	0.78
Segment	SE	2310	19		7	0.88	0.88	**0.90**	**0.61**	0.53	0.59	**0.62**	0.60
Sonar	SO	208	60		2	0.23	0.23	**0.23**	0.01	0.01	**0.02**	**0.01**	**0.01**
Vehicle	VE	846	18		4	**0.51**	**0.51**	0.48	0.11	**0.19**	**0.19**	0.10	**0.19**
Vowel	VO	990	10	3	11	0.63	0.63	**0.76**	0.06	**0.34**	0.23	0.19	**0.25**
		Total				0.52	0.52	**0.55**	0.30	0.31	**0.36**	**0.35**	0.34

Table 8.1: Data sets that are used for the subsequent evaluation of supervised and unsupervised algorithms. The V-Measure results of these evaluations are also listed in the table.

8.1.6 Algorithm Setup and Tools

For the evaluation, the *K-Means*, *EM*, *J48*, and *SMO* algorithms are utilized within a Java program that uses the Weka 3.6 API [90]. Thereby, in order to calculate the V-Measure, the Weka functionality is extended by the cluster eval-

uation tools provided by the authors of [69]. Only the following parameters of the employed algorithms were changed: For the unsupervised algorithms, the number of clusters was set to the number of classes contained within the analyzed data sets. For all algorithms, when available, the normalization of the raw data is activated or deactivated as listed in the result tables. Furthermore, each algorithm was applied to the respective data for ten iterations by selecting a different seed value for each iteration. In case of the *Semantic Patterns*, the ARFF files[1] of the respective data sets are transformed into the *Semantic Patterns* and exported as ARFF formatted files again, which then can be read by the Weka API.

8.2 Unsupervised Learning Evaluation

For this evaluation, the two unsupervised clustering algorithms *K-Means* and *EM* are applied to the raw data and to the *Semantic Patterns*. Since the *Semantic Pattern Transformation* is influenced by a wide range of parameters that yield a huge number of different combinations, the following approach is used:

First, a baseline configuration that completely deactivates spreading activation, is deployed. This is achieved by setting the decay value to zero, which eliminates the spreading of activation energy from activated nodes to their neighbors. Thereby, most other parameters become irrelevant, because they are only applicable when spreading activation is activated. This baseline configuration corresponds to a simple binary feature vector representation, where the presence of a feature value is indicated with 1.0 and its absence by 0.0.

Second, the influence of spreading activation is evaluated by utilizing various combinations of parameters. Thereby, in order to reduce the number of test runs, the first tests are aimed to eliminate certain parameters that have a negative influence on the performance. The focus is then placed on the remaining parameters. The complete evaluation procedure is based on the following steps:

- *K-Means* and *EM* are applied to the raw value-centric feature vectors of the respective data sets.
- The raw feature vectors of the analyzed data sets are transformed into baseline *Semantic Patterns*. Here, spreading activation is deactivated by setting $D = 0.0$. The two algorithms are then applied to the gained baseline (binary) *Semantic Patterns*, and the corresponding V-Measure and standard deviation values are determined.
- Finally, the real *Semantic Patterns* are generated by using different parameters for the *Semantic Pattern Transformation*. The evaluated parameters depend on the type of analyzed data. In that sense, the evaluation of pre-spreading techniques can only be conducted for the mixed and numerical data sets due to the availability of distance-based features.

[1] Attribute-Relation File Format (ARFF): The data sets and the contained instances read by Weka are stored in this file format.

- The different results are then analyzed by discussing the gained observations.

8.2.1 Result Tables

The subsequent sections will discuss the results gained by utilizing different parameters for the generation of the *Semantic Patterns*. Thereby, the tables used for presenting the results are organized as follows: There is one table that lists the V-Measure results of the corresponding evaluation, followed by a table that contains the associated standard deviation values for the 10 iterations of each algorithm and parameter combination. In addition to the individual results, there is always a total result that is gained by calculating the mean value of the individual results. The rationale behind these total values is to get a quick overview of the overall performance when evaluating a parameter set used for the analysis of the individual data sets.

In all tables, the best results for a group are highlighted by using a bold font. For the V-Measure values the best results are represented by the largest values, whereas for the standard deviation values the lowest ones are marked. The latter give an indication on the robustness of the gained results.

8.2.2 Categorical Data

Nine categorical data sets from the UCI Machine Learning Repository are used for the evaluation of the unsupervised learning algorithms. For the categorical data all parameters that influence the pre-spreading process become irrelevant due to the lack of distance-based features. The reduced number of parameters simplifies the evaluation process, which focuses on the evaluation of the decay factors, the link normalization strategy, and the combination function.

Baseline

For the baseline *Semantic Patterns*, which are generated for the categorical data sets, the following parameters are chosen:

- **Decay factors**: $D = \{0.0\}$. For the baseline configuration $D = 0.0$ is used, which completely deactivates the spreading activation algorithm and makes the combination function and link normalization parameters irrelevant.

- **Others:** The parameters for link normalization (**Norm**) and the combination of the activation values (**Comb**) are not relevant for the baseline *Semantic Pattern*, because of the decay factor $D = 0.0$. Also, the MDL override parameter (**C**) and the pre-spreading parameter σ_p are not considered for the generation of the baseline *Semantic Patterns* due to the lack of distance-based features.

The results gained by applying the *K-Means* and *EM* algorithms to the baseline *Semantic Patterns* are listed in Tables 8.2 and 8.3. Thereby, the following observations are made:

- **Raw data**: For the raw data, the *EM* algorithm significantly outperforms the *K-Means* algorithm. While the total result for the *EM* algorithm is $V = 0.477$ at $V_\sigma = 0.019$, it is only $V = 0.342$ at $V_\sigma = 0.052$ for the *K-Means* algorithm when the data is normed. This performance drops further when the *K-Means* algorithm is applied directly to the raw data, where $V = 0.296$ and $V_\sigma = 0.050$. These results are not surprising due to the higher sophistication of the Gaussian kernels employed by *EM* in comparison to the simple centroid model of the *K-Means* algorithm. This increased sophistication yields better results while also increasing the robustness of the algorithm, which is indicated by the lower standard deviation values.

- **Baseline *Semantic Patterns***: In this case, the *Semantic Patterns* are generated without applying spreading activation, which is achieved by setting the decay factor to $D = 0.0$. The thereby generated patterns correspond to a binary representation where the presence or absence of a feature value is described with the values 1.0 and 0.0 respectively. By using this representation, the total result for the *K-Means* algorithm is $V = 0.443$ at $V_\sigma = 0.058$, which means it is getting closer to that of the *EM* algorithm when applied to the raw data ($V = 0.477$ at $V_\sigma = 0.019$). The result of *K-Means* and *EM* on the baseline *Semantic Patterns* is roughly the same: $V = 0.443$ at $V_\sigma = 0.058$ for *K-Means* and $V = 0.449$ at $V_\sigma = 0.023$ for *EM*. The standard deviation values for *K-Means* and *EM* are slightly higher than for the raw data when applying the algorithms to the baseline *Semantic Patterns*.

	K-Means											EM									
Par	**Total**	**BC**	**DE**	**KR**	**LY**	**MU**	**SO**	**SP**	**VO**	**ZO**		**Total**	**BC**	**DE**	**KR**	**LY**	**MU**	**SO**	**SP**	**VO**	**ZO**
	Raw Data																				
N	0.341	0.012	0.584	0.004	0.131	**0.475**	0.587	0.031	0.467	0.782		Not available									
NN	0.296	0.007	0.094	**0.010**	0.176	0.472	0.616	0.030	0.476	0.783		**0.477**	0.002	**0.871**	**0.036**	**0.258**	**0.610**	**0.789**	0.410	**0.494**	**0.822**
	Semantic Patterns																				
D 0.0	**0.443**	**0.025**	**0.849**	0.003	**0.199**	0.413	**0.728**	**0.465**	**0.493**	**0.814**		0.449	**0.004**	0.767	0.001	0.222	0.590	0.740	**0.423**	0.489	0.801

Table 8.2: Unsupervised V-Measure results for the raw categorical data sets and the respective baseline *Semantic Patterns*.

Semantic Patterns

For this evaluation, the *K-Means* and *EM* algorithms are applied to the *Semantic Patterns* generated for the categorical data sets. In this case, a decay factor larger than zero is used. Therefore, in contrast to the baseline patterns the semantic relations between the feature values are modeled within the generated patterns. Furthermore, the influence of the normalization strategy during the

	K-Means											EM									
Par	**Total**	**BC**	**DE**	**KR**	**LY**	**MU**	**SO**	**SP**	**VO**	**ZO**		**Total**	**BC**	**DE**	**KR**	**LY**	**MU**	**SO**	**SP**	**VO**	**ZO**
	Raw Data																				
N	0.052	**0.015**	0.124	**0.005**	0.054	**0.140**	0.026	**0.015**	0.031	0.054		Not available									
NN	**0.050**	0.025	**0.011**	0.022	**0.022**	0.228	0.035	0.018	0.029	0.058		**0.019**	**0.000**	**0.031**	0.018	0.059	**0.000**	0.024	**0.000**	**0.000**	**0.042**
	Semantic Patterns																				
D 0.0	0.058	0.035	0.059	0.006	0.041	0.245	**0.019**	0.079	**0.005**	**0.033**		0.023	**0.000**	0.053	**0.000**	**0.050**	**0.000**	**0.020**	0.033	**0.000**	0.052

Table 8.3: Unsupervised V-Measure standard deviation results for the raw categorical data sets and the respective baseline *Semantic Patterns*.

construction of the associative network, and the spreading activation combination function are evaluated:

- **Decay factors**: $D = \{0.1, 0.3, 0.5, 0.7\}$. The value $D = 0.0$ is already covered within the baseline patterns. The other values are chosen to cover the complete range of useable decay values $0.0 \leq D \leq 1.0$. Thereby, low decay factors attenuate the influence of the activated nodes on the neighboring ones, which means that the information about the semantic relations carried in the associative network's links is only utilized in a very limited way.

- **Link normalization (Norm)**: These parameters define how the link weights are normed during the generation of the associative network. The first strategy – **LocalMaxNorm** (**Norm=L**) – norms the emerging link weights of each node by the maximum weight of these links. The second strategy – **SumLocalNorm** (**Norm=S**) – norms the outgoing link weights with the sum of the weights of all outgoing links.

- **Combination function (Comb)**: The activation function defines how a target nodes combines the activation values that it receives from multiple source nodes. For the evaluation, two combination functions are used: The first one – **SumLocalNorm** (**Comb=S**) – simply sums up the incoming activation values and sets the gained value as target node activation value. The second one – **ExpSumActivation** (**Comb=E**) – applies the *exp* function to each incoming activation value and then sums them up.

- **MDL override factor (C) and pre-spreading influence** σ_p: Due to the lack of distance-based features within the categorical data sets, the parameters for pre-spreading do not play a role and are not further discussed in this evaluation.

The results gained by applying the *K-Means* and *EM* algorithms to the *Semantic Patterns* are listed in Tables 8.4 and 8.5. Thereby, the following observations are made:

- ***K-Means***: The application of *K-Means* to the *Semantic Patterns* results in a significant performance increase when compared to its application to the raw data ($V = 0.452$ at $V_\sigma = 0.044$ vs. $V = 0.341$ at $V_\sigma = 0.051$).

This is also reflected by the slightly increased robustness as indicated by the lower standard deviations of the *K-Means* iterations for the **Comb=E**, **Norm=L** parameter combination.

- ***EM***: The application of the *EM* algorithm to the *Semantic Patterns* results in roughly the same performance ($V = 0.480$ at $V_\sigma = 0.010$ vs. $V = 0.477$ at $V_\sigma = 0.019$) when using the appropriate normalization strategy and combination function (**Comb=S**, **Norm=S**). Otherwise, there is a slight drop in the results. However, one interesting observation refers to the robustness of the algorithm, which is increased regardless of the employed parameters as indicated by the smaller standard deviation values of the results.

- ***K-Means* vs. *EM***: Although, the performance of the *K-Means* algorithm can be significantly improved when transforming the data into *Semantic Patterns*, the gained results are still below those of the *EM* algorithm: $V = 0.452$ at $V_\sigma = 0.044$ vs. $V = 0.480$ at $V_\sigma = 0.010$.

- **Influence of the combination function - *K-Means***: When looking at the two result blocks that use the parameter **Comb=E**, one observes that the best results and the overall results for the blocks are better than those that use **Comb=S**. The best V-Measure values for both blocks are $V = 0.452$ versus $V = 0.441$ and $V = 0.439$ for the other blocks. The best standard deviation values are achieved by the block that employs **Norm=L** together with **Comb=E**. The reason for the performance increase for the parameter **Comb=E** could be the nonlinear combination of the activation values within the patterns, which in terms of the Euclidean distance function helps to move non-related patterns further apart, while keeping closely related together.

- **Influence of the combination function - *EM***: The observations for the *K-Means* algorithm cannot be applied to the results gained by the *EM* algorithm. There is no significant difference between the results for the two possible combination functions. Interestingly, and in contrast to *K-Means*, the best result is achieved when the parameter **Comb=S** is used in combination with the parameter **Norm=S**. The reason here might be that due to the Gaussian models used by the *EM* algorithm the modeling is much better and does not require the nonlinear combination of the activation values.

- **Influence of the network normalization method - *K-Means***: One observes that the choice of the normalization strategy does not have an influence on the quality of the results. These are pretty much the same regardless of the employed normalization function (**Norm=S** or **Norm=L**). According to the results, the choice of the combination function has a more significant impact. However, at least in terms of robustness it seems that the combination of the parameters **Norm=L** and **Comb=E** has a slight advantage over the combination **Norm=S** and **Comb=E**.

- **Influence of the network normalization method - *EM***: There is no significant influence of the combination function on the gained results.

- **Influence of the decay factor - *K-Means***: The influence of the decay factor depends on the normalization strategy and the employed combination function. While the decay factors $D = 0.3$ and $D = 0.5$ yield the best results for the **(Comb=E, Norm=L)** and **(Comb=E, Norm=S)** combinations, a weak decay factor of $D = 0.1$ seems to be the better choice for the **(Comb=S, Norm=L)** and **(Comb=S, Norm=S)** parameter sets.

  In general, the size of the decay factor determines how strong each feature value influences its neighbors when applying the spreading activation algorithm. The results indicate that for the **(Comb=E, Norm=L)** and **(Comb=E, Norm=S)** combinations too high ($D = 0.7$) or too low decay factors ($D = 0.1$) yield worse results. In the first case, this performance drop is caused by flooding the network with too much energy, while in the second case the utilization of the semantic relations is too weak in order to provide enough information for the *K-Means* clustering algorithm. This is indicated by the total result for the parameter combination of **Comb=E, Norm=S** and $D = 0.1$, which is $V = 0.442$ at $V_{var} = 0.042$. This result is similar to this gained by applying *K-Means* to the binary *Semantic Patterns*, which yields $V = 0.443$ at $V_{var} = 0.058$ when spreading activation is deactivated. In both cases, the semantic relations are not or only weakly utilized during the *Semantic Patterns* generation. It seems that the robustness of the results is not so much influenced by the decay factors as it is by the combination function and link normalization parameters.

- **Influence of the decay factor - *EM***: Similar to the other parameters, the decay factor does not significantly influence the results gained by the *EM* algorithm.

Summary

In summary, the performance of the *K-Means* algorithm can be significantly improved by the *Semantic Patterns*. However, the *K-Means* performance stays below that of the *EM* algorithm, which is roughly the same for the raw data and the *Semantic Patterns*.

8.2.3 Mixed Data

Seven mixed data sets from the UCI Machine Learning Repository are used for this evaluation. These data sets contain distance-based and symbolic feature values. Due to the availability of distance-based features, the pre-spreading related parameters need to be evaluated.

	K-Means											EM									
Par	**Total**	**BC**	**DE**	**KR**	**LY**	**MU**	**SO**	**SP**	**VO**	**ZO**		**Total**	**BC**	**DE**	**KR**	**LY**	**MU**	**SO**	**SP**	**VO**	**ZO**
	Raw Data																				
N	**0.341**	**0.012**	**0.584**	0.004	0.131	**0.475**	0.587	**0.031**	0.467	0.782		Not available									
NN	0.296	0.007	0.094	**0.010**	**0.176**	0.472	**0.616**	0.030	**0.476**	**0.783**		**0.477**	**0.002**	**0.871**	**0.036**	**0.258**	**0.610**	**0.789**	**0.410**	**0.494**	**0.822**
	Semantic Patterns																				
D 0.0	**0.443**	**0.025**	**0.849**	**0.003**	**0.199**	**0.413**	**0.728**	**0.465**	**0.493**	**0.814**		**0.449**	**0.004**	**0.767**	**0.001**	**0.222**	**0.590**	**0.740**	**0.423**	**0.489**	**0.801**
	Comb=E Norm=L																				
D 0.1	0.442	0.029	0.811	**0.004**	**0.245**	**0.545**	0.726	0.387	**0.476**	0.759		0.441	0.074	**0.885**	0.000	**0.271**	**0.615**	0.786	0.004	**0.505**	0.826
D 0.3	0.447	0.068	0.846	**0.004**	0.241	0.482	0.724	0.424	**0.476**	0.758		0.460	**0.079**	0.875	**0.001**	0.258	0.592	0.788	0.250	0.449	0.846
D 0.5	**0.452**	0.061	**0.856**	0.000	**0.245**	0.448	**0.733**	**0.437**	0.467	**0.820**		**0.468**	**0.079**	0.874	**0.001**	0.265	0.592	0.789	0.306	0.452	**0.850**
D 0.7	0.422	**0.069**	0.826	0.000	0.209	0.275	0.728	0.419	0.463	0.804		0.465	**0.079**	0.874	**0.001**	0.252	0.579	**0.799**	**0.312**	0.445	0.847
	Comb=S Norm=L																				
D 0.1	**0.441**	0.056	**0.853**	**0.000**	**0.244**	**0.453**	0.733	0.399	**0.476**	0.759		0.433	0.079	0.872	**0.001**	0.270	0.572	0.794	0.001	**0.476**	0.829
D 0.3	0.434	**0.075**	0.820	**0.000**	0.228	0.411	0.718	**0.431**	0.472	0.750		**0.466**	0.079	**0.881**	**0.001**	**0.280**	**0.592**	0.802	0.298	0.437	0.828
D 0.5	0.439	0.060	0.792	**0.000**	0.235	0.416	**0.741**	0.405	0.463	**0.836**		**0.466**	0.079	0.871	**0.001**	0.251	0.581	**0.805**	**0.310**	0.445	**0.848**
D 0.7	0.422	0.067	0.798	**0.000**	0.224	0.364	0.726	0.376	0.462	0.782		0.462	**0.087**	0.875	**0.001**	0.254	0.580	0.776	0.292	0.445	0.845
	Comb=E Norm=S																				
D 0.1	0.418	0.029	0.790	0.006	**0.236**	0.311	0.705	0.449	**0.496**	0.742		0.472	**0.002**	0.893	**0.000**	0.263	0.571	0.767	**0.432**	**0.495**	0.820
D 0.3	**0.452**	0.030	**0.860**	0.001	0.231	0.470	0.715	**0.475**	0.491	**0.799**		**0.476**	**0.002**	**0.914**	**0.000**	0.261	**0.586**	**0.775**	0.427	**0.495**	0.823
D 0.5	0.448	**0.048**	0.799	**0.009**	0.215	**0.539**	**0.725**	0.450	0.493	0.758		0.472	**0.002**	0.897	**0.000**	**0.267**	0.584	0.758	0.427	0.484	**0.829**
D 0.7	0.448	0.033	0.850	0.000	0.230	0.495	0.712	0.435	0.493	0.787		0.473	**0.002**	0.903	**0.000**	0.250	**0.586**	0.773	0.427	0.484	**0.829**
	Comb=S Norm=S																				
D 0.1	**0.439**	0.029	0.806	**0.009**	**0.250**	0.435	**0.727**	0.439	**0.494**	0.760		0.475	**0.002**	0.903	**0.000**	0.254	0.576	0.764	**0.429**	**0.495**	0.852
D 0.3	0.420	0.015	0.775	0.004	0.210	**0.436**	0.717	0.409	0.443	0.774		0.474	**0.002**	0.901	**0.000**	**0.271**	0.584	0.763	0.427	0.484	0.837
D 0.5	0.429	0.030	0.789	**0.009**	0.226	0.410	0.716	**0.448**	0.485	0.749		0.476	**0.002**	0.904	**0.000**	0.255	0.586	0.767	0.427	0.484	**0.854**
D 0.7	0.438	**0.040**	**0.839**	0.006	0.246	0.418	0.726	0.409	0.480	**0.775**		**0.480**	**0.002**	**0.910**	**0.000**	0.269	**0.615**	**0.771**	0.431	0.494	0.825

Table 8.4: Unsupervised V-Measure results for the raw categorical data transformed into *Semantic Patterns*.

	K-Means											EM									
Par	**Total**	**BC**	**DE**	**KR**	**LY**	**MU**	**SO**	**SP**	**VO**	**ZO**		**Total**	**BC**	**DE**	**KR**	**LY**	**MU**	**SO**	**SP**	**VO**	**ZO**
	Raw Data																				
N	0.051	**0.015**	0.124	**0.005**	0.054	**0.140**	**0.026**	**0.015**	0.031	**sO**		Not available									
NN	**0.050**	0.025	**0.011**	0.022	**0.022**	0.228	0.035	0.018	**0.029**	0.058		**0.019**	**0.000**	**0.031**	**0.018**	**0.059**	**0.000**	**0.024**	**0.000**	**0.000**	**0.042**
	Semantic Patterns																				
D 0.0	**0.058**	**0.035**	**0.059**	**0.006**	**0.041**	**0.245**	**0.019**	**0.079**	**0.005**	**0.033**		**0.023**	**0.000**	**0.053**	**0.000**	**0.050**	**0.000**	**0.020**	**0.033**	**0.000**	**0.052**
	Comb=E Norm=L																				
D 0.1	**0.042**	**0.041**	0.079	0.010	0.044	**0.081**	0.027	0.025	**0.000**	0.073		0.010	**0.000**	0.022	**0.000**	**0.019**	**0.000**	**0.010**	0.002	**0.000**	0.034
D 0.3	0.045	0.043	0.080	0.010	0.035	0.135	**0.020**	0.004	**0.000**	0.080		**0.006**	**0.000**	**0.001**	**0.000**	0.020	**0.000**	0.014	**0.000**	**0.000**	**0.021**
D 0.5	0.044	0.039	**0.049**	**0.000**	0.033	0.190	0.021	**0.001**	**0.000**	0.060		0.009	**0.000**	0.002	**0.000**	0.023	**0.000**	0.021	**0.000**	**0.000**	0.037
D 0.7	0.044	**0.041**	0.076	**0.000**	**0.023**	0.175	0.025	0.016	0.004	**0.039**		0.010	**0.000**	0.015	**0.000**	0.023	**0.000**	0.014	**0.000**	**0.000**	0.042
	Comb=S Norm=L																				
D 0.1	0.050	0.044	**0.057**	**0.000**	0.039	**0.193**	0.019	0.014	**0.000**	0.082		**0.007**	**0.000**	**0.005**	**0.000**	**0.009**	**0.000**	0.015	**0.000**	**0.000**	0.036
D 0.3	**0.048**	**0.034**	0.079	**0.000**	**0.031**	0.202	0.022	**0.000**	0.007	0.058		0.009	**0.000**	0.018	**0.000**	0.025	**0.000**	0.020	**0.000**	**0.000**	**0.021**
D 0.5	**0.048**	0.043	0.087	**0.000**	**0.031**	0.196	0.021	0.008	0.004	**0.042**		0.010	**0.000**	0.006	**0.000**	0.020	**0.000**	0.024	**0.000**	**0.000**	0.039
D 0.7	0.051	0.040	0.096	**0.000**	0.032	0.219	**0.017**	0.007	0.004	0.046		0.013	**0.000**	0.006	**0.000**	0.017	**0.000**	**0.010**	0.056	**0.000**	0.028
	Comb=E Norm=S																				
D 0.1	0.057	**0.038**	0.077	0.012	0.032	0.227	**0.019**	**0.038**	0.005	0.064		0.012	**0.000**	0.031	**0.000**	0.025	**0.000**	**0.013**	0.001	**0.000**	0.038
D 0.3	0.055	0.039	0.062	0.002	0.032	0.206	0.031	0.080	0.006	**0.037**		**0.010**	**0.000**	**0.016**	**0.000**	0.030	**0.000**	0.014	0.001	**0.000**	**0.032**
D 0.5	0.052	0.046	0.071	0.014	0.061	**0.101**	0.032	0.093	**0.004**	0.050		0.012	**0.000**	0.021	**0.000**	**0.022**	**0.000**	0.019	**0.000**	**0.000**	0.047
D 0.7	**0.045**	0.039	**0.043**	**0.000**	**0.029**	0.166	0.022	0.054	**0.004**	0.049		0.012	**0.000**	0.023	**0.000**	0.032	**0.000**	0.014	**0.000**	**0.000**	0.040
	Comb=S Norm=S																				
D 0.1	**0.055**	0.041	0.088	0.012	0.031	**0.181**	0.030	0.059	0.006	0.044		**0.010**	**0.000**	**0.021**	**0.000**	0.024	**0.000**	0.020	0.001	**0.000**	**0.022**
D 0.3	0.073	**0.030**	**0.055**	**0.008**	0.051	0.214	0.031	0.060	0.146	0.059		**0.010**	**0.000**	0.025	**0.000**	0.024	**0.000**	**0.009**	**0.000**	**0.000**	0.031
D 0.5	0.059	0.043	0.086	0.014	**0.019**	0.211	0.032	**0.041**	**0.000**	0.084		0.012	**0.000**	0.024	**0.000**	0.030	**0.000**	0.023	**0.000**	**0.000**	0.031
D 0.7	0.062	0.047	0.068	0.012	0.027	0.235	**0.025**	0.101	0.005	**0.041**		**0.010**	**0.000**	0.023	**0.000**	**0.012**	**0.000**	0.016	0.008	**0.000**	0.028

Table 8.5: Unsupervised results (V-Measure standard deviation values): *Semantic Patterns* for the categorical data sets

Baseline

For the baseline *Semantic Patterns*, which are generated for the mixed data sets, the pre-spreading related parameters C, and σ_p are required. The following parameters are used for the generation of the patterns: For the baseline *Semantic Patterns*, which are generated for the mixed data sets, the following parameters are chosen:

- **Decay factors**: $D = \{0.0\}$. For the baseline configuration $D = 0.0$ is used, which completely deactivates the spreading activation algorithm and makes the combination function (**Comb**) and link normalization parameters (**Norm**) irrelevant.

- **Pre-spreading:** Although the main spreading process is deactivated for the baselines patterns by choosing $D = 0.0$, there is still the pre-spreading process for distance-based features that is applied prior to the main spreading activation process. Therefore, the parameters for the model complexity C and the parameter σ_p that determines the influence on the neighboring nodes, must be specified. For this evaluation, the different combinations for $C = \{1.0, 1.5, 2.0, 3.0\}$ and $\sigma_p = \{0.0, 0.2, 0.4, 0.6, 0.8\}$ are evaluated.

The results gained by applying the *K-Means* and *EM* algorithms to the baseline *Semantic Patterns* are listed in Tables 8.6 and 8.7. Thereby, the following observations are made:

- **Raw data**: Similar to the results for the raw categorical data, the *EM* algorithm significantly outperforms the *K-Means* algorithm. While the total result for the *EM* algorithm is $V = 0.165$ at $V_\sigma = 0.013$, it is only $V = 0.342$ at $V_\sigma = 0.069$ for the *K-Means* algorithm when the data is normed. This value drops completely when the *K-Means* algorithm is applied directly to the raw data without normalizing the data, where $V = 0.017$ and $V_\sigma = 0.003$. Again, the more sophisticated model employed by the *EM* algorithm might be the reason for this performance difference.

- **Baseline *Semantic Patterns - K-Means***: For the mixed data sets the parameters C and σ_p related to the distance-based features need to be evaluated. For the *K-Means* algorithm, one observes a significant increase in performance when using the baseline patterns. The performance drops when a too complex model $C = 3.0$ is used. This is related to the discretization complexity problem described in Section 7.3.3 of the previous chapter, which is also indicated by the slight increase in performance when using a large $\sigma_p = 0.8$ value that mitigates the negative effect of too complex models.

- **Baseline *Semantic Patterns - EM***: Again, the *EM* algorithm performs significantly better than the *K-Means* algorithm ($V = 0.201$ at $V_\sigma = 0.013$ vs. $V = 0.165$ at $V_\sigma = 0.069$). For the baseline *Semantic Patterns* the performance drops when the discretization model complexity increases.

Similar to the *K-Means* algorithm, one observes a slight mitigation of this problem with an increasing value for σ_p, but even then the performance is much worse when compared to the application to raw data.

	K-Means									EM							
Par	**Total**	**AN**	**CO**	**CA**	**CG**	**HC**	**HH**	**HE**		**Total**	**AN**	**CO**	**CA**	**CG**	**HC**	**HH**	**HE**
	Raw Data																
N	0.165	0.226	0.129	0.155	0.009	0.237	0.269	0.131		Not available							
NN	**0.017**	**0.028**	**0.030**	**0.016**	**0.012**	**0.014**	**0.012**	**0.004**		**0.201**	**0.312**	**0.103**	**0.171**	**0.013**	**0.309**	**0.278**	**0.223**
	Semantic Patterns																
	No Spreading/1.0																
σ 0.0	0.192	0.246	0.142	0.142	0.004	0.288	0.317	**0.202**		0.190	0.291	0.098	0.227	0.003	0.228	0.258	0.227
σ 0.2	0.197	**0.272**	0.143	**0.158**	**0.006**	0.285	0.317	0.195		0.182	0.280	0.098	0.162	0.003	0.244	0.258	**0.231**
σ 0.4	0.183	0.243	0.140	0.118	0.005	0.296	0.286	0.194		0.184	0.226	**0.099**	0.229	**0.004**	**0.245**	0.258	0.227
σ 0.6	**0.201**	0.297	**0.145**	**0.158**	0.003	0.287	0.318	0.196		**0.194**	0.291	0.097	**0.240**	0.003	0.217	**0.281**	0.229
σ 0.8	0.194	0.242	0.142	**0.158**	**0.006**	**0.293**	**0.319**	0.197		0.192	**0.293**	0.097	0.232	**0.004**	0.228	0.258	0.230
	No Spreading/1.5																
σ 0.0	0.173	0.223	0.132	0.129	0.005	0.279	0.246	0.198		**0.158**	**0.206**	0.118	0.230	**0.003**	0.136	0.210	0.204
σ 0.2	0.199	**0.272**	0.152	**0.158**	0.006	0.331	0.272	**0.203**		0.157	0.205	0.109	**0.231**	0.001	0.136	**0.225**	0.196
σ 0.4	0.195	0.247	0.148	0.129	0.007	**0.337**	0.295	0.198		0.157	**0.206**	0.116	0.221	**0.003**	0.136	0.212	0.207
σ 0.6	**0.200**	0.255	**0.154**	0.131	**0.008**	0.334	**0.321**	0.198		0.160	0.199	**0.133**	0.219	0.002	0.136	0.220	**0.208**
σ 0.8	0.196	0.264	0.152	0.131	**0.008**	0.335	0.289	0.194		**0.158**	0.199	0.118	0.226	0.002	**0.138**	0.220	0.207
	No Spreading/2.0																
σ 0.0	0.193	0.253	0.135	0.113	0.007	0.356	0.293	0.195		0.167	0.219	0.108	0.209	0.003	**0.178**	0.217	**0.235**
σ 0.2	0.198	**0.271**	0.147	0.116	0.007	0.356	0.301	0.189		0.168	0.213	0.096	0.203	0.005	**0.178**	0.243	**0.235**
σ 0.4	**0.204**	0.240	**0.157**	**0.145**	**0.009**	0.356	**0.327**	0.194		**0.174**	**0.223**	**0.131**	**0.231**	0.002	**0.178**	0.214	**0.235**
σ 0.6	0.194	0.221	0.154	**0.145**	0.008	**0.359**	0.275	0.196		0.168	0.220	0.109	0.171	0.004	**0.178**	**0.261**	**0.235**
σ 0.8	0.200	0.258	0.152	0.098	0.007	0.358	**0.327**	**0.197**		0.165	0.217	0.103	0.179	**0.007**	0.175	0.238	**0.235**
	No Spreading/3.0																
σ 0.0	0.189	0.238	**0.155**	**0.145**	0.007	0.332	0.262	0.183		0.159	0.197	0.108	0.215	0.001	0.149	0.235	0.205
σ 0.2	**0.190**	**0.285**	0.148	0.100	**0.008**	**0.340**	0.277	0.171		0.160	0.214	0.112	0.230	0.001	0.139	0.211	**0.216**
σ 0.4	0.181	0.219	0.150	0.086	0.007	0.338	0.281	0.184		0.163	0.201	0.107	0.214	0.001	**0.158**	0.247	0.213
σ 0.6	0.179	0.227	0.152	0.131	0.006	0.316	0.236	0.186		0.169	**0.227**	**0.120**	**0.232**	**0.002**	0.141	0.252	0.207
σ 0.8	0.191	0.246	0.148	0.118	0.007	0.304	**0.328**	**0.188**		**0.167**	0.215	0.117	0.206	**0.002**	0.157	**0.254**	**0.216**

Table 8.6: Unsupervised V-Measure results for the raw mixed data sets and the respective baseline *Semantic Patterns*.

Semantic Patterns

In this evaluation, the *Semantic Pattern Transformation* is applied to the raw mixed data sets for the generation of the associated *Semantic Patterns*. Here, the main focus is placed on the influence of the spreading activation parameters for the distance-base features C and σ_p. Thereby, the following parameters are used:

- **Decay factors**: $D = \{0.1, 0.3, 0.5, 0.7\}$. Together with the possible values for the combination function (**Comb**) and the link normalization strategy (**Norm**), the number of possible combinations is quite huge and the results in the tables are limited to those decay values that yield the best results for the *K-Means* and *EM* algorithm. For the *K-Means* algorithm that is achieved with $D = 0.5$, and for the *EM* algorithm $D = 0.7$ yields the best results.

	K-Means									EM							
Par	**Total**	**AN**	**CO**	**CA**	**CG**	**HC**	**HH**	**HE**		**Total**	**AN**	**CO**	**CA**	**CG**	**HC**	**HH**	**HE**
	Raw Data																
N	0.069	0.054	0.071	0.093	0.007	0.109	0.097	0.051		Not available							
NN	**0.003**	**0.007**	**0.010**	**0.000**	**0.000**	**0.000**	**0.001**	**0.000**		**0.013**	**0.057**	**0.000**	**0.032**	**0.002**	**0.000**	**0.000**	**0.000**
	Semantic Patterns																
	No Spreading/1.0																
σ 0.0	0.022	0.075	**0.007**	0.047	**0.003**	0.007	0.004	0.012		0.012	0.047	**0.003**	**0.000**	**0.000**	0.027	**0.000**	0.007
σ 0.2	**0.012**	0.058	**0.007**	**0.000**	**0.003**	**0.004**	0.004	**0.009**		0.015	0.070	**0.003**	**0.000**	0.001	0.025	**0.000**	**0.005**
σ 0.4	0.035	0.061	**0.007**	0.062	**0.003**	0.009	0.094	**0.009**		0.017	0.085	**0.003**	**0.000**	0.001	0.026	**0.000**	0.007
σ 0.6	**0.012**	**0.056**	0.008	**0.000**	**0.003**	**0.004**	0.003	**0.009**		**0.011**	0.049	**0.003**	**0.000**	**0.000**	**0.022**	**0.000**	0.006
σ 0.8	0.013	0.063	**0.007**	**0.000**	**0.003**	0.010	**0.002**	**0.009**		0.012	**0.044**	**0.003**	**0.000**	0.001	0.027	**0.000**	0.006
	No Spreading/1.5																
σ 0.0	0.058	0.067	0.044	0.059	0.004	0.108	0.113	0.011		0.008	0.028	0.011	**0.000**	0.003	**0.002**	**0.000**	0.015
σ 0.2	0.027	0.061	0.009	**0.000**	0.004	0.012	0.097	**0.008**		0.009	0.033	0.017	**0.000**	**0.000**	**0.002**	**0.000**	0.014
σ 0.4	0.035	0.076	0.009	0.059	0.003	0.012	0.076	**0.008**		0.008	0.031	**0.009**	**0.000**	0.001	**0.002**	**0.000**	**0.013**
σ 0.6	**0.019**	**0.046**	**0.008**	0.054	**0.002**	0.013	**0.003**	**0.008**		0.008	0.019	0.018	**0.000**	0.001	**0.002**	**0.000**	**0.013**
σ 0.8	0.037	0.076	0.010	0.054	0.004	**0.011**	0.096	**0.008**		**0.006**	**0.015**	0.012	**0.000**	0.001	**0.002**	**0.000**	0.014
	No Spreading/2.0																
σ 0.0	0.043	0.072	0.047	0.069	0.004	**0.005**	0.098	0.008		0.009	0.044	0.012	**0.000**	0.005	**0.000**	**0.000**	0.001
σ 0.2	0.031	**0.045**	**0.010**	0.065	0.004	0.006	0.078	0.009		0.006	0.027	**0.008**	**0.000**	0.007	**0.000**	**0.000**	0.001
σ 0.4	**0.020**	0.066	0.017	**0.040**	**0.003**	**0.005**	**0.002**	0.007		0.008	**0.020**	0.032	**0.000**	**0.004**	**0.000**	**0.000**	**0.000**
σ 0.6	0.035	0.066	0.014	**0.040**	0.004	0.010	0.104	**0.006**		**0.005**	0.021	**0.008**	**0.000**	0.005	**0.000**	**0.000**	0.001
σ 0.8	0.024	0.056	0.014	0.075	**0.003**	0.010	**0.002**	0.009		0.010	0.035	0.018	**0.000**	0.007	0.010	**0.000**	0.001
	No Spreading/3.0																
σ 0.0	0.039	0.067	**0.007**	**0.040**	0.003	**0.015**	0.131	**0.010**		0.012	0.044	0.013	**0.000**	**0.001**	0.016	**0.000**	**0.012**
σ 0.2	0.044	0.068	**0.007**	0.072	**0.001**	**0.015**	0.100	0.044		**0.010**	**0.032**	0.011	**0.000**	**0.001**	**0.015**	**0.000**	0.014
σ 0.4	0.038	0.051	0.010	0.072	0.003	0.018	0.097	0.014		0.012	0.043	0.008	**0.000**	0.002	0.017	**0.000**	0.015
σ 0.6	0.053	0.056	0.008	0.054	0.003	0.098	0.141	0.012		0.011	0.034	0.013	**0.000**	0.002	**0.015**	**0.000**	0.014
σ 0.8	**0.033**	**0.048**	0.010	0.062	0.002	0.096	**0.004**	0.011		**0.010**	0.033	**0.007**	**0.000**	0.002	0.016	**0.000**	0.014

Table 8.7: Unsupervised V-Measure standard deviation results for the raw mixed data sets and the respective baseline *Semantic Patterns*.

- **Link normalization (Norm) and combination function (Comb)**: Here, a similar approach as for the decay factors is used: Only the best combination of the link normalization strategy and the combination function are shown in the results tables: **Comb=E** and **Norm=L**.
- **Pre-spreading parameters**: The focus of the results tables is placed on the pre-spreading parameters C and σ_p. Here, the same combinations as for the baseline *Semantic Patterns* are employed: $C = \{1.0, 1.5, 2.0, 3.0\}$ and $\sigma_p = \{0.0, 0.2, 0.4, 0.6, 0.8\}$.

The results gained by applying the *K-Means* and *EM* algorithms to the *Semantic Patterns* are listed in Tables 8.8 and 8.9. Thereby, the following observations are made:

- ***K-Means***: Similar to the evaluation of the categorical data sets, the performance of *K-Means* significantly increases when applied to the *Semantic Patterns*. The best results achieved are $V = 0.221$ at $V_\sigma = 0.009$ compared to $V = 0.204$ at $V_\sigma = 0.020$ of the baseline *Semantic Patterns*, and $V = 0.165$ at $V_\sigma = 0.069$ of the raw data. Also, the stability of the results increases, which is indicated by the significantly lower standard deviation values.

- ***EM***: For the *EM* algorithm, the performance increases ($V = 0.211$ at $V_\sigma = 0.001$) when compared to the raw data ($V = 0.201$ at $V_\sigma = 0.013$), and the baseline *Semantic Patterns* ($V = 0.194$ at $V_\sigma = 0.011$). Similar to *K-Means*, the stability of the results – gained by applying *EM* to the *Semantic Patterns* – significantly increases.

- ***K-Means* vs. *EM***: In contrast to the categorical data, the best result achieved by the *EM* algorithm ($V = 0.211$ at $V_\sigma = 0.001$) is lower than that achieved by *K-Means* ($V = 0.221$ at $V_\sigma = 0.009$). Since this is also observed for the numerical data, there is an indication that the *EM* algorithm slightly looses its advantages when applied to distance-based features and stays below the performance that is achieved by *K-Means* when applied to the *Semantic Patterns*.

- **Influence of the MDL override factor (C) - *K-Means***: The best results are achieved when the factor $C = 2.0$ is employed, which means that the discretization models used for the distance-based features are twice as complex as the models found by using the MDL criterion ($C = 1.0$). The overall performance increases from $C = 1.0$ over $C = 1.5$ to $C = 2.0$ and drops again for $C = 3.0$. These results indicate – as postulated in 7.3.3 of the previous chapter – that a too low or too high model complexity has a negative influence on the performance.

- **Influence of the MDL override factor (C) - *EM***: In contrast to the *K-Means* algorithm, the *EM* algorithm does not benefit from complex discretization models. Here, the best results are achieved by choosing *MDL*=1.0, which corresponds to the model found when using the standard MDL criterion.

- **Influence of the pre-spreading factor σ_p - *K-Means***: In 7.3.3 of the previous chapter it was also postulated that the effect of too complex models can be compensated by introducing larger values for σ_p. This effect can be observed when looking at the σ_p values that achieve the best result for the different C values: While for lower values of C, the best results are achieved with low σ_p values, for higher values of C the best results are achieved with larger σ_p values, which increase the influence onto neighboring nodes within the discretization models.

- **Influence of the pre-spreading factor σ_p - *EM***: In contrast to *K-Means*, there is no significant influence of the pre-spreading factor σ_p.

Summary

In summary, the performance of the *K-Means* algorithm can be significantly improved by the *Semantic Patterns*. In contrast to the categorical data, the results for the *K-Means* algorithm, when applied to the *Semantic Patterns*, are even slightly better than those of the *EM* algorithm.

	K-Means								EM							
Par	**Total**	**AN**	**CO**	**CA**	**CG**	**HC**	**HH**	**HE**	**Total**	**AN**	**CO**	**CA**	**CG**	**HC**	**HH**	**HE**
	Raw Data															
N	**0.165**	**0.226**	**0.129**	**0.155**	0.009	**0.237**	**0.269**	**0.131**	Not available							
NN	0.017	0.028	0.030	0.016	**0.012**	0.014	0.012	0.004	**0.201**	**0.312**	**0.103**	**0.171**	**0.013**	**0.309**	**0.278**	**0.223**
	Semantic Patterns															
	D=0.0 MDL=2.0								**D=0.0 MDL=1.0**							
σ 0.0	0.193	0.253	0.135	0.113	0.007	0.356	0.293	0.195	0.190	0.291	0.098	0.227	0.003	0.228	0.258	0.227
σ 0.2	0.198	**0.271**	0.147	0.116	0.007	0.356	0.301	0.189	0.182	0.280	0.098	0.162	0.003	0.244	0.258	**0.231**
σ 0.4	**0.204**	0.240	**0.157**	**0.145**	**0.009**	0.356	**0.327**	0.194	0.184	0.226	**0.099**	0.229	**0.004**	**0.245**	0.258	0.227
σ 0.6	0.194	0.221	0.154	**0.145**	0.008	**0.359**	0.275	0.196	**0.194**	0.291	0.097	**0.240**	0.003	0.217	**0.281**	0.229
σ 0.8	0.200	0.258	0.152	0.098	0.007	0.358	**0.327**	**0.197**	0.192	**0.293**	0.097	0.232	**0.004**	0.228	0.258	0.230
	D=0.5 MDL=1.0								**D=0.7 MDL=1.0**							
σ 0.0	**0.211**	**0.320**	**0.042**	0.262	**0.001**	0.325	**0.311**	0.215	0.210	**0.327**	0.127	**0.218**	**0.021**	**0.237**	0.311	**0.229**
σ 0.2	0.201	0.257	0.032	0.262	**0.001**	0.323	**0.311**	**0.222**	0.210	0.322	0.126	**0.218**	**0.021**	**0.237**	**0.320**	**0.229**
σ 0.4	0.208	0.299	0.035	0.261	**0.001**	**0.326**	**0.311**	0.220	**0.211**	0.322	0.127	**0.218**	**0.021**	**0.237**	**0.320**	**0.229**
σ 0.6	0.204	0.281	0.029	0.262	**0.001**	0.325	**0.311**	0.220	**0.211**	0.321	**0.128**	**0.218**	**0.021**	**0.237**	**0.320**	**0.229**
σ 0.8	0.207	0.292	0.041	**0.263**	**0.001**	**0.326**	**0.311**	0.216	0.209	0.310	0.127	**0.218**	**0.021**	**0.237**	**0.320**	**0.229**
	D=0.5 MDL=1.5								**D=0.7 MDL=1.5**							
σ 0.0	**0.216**	**0.317**	0.065	**0.249**	**0.001**	**0.357**	**0.320**	0.203	0.204	0.322	0.123	**0.212**	**0.016**	**0.275**	0.247	0.233
σ 0.2	0.211	0.295	0.052	0.247	0.000	0.355	**0.320**	**0.209**	0.204	0.322	0.123	**0.212**	**0.016**	**0.275**	0.247	0.236
σ 0.4	0.216	0.314	**0.074**	0.248	**0.001**	**0.357**	**0.320**	0.198	**0.205**	**0.323**	0.123	0.206	**0.016**	**0.275**	**0.252**	**0.237**
σ 0.6	0.212	0.308	0.046	**0.249**	**0.001**	0.356	**0.320**	**0.209**	0.204	0.320	**0.125**	0.208	**0.016**	**0.275**	0.246	0.236
σ 0.8	0.211	0.293	0.063	0.248	0.000	0.354	**0.320**	0.201	0.204	**0.323**	**0.125**	0.208	**0.016**	**0.275**	0.249	0.232
	D=0.5 MDL=2.0								**D=0.7 MDL=2.0**							
σ 0.0	0.217	0.304	0.048	**0.244**	**0.000**	**0.390**	**0.311**	**0.219**	0.206	**0.319**	0.117	0.229	**0.010**	**0.255**	**0.277**	**0.233**
σ 0.2	0.218	**0.313**	0.062	**0.244**	**0.000**	0.388	**0.311**	0.208	**0.207**	0.317	0.126	0.239	**0.010**	**0.255**	0.268	**0.233**
σ 0.4	**0.221**	0.309	**0.084**	0.243	**0.000**	0.389	**0.311**	0.209	0.205	**0.319**	**0.127**	0.224	**0.010**	**0.255**	0.268	**0.233**
σ 0.6	0.213	0.285	0.057	0.243	**0.000**	0.387	**0.311**	0.210	0.206	0.307	**0.127**	**0.240**	**0.010**	**0.255**	0.268	**0.233**
σ 0.8	0.211	0.295	0.036	0.244	**0.000**	0.387	**0.311**	0.205	0.204	0.305	**0.127**	**0.240**	**0.010**	**0.255**	0.259	**0.233**
	D=0.5 MDL=3.0								**D=0.7 MDL=3.0**							
σ 0.0	0.203	0.294	0.030	**0.248**	0.000	0.335	**0.315**	**0.196**	0.192	**0.323**	0.108	**0.248**	0.009	**0.201**	0.250	**0.205**
σ 0.2	0.208	0.306	0.059	**0.248**	0.000	0.334	**0.315**	0.193	0.190	0.321	0.107	0.237	0.009	**0.201**	**0.251**	**0.205**
σ 0.4	0.205	0.310	0.050	**0.248**	0.000	0.334	**0.315**	0.178	**0.193**	0.322	**0.122**	0.243	0.009	**0.201**	0.249	**0.205**
σ 0.6	0.207	0.300	**0.063**	**0.248**	**0.001**	0.333	0.313	0.192	0.192	0.321	**0.122**	0.243	**0.010**	**0.201**	0.245	**0.205**
σ 0.8	**0.210**	**0.330**	0.050	0.246	**0.001**	**0.336**	**0.315**	0.191	0.192	**0.323**	**0.122**	0.243	0.009	**0.201**	0.240	**0.205**

Table 8.8: Unsupervised V-Measure results for the raw mixed data transformed into *Semantic Patterns*.

8.2.4 Numerical Data

Ten numerical data sets from the UCI Machine Learning Repository are used for this evaluation. These data sets contain only distance-based feature values. Thus, the pre-spreading parameters play a significant role within the evaluation process.

Baseline

For the generation of the baseline *Semantic Patterns* the same parameters as for the mixed data sets are used. Thereby, the results gained by applying the *K-Means* and *EM* algorithms to the baseline *Semantic Patterns*, are listed in Tables 8.10 and 8.11, and the following observations are made:

- **Raw data**: The result for the *EM* algorithm ($V = 0.346$ at $V_\sigma = 0.005$)

	K-Means									EM							
Par	**Total**	**AN**	**CO**	**CA**	**CG**	**HC**	**HH**	**HE**		**Total**	**AN**	**CO**	**CA**	**CG**	**HC**	**HH**	**HE**
	Raw Data																
N	0.069	0.054	0.071	0.093	0.007	0.109	0.097	0.051		Not available							
NN	**0.003**	**0.007**	**0.010**	**0.000**	**0.000**	**0.000**	**0.001**	**0.000**		**0.013**	**0.057**	**0.000**	**0.032**	**0.002**	**0.000**	**0.000**	**0.000**
	Semantic Patterns																
	D=0.0 MDL=2.0									**D=0.0 MDL=1.0**							
σ 0.0	0.043	0.072	0.047	0.069	0.004	**0.005**	0.098	0.008		0.012	0.047	**0.003**	**0.000**	**0.000**	0.027	**0.000**	0.007
σ 0.2	0.031	**0.045**	**0.010**	0.065	0.004	0.006	0.078	0.009		0.015	0.070	**0.003**	**0.000**	0.001	0.025	**0.000**	**0.005**
σ 0.4	**0.020**	0.066	0.017	**0.040**	**0.003**	**0.005**	**0.002**	0.007		0.017	0.085	**0.003**	**0.000**	0.001	0.026	**0.000**	0.007
σ 0.6	0.035	0.066	0.014	**0.040**	0.004	0.010	0.104	**0.006**		**0.011**	0.049	**0.003**	**0.000**	**0.000**	**0.022**	**0.000**	0.006
σ 0.8	0.024	0.056	0.014	0.075	**0.003**	0.010	**0.002**	0.009		0.012	**0.044**	**0.003**	**0.000**	0.001	0.027	**0.000**	0.006
	D=0.5 MDL=1.0									**D=0.7 MDL=1.0**							
σ 0.0	**0.011**	**0.016**	0.049	0.001	**0.000**	0.003	**0.000**	0.008		0.002	0.004	0.005	**0.000**	**0.000**	0.006	**0.000**	**0.000**
σ 0.2	0.017	0.079	**0.037**	**0.000**	**0.000**	0.003	**0.000**	**0.001**		**0.001**	0.008	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**
σ 0.4	0.013	0.051	0.039	0.001	**0.000**	**0.001**	**0.000**	0.002		**0.001**	**0.000**	0.005	**0.000**	**0.000**	0.001	**0.000**	**0.000**
σ 0.6	0.015	0.055	0.039	0.001	**0.000**	0.004	**0.000**	0.004		0.002	0.006	**0.000**	**0.000**	**0.000**	0.006	**0.000**	**0.000**
σ 0.8	0.013	0.036	0.044	0.001	**0.000**	0.002	**0.000**	0.010		**0.001**	0.006	**0.000**	**0.000**	**0.000**	0.001	**0.000**	**0.000**
	D=0.5 MDL=1.5									**D=0.7 MDL=1.5**							
σ 0.0	0.011	0.019	0.039	0.004	**0.000**	**0.004**	**0.000**	0.011		0.006	0.044	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**
σ 0.2	0.014	0.050	**0.030**	**0.003**	**0.000**	**0.004**	**0.000**	0.013		0.006	0.040	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**
σ 0.4	**0.009**	**0.017**	0.036	0.004	**0.000**	**0.004**	**0.000**	**0.000**		**0.000**	**0.001**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**
σ 0.6	0.013	0.038	0.032	0.004	**0.000**	**0.004**	**0.000**	0.013		0.001	0.005	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**
σ 0.8	0.014	0.045	0.040	0.004	**0.000**	**0.004**	**0.000**	0.008		0.001	0.006	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**
	D=0.5 MDL=2.0									**D=0.7 MDL=2.0**							
σ 0.0	0.012	0.030	0.040	0.002	**0.000**	**0.003**	**0.000**	**0.009**		**0.001**	**0.005**	0.001	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**
σ 0.2	0.012	0.015	0.050	0.002	**0.000**	0.004	**0.000**	0.014		0.002	0.007	**0.000**	**0.000**	**0.000**	0.006	**0.000**	**0.000**
σ 0.4	**0.009**	**0.014**	**0.027**	0.002	**0.000**	0.004	**0.000**	0.014		0.003	0.006	0.011	**0.000**	**0.000**	0.006	**0.000**	**0.000**
σ 0.6	0.015	0.049	0.036	0.002	**0.000**	0.004	**0.000**	0.014		0.003	0.006	0.007	**0.000**	**0.000**	0.006	**0.000**	**0.000**
σ 0.8	0.015	0.047	0.038	**0.001**	**0.000**	0.004	**0.000**	0.014		0.002	**0.005**	0.006	**0.000**	**0.000**	0.006	**0.000**	**0.000**
	D=0.5 MDL=3.0									**D=0.7 MDL=3.0**							
σ 0.0	0.014	0.047	0.032	0.002	**0.000**	**0.003**	0.004	0.011		0.002	0.005	0.006	**0.000**	0.002	**0.000**	**0.000**	**0.000**
σ 0.2	0.014	0.040	0.040	0.002	**0.000**	0.004	0.004	0.010		0.002	0.005	0.008	**0.000**	0.002	**0.000**	**0.000**	**0.000**
σ 0.4	0.019	0.033	**0.030**	0.002	**0.000**	0.004	0.004	0.058		0.002	0.006	0.008	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**
σ 0.6	0.015	0.044	0.044	**0.001**	**0.000**	**0.003**	**0.003**	**0.009**		**0.001**	**0.004**	**0.004**	**0.000**	0.002	**0.000**	**0.000**	**0.000**
σ 0.8	**0.012**	**0.020**	0.044	0.002	**0.000**	**0.003**	0.004	0.012		0.002	0.005	0.007	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**

Table 8.9: Unsupervised V-Measure standard deviation results for the raw mixed data transformed into *Semantic Patterns*.

significantly outperforms the results for the *K-Means* algorithm, which are at $V = 0.307$ at $V_\sigma = 0.021$ for the normed raw data and $V = 0.299$ at $V_\sigma = 0.025$ for the unprocessed raw data. This observation is similar to that gained for the categorical and mixed data sets. In contrast to the mixed data sets, where the performance of the *K-Means* algorithm completely drops when applied to the unprocessed (not normed) raw data, only an insignificant difference can be observed here.

- **Baseline *Semantic Patterns - K-Means***: The results gained by applying *K-Means* to the baseline *Semantic Patterns* are in general better than those for the raw data. Similar to the results of the mixed data sets, one observes a performance drop when a too complex model is used. Here, the performance of the model for $C = 3.0$ ($V = 0.289$ at $V_\sigma = 0.028$) even drops below the results, which are gained when applying *K-Means* to the

raw data.

- **Baseline *Semantic Patterns - EM***: For the baseline *Semantic Patterns*, the performance of the *EM* algorithm drops when the discretization model complexity increases. Similar to the *K-Means* algorithm, one observes a slight mitigation of this problem with an increasing value for σ_p, but even then the performance is much worse than that gained for the raw data.

	K-Means												EM										
Par	**Total**	**BW**	**DI**	**GL**	**HS**	**IO**	**IR**	**SE**	**SO**	**VE**	**VO**		**Total**	**BW**	**DI**	**GL**	**HS**	**IO**	**IR**	**SE**	**SO**	**VE**	**VO**
	Raw Data																						
N	0.299	0.734	**0.052**	0.335	**0.254**	0.121	**0.708**	**0.608**	0.006	0.113	0.057		Not available										
NN	**0.307**	**0.735**	0.030	**0.388**	0.019	**0.123**	0.705	0.529	**0.008**	**0.188**	**0.342**		**0.346**	**0.718**	**0.103**	**0.370**	**0.289**	**0.254**	**0.806**	**0.621**	**0.005**	**0.103**	**0.194**
	Semantic Patterns																						
	No Spreading/1.0																						
σ 0.0	**0.316**	0.722	0.032	**0.297**	0.315	0.113	**0.676**	0.597	0.011	0.156	0.244		0.317	**0.777**	0.006	0.312	0.239	0.218	0.651	0.592	0.016	0.174	0.186
σ 0.2	0.307	0.722	0.043	0.294	0.306	0.123	0.593	0.588	0.012	0.145	0.248		**0.327**	0.752	0.001	**0.318**	**0.240**	0.218	0.766	0.598	0.016	0.167	0.197
σ 0.4	0.309	**0.724**	0.039	0.275	0.311	0.125	0.563	0.619	**0.013**	**0.164**	**0.253**		0.323	0.727	**0.011**	0.287	0.229	0.217	0.749	0.600	0.018	0.176	**0.218**
σ 0.6	**0.316**	0.723	0.034	0.273	**0.323**	0.137	0.626	**0.633**	0.009	0.157	0.248		0.317	0.732	0.009	0.316	0.232	**0.221**	0.637	**0.606**	**0.025**	0.175	0.214
σ 0.8	0.315	0.719	**0.063**	0.273	0.316	**0.145**	0.633	0.590	0.012	0.144	0.248		0.325	0.703	0.006	0.305	0.233	0.216	**0.796**	0.594	0.019	**0.181**	0.195
	No Spreading/1.5																						
σ 0.0	0.315	**0.724**	**0.039**	0.329	0.309	0.045	0.717	0.582	**0.026**	0.198	0.183		**0.301**	**0.777**	0.001	0.317	**0.150**	**0.088**	0.721	0.591	0.011	0.175	**0.183**
σ 0.2	**0.323**	**0.724**	0.025	**0.334**	0.344	**0.071**	**0.730**	0.590	0.012	0.198	0.196		0.287	0.752	0.002	0.321	0.125	0.087	0.615	**0.609**	0.010	**0.187**	0.166
σ 0.4	0.318	0.719	0.026	0.285	0.316	0.051	0.769	0.600	0.008	**0.199**	**0.203**		0.294	0.727	0.004	**0.338**	0.125	0.086	**0.764**	0.605	0.012	0.106	0.169
σ 0.6	0.317	0.722	0.025	0.298	**0.357**	0.040	0.712	**0.602**	0.013	**0.199**	0.201		0.297	0.732	**0.005**	0.324	0.133	**0.088**	0.748	0.594	0.013	0.161	0.179
σ 0.8	0.299	0.646	0.015	0.294	0.328	0.026	0.686	0.581	0.014	0.198	0.200		0.283	0.704	0.002	0.326	0.131	0.086	0.642	0.584	**0.014**	0.161	0.175
	No Spreading/2.0																						
σ 0.0	0.296	**0.724**	**0.028**	0.291	0.297	0.055	0.664	0.573	0.011	0.162	0.154		**0.311**	**0.777**	0.002	0.302	0.189	0.052	**0.901**	**0.555**	0.005	**0.155**	**0.170**
σ 0.2	0.294	0.651	0.010	0.290	**0.328**	0.038	0.724	0.572	0.017	0.151	0.157		0.296	0.752	0.013	0.322	**0.190**	**0.059**	0.766	0.552	0.004	0.140	0.164
σ 0.4	0.310	0.723	**0.028**	0.304	0.326	0.042	**0.781**	0.574	0.016	0.160	0.142		0.298	0.727	0.017	0.298	**0.190**	0.056	0.859	0.545	**0.007**	0.118	0.160
σ 0.6	**0.312**	0.723	0.017	**0.325**	0.327	0.051	0.745	**0.589**	**0.029**	0.154	**0.160**		0.303	0.734	0.020	0.322	0.188	0.054	0.857	0.543	0.006	0.154	0.157
σ 0.8	0.305	0.719	0.010	0.297	**0.328**	**0.066**	0.753	0.555	0.010	**0.164**	0.150		0.294	0.704	**0.021**	**0.327**	**0.190**	0.057	0.783	0.550	**0.007**	0.153	0.153
	No Spreading/3.0																						
σ 0.0	0.285	0.723	0.002	0.275	0.294	0.048	0.660	0.580	0.010	0.154	**0.101**		0.283	**0.777**	0.003	0.287	**0.189**	0.063	0.698	0.558	0.005	0.152	0.098
σ 0.2	0.290	0.722	0.010	0.288	0.320	0.052	**0.669**	0.572	0.013	0.173	0.079		0.265	0.752	**0.007**	0.281	**0.189**	0.076	0.508	0.553	0.006	**0.160**	0.112
σ 0.4	**0.289**	**0.724**	0.010	**0.299**	**0.323**	**0.086**	0.597	**0.595**	**0.015**	0.167	0.078		0.289	0.727	0.002	0.259	**0.189**	0.060	**0.832**	0.563	**0.007**	0.150	0.106
σ 0.6	0.276	0.723	0.003	**0.299**	0.290	0.036	0.566	0.584	0.010	**0.169**	0.080		**0.294**	0.732	0.005	**0.293**	**0.189**	0.070	0.777	**0.591**	**0.007**	0.156	**0.117**
σ 0.8	0.260	0.718	**0.013**	0.271	0.229	0.042	0.520	0.561	0.014	0.151	0.079		0.273	0.703	0.004	0.271	0.186	**0.081**	0.647	0.586	0.006	0.145	0.100

Table 8.10: Unsupervised V-Measure results for the raw numerical data sets and the respective baseline *Semantic Patterns*.

Semantic Patterns

For the generation of the *Semantic Patterns* the same parameters as for the evaluation of the mixed data sets are used. The results gained by applying the *K-Means* and *EM* algorithms to the *Semantic Patterns* are listed in Tables 8.12 and 8.13. Thereby, the following observations are made:

- ***K-Means***: The performance of *K-Means*, when applied to the *Semantic Patterns* significantly increases ($V = 0.358$ at $V_\sigma = 0.021$) when compared to the results gained for the normed raw data ($V = 0.307$ at $V_\sigma = 0.021$), the unprocessed raw data ($V = 0.299$ at $V_\sigma = 0.025$) and the baseline

	K-Means												EM										
Par	**Total**	**BW**	**DI**	**GL**	**HS**	**IO**	**IR**	**SE**	**SO**	**VE**	**VO**		**Total**	**BW**	**DI**	**GL**	**HS**	**IO**	**IR**	**SE**	**SO**	**VE**	**VO**
	Raw Data																						
N	0.025	**0.006**	**0.000**	0.033	0.040	0.037	**0.061**	**0.020**	0.004	0.023	**0.023**		Not available										
NN	**0.021**	**0.006**	**0.000**	**0.023**	**0.001**	**0.033**	0.078	0.039	**0.003**	**0.004**	**0.023**		**0.005**	**0.000**	**0.000**	**0.018**	**0.008**	**0.000**	**0.000**	**0.001**	**0.001**	**0.000**	**0.026**
	Semantic Patterns																						
	No Spreading/1.0																						
σ 0.0	0.039	0.004	0.038	0.047	0.014	0.029	0.112	0.065	0.009	0.035	0.037		0.028	**0.000**	0.002	0.018	**0.020**	0.007	0.140	0.032	**0.008**	0.011	0.046
σ 0.2	**0.033**	0.004	0.050	**0.037**	**0.008**	0.040	**0.066**	0.069	**0.007**	0.034	**0.012**		0.024	**0.000**	**0.000**	0.031	0.024	0.005	0.109	0.026	0.010	**0.005**	**0.026**
σ 0.4	0.047	**0.003**	0.047	0.059	0.015	0.031	0.187	0.048	0.011	0.037	0.029		0.024	**0.000**	0.004	0.036	0.023	0.005	0.097	0.022	0.010	0.009	0.035
σ 0.6	0.034	0.002	0.030	0.041	0.016	0.039	0.132	**0.028**	0.008	0.028	0.020		**0.015**	**0.000**	0.003	**0.015**	0.023	0.006	**0.043**	**0.019**	0.009	0.007	0.027
σ 0.8	0.044	0.003	**0.044**	0.040	0.010	**0.036**	0.170	0.065	0.008	**0.030**	0.030		0.023	0.004	0.002	0.024	0.023	**0.000**	0.086	0.026	0.013	0.009	0.042
	No Spreading/1.5																						
σ 0.0	0.045	0.003	0.033	**0.030**	0.095	0.050	0.098	0.043	0.048	0.020	0.032		0.020	**0.000**	**0.000**	0.038	0.055	**0.001**	0.050	0.026	0.004	0.003	**0.018**
σ 0.2	0.043	0.003	0.031	0.059	**0.016**	0.072	0.157	**0.040**	0.007	**0.014**	0.031		0.012	**0.000**	**0.000**	0.047	0.018	**0.001**	0.011	**0.006**	0.004	**0.000**	0.034
σ 0.4	0.048	0.004	**0.030**	0.035	0.107	0.057	0.147	0.055	**0.005**	**0.014**	0.028		0.011	**0.000**	**0.000**	0.024	0.018	0.005	0.010	0.015	0.005	**0.000**	0.030
σ 0.6	**0.031**	**0.002**	0.031	0.035	0.023	0.053	**0.073**	0.038	0.019	**0.014**	0.026		**0.008**	**0.000**	**0.000**	**0.019**	**0.006**	**0.001**	**0.008**	0.017	0.004	0.002	0.023
σ 0.8	0.066	0.215	0.021	0.043	0.103	0.036	0.148	0.043	0.013	0.016	**0.023**		0.013	0.004	**0.000**	0.032	0.008	0.003	0.026	0.023	**0.003**	**0.000**	0.030
	No Spreading/2.0																						
σ 0.0	0.048	0.003	0.029	0.032	0.090	0.059	0.140	0.054	0.011	0.024	0.034		**0.008**	**0.000**	**0.000**	0.030	**0.002**	0.009	**0.000**	0.012	**0.003**	0.005	0.018
σ 0.2	0.060	0.217	**0.018**	**0.028**	0.004	0.065	0.152	0.036	0.031	**0.021**	0.031		0.019	**0.000**	0.002	0.035	**0.002**	**0.004**	0.110	**0.011**	**0.003**	**0.003**	0.021
σ 0.4	0.035	0.004	0.029	0.055	**0.003**	**0.054**	**0.093**	0.048	0.011	0.022	0.026		0.011	**0.000**	0.002	0.032	**0.002**	0.011	0.004	0.030	0.009	**0.003**	**0.015**
σ 0.6	0.039	0.002	0.023	0.039	**0.003**	0.064	0.139	**0.031**	0.038	**0.021**	0.028		0.014	0.006	0.002	0.060	0.003	0.007	**0.000**	0.028	0.006	0.005	0.021
σ 0.8	**0.034**	**0.000**	**0.018**	0.044	0.004	0.068	0.102	0.049	**0.009**	0.025	**0.021**		0.017	0.004	0.006	**0.028**	**0.002**	**0.004**	0.061	0.030	0.006	0.006	0.024
	No Spreading/3.0																						
σ 0.0	0.048	0.004	**0.001**	0.040	0.090	0.061	0.183	0.047	0.013	0.023	**0.019**		**0.015**	**0.000**	0.002	0.054	0.001	**0.004**	**0.016**	0.040	0.006	0.011	**0.018**
σ 0.2	0.037	0.004	0.019	0.033	**0.012**	0.064	0.124	0.050	0.010	0.024	0.029		0.017	**0.000**	0.002	**0.019**	0.001	0.012	0.040	0.047	**0.005**	0.013	0.029
σ 0.4	**0.028**	0.003	0.019	0.035	0.013	0.067	**0.062**	**0.036**	**0.007**	**0.015**	0.027		0.023	**0.000**	**0.000**	0.046	**0.000**	0.023	0.071	0.040	**0.005**	0.016	0.033
σ 0.6	0.034	**0.002**	0.002	**0.025**	0.088	0.061	0.072	**0.036**	0.011	**0.015**	0.030		0.017	**0.000**	0.002	0.031	0.001	0.016	0.046	0.032	0.006	**0.005**	0.026
σ 0.8	0.054	0.004	0.026	0.060	0.133	**0.059**	0.143	0.051	0.012	0.032	0.024		0.018	0.004	0.001	0.023	0.008	0.031	0.044	**0.030**	**0.005**	**0.005**	0.031

Table 8.11: Unsupervised V-Measure standard deviation results for the raw numerical data sets and the respective baseline *Semantic Patterns*.

Semantic Patterns ($V = 0.323$ at $V_\sigma = 0.043$). The robustness drops for the baseline *Semantic Patterns* but is roughly the same for the *Semantic Patterns* and the raw data.

- ***EM***: The results of the *EM* algorithm on the *Semantic Patterns* (best result: $V = 0.340$ at $V_\sigma = 0.007$) and the baseline *Semantic Patterns* ($V = 0.323$ at $V_\sigma = 0.043$) is in general lower or at maximum roughly the same in comparison to its application to the raw data ($V = 0.346$ at $V_\sigma = 0.005$).

- ***K-Means* vs. *EM***: The performance of *K-Means* on the *Semantic Patterns* gains a significant boost and beats the best *EM* result, which is achieved on the raw data. This was already observed for the mixed data.

- **Influence of the MDL override factor (C) - *K-Means***: Here, roughly the same observations as for the mixed data sets can be made. The best results can be achieved with a model that is neither too complex nor too simple. Here, this is achieved by using the value $C = 1.5$. For the mixed data sets the best choice is $C = 2.0$.

- **Influence of the MDL override factor (C) - *EM***: The *EM* algorithm cannot benefit from the *Semantic Patterns* generated for the numerical

data. Regardless of the employed C value, the results are at maximum at the same level as for the *EM* algorithm, when applied to the raw data. However, in contrast to the mixed data sets, there is a similar behavior as observed for *K-Means*: The results get better at $C = 1.5$, slightly decrease at $C = 2.0$ and drop significantly at $C = 3.0$.

- **Influence of the pre-spreading factor σ_p - *K-Means***: For *K-Means* similar patterns as for the mixed data sets can be observed: The best results for lower values of C are achieved when σ_p is lower. For larger values of C the too complex models need to be compensated with larger σ_p values.
- **Influence of the pre-spreading factor σ_p - *EM***: The different σ_p values do not have a significant influence on the results gained by the *EM* algorithm. The same behavior was already observed for the application of the *EM* algorithm to the *Semantic Patterns* of the mixed data sets.

	K-Means											EM										
Par	**Total**	**BW**	**DI**	**GL**	**HS**	**IO**	**IR**	**SE**	**SO**	**VE**	**VO**	**Total**	**BW**	**DI**	**GL**	**HS**	**IO**	**IR**	**SE**	**SO**	**VE**	**VO**
	Raw Data																					
N	0.299	0.734	**0.052**	0.335	**0.254**	0.121	**0.708**	**0.608**	0.006	0.113	0.057	Not available										
NN	**0.307**	**0.735**	0.030	**0.388**	0.019	**0.123**	0.705	0.529	**0.008**	**0.188**	**0.342**	**0.346**	**0.718**	**0.103**	**0.370**	**0.289**	**0.254**	**0.806**	**0.621**	**0.005**	**0.103**	**0.194**
	Semantic Patterns																					
	D=0.0 MDL=1.5											**D=0.0 MDL=1.0**										
σ 0.0	0.315	0.724	**0.039**	0.329	0.309	0.045	0.717	0.582	**0.026**	0.198	0.183	0.317	**0.777**	0.006	0.312	0.239	0.218	0.651	0.592	0.016	0.174	0.186
σ 0.2	**0.323**	0.724	0.025	**0.334**	0.344	**0.071**	**0.730**	0.590	0.012	0.198	0.196	**0.327**	0.752	0.001	0.318	**0.240**	0.218	0.766	**0.598**	**0.016**	**0.167**	**0.197**
σ 0.4	0.318	0.719	0.026	0.285	0.316	0.051	0.769	0.600	0.008	**0.199**	**0.203**	0.323	0.727	**0.011**	0.287	0.229	0.217	0.749	0.600	0.018	0.176	0.218
σ 0.6	0.317	0.722	0.025	0.298	**0.357**	0.040	0.712	**0.602**	0.013	**0.199**	0.201	0.317	0.732	0.009	**0.316**	0.232	**0.221**	**0.637**	0.606	0.025	0.175	0.214
σ 0.8	0.299	0.646	0.015	0.294	0.328	0.026	0.686	0.581	0.014	0.198	0.200	0.325	0.703	0.006	0.305	0.233	0.216	0.796	0.594	0.019	0.181	0.195
	D=0.5 MDL=1.0											**D=0.7 MDL=1.0**										
σ 0.0	0.333	**0.817**	0.072	0.293	0.338	**0.181**	0.611	0.614	0.009	0.164	0.234	**0.302**	**0.579**	0.082	**0.332**	**0.285**	**0.184**	0.633	**0.634**	**0.006**	0.099	0.183
σ 0.2	0.333	**0.817**	**0.076**	0.278	**0.340**	**0.181**	**0.621**	**0.621**	0.009	0.151	**0.237**	0.300	**0.579**	0.082	0.307	**0.285**	**0.184**	0.636	0.632	**0.006**	**0.117**	0.176
σ 0.4	0.326	**0.817**	0.068	0.286	0.335	**0.181**	0.587	0.604	0.009	0.149	0.228	0.301	**0.579**	**0.086**	0.310	**0.285**	**0.184**	**0.639**	0.643	**0.006**	0.095	0.183
σ 0.6	0.327	**0.817**	0.072	0.269	0.337	**0.181**	0.604	0.580	0.009	**0.166**	0.232	0.301	**0.579**	0.076	0.319	**0.285**	**0.184**	**0.639**	0.632	**0.006**	0.109	**0.185**
σ 0.8	**0.334**	**0.817**	0.071	**0.303**	0.336	**0.181**	0.610	0.605	**0.011**	0.163	0.244	0.300	**0.579**	0.079	0.311	**0.285**	**0.184**	0.633	0.633	**0.006**	0.109	0.183
	D=0.5 MDL=1.5											**D=0.7 MDL=1.5**										
σ 0.0	0.352	**0.817**	0.099	0.298	0.382	0.143	0.751	0.601	0.018	0.193	0.218	0.339	**0.579**	0.086	0.348	**0.324**	**0.242**	**0.761**	0.596	**0.013**	0.187	**0.252**
σ 0.2	**0.358**	**0.817**	0.100	**0.330**	0.385	0.163	0.751	0.588	0.015	**0.194**	**0.232**	0.339	**0.579**	0.086	**0.356**	**0.324**	**0.242**	**0.761**	0.595	0.012	0.192	0.239
σ 0.4	0.352	**0.817**	0.096	0.315	**0.387**	0.143	0.738	0.576	**0.019**	0.193	0.231	**0.340**	**0.579**	0.092	0.348	**0.324**	**0.242**	**0.761**	**0.603**	0.012	**0.194**	0.241
σ 0.6	0.348	**0.817**	**0.103**	0.288	0.383	0.158	0.716	0.579	0.015	**0.194**	0.226	0.339	**0.579**	0.094	0.355	**0.324**	**0.242**	**0.761**	0.602	0.012	0.181	0.240
σ 0.8	0.356	**0.817**	0.098	0.296	0.378	**0.166**	**0.776**	**0.604**	0.012	0.190	0.225	0.338	**0.579**	**0.107**	0.355	**0.324**	**0.242**	0.752	0.597	0.012	0.177	0.236
	D=0.5 MDL=2.0											**D=0.7 MDL=2.0**										
σ 0.0	0.329	**0.817**	0.054	**0.339**	**0.330**	0.064	0.752	0.563	0.017	0.151	0.199	0.323	**0.579**	**0.105**	0.347	**0.266**	**0.228**	0.784	0.585	**0.015**	**0.092**	0.227
σ 0.2	0.328	**0.817**	0.052	0.320	**0.330**	0.064	0.753	0.585	0.017	0.144	0.196	0.325	**0.579**	0.098	**0.359**	**0.266**	**0.228**	0.784	0.584	**0.015**	0.098	**0.238**
σ 0.4	0.331	**0.817**	0.055	0.313	**0.330**	**0.109**	**0.767**	0.562	0.012	0.149	0.194	0.323	**0.579**	**0.105**	0.358	**0.266**	**0.228**	0.784	0.576	**0.015**	0.090	0.230
σ 0.6	0.330	**0.817**	0.059	0.335	0.328	0.073	0.765	0.560	**0.019**	0.148	0.199	**0.326**	**0.579**	0.099	0.351	**0.266**	**0.228**	**0.798**	**0.595**	**0.015**	0.091	0.235
σ 0.8	**0.333**	**0.817**	**0.064**	0.321	**0.330**	0.068	0.764	**0.593**	0.013	**0.158**	**0.200**	**0.326**	**0.579**	0.104	0.361	**0.266**	**0.228**	**0.798**	0.585	**0.015**	0.090	0.237
	D=0.5 MDL=3.0											**D=0.7 MDL=3.0**										
σ 0.0	0.322	**0.817**	0.026	0.326	**0.333**	0.099	0.739	0.567	0.022	0.136	0.153	0.304	**0.579**	**0.001**	0.362	0.200	**0.228**	0.728	0.574	**0.032**	0.114	0.224
σ 0.2	0.322	**0.817**	0.029	0.326	0.320	**0.127**	0.702	**0.583**	0.017	0.150	0.150	**0.307**	**0.579**	0.000	**0.364**	0.208	**0.228**	**0.735**	0.573	0.029	0.113	0.236
σ 0.4	0.317	**0.817**	**0.035**	0.318	0.320	0.099	0.705	0.556	**0.024**	0.140	0.154	0.306	**0.579**	**0.001**	0.355	0.211	**0.228**	0.726	0.572	0.035	0.113	**0.237**
σ 0.6	**0.328**	**0.817**	0.026	**0.342**	0.328	0.118	**0.759**	0.563	0.020	0.150	0.153	**0.307**	**0.579**	**0.001**	0.363	**0.219**	**0.228**	0.729	0.575	0.029	0.113	0.233
σ 0.8	0.323	**0.817**	0.029	0.330	0.322	0.099	0.731	0.563	0.023	**0.151**	**0.161**	0.304	**0.579**	**0.001**	0.356	0.204	0.224	0.713	**0.589**	0.030	**0.119**	0.226

Table 8.12: Unsupervised V-Measure results for the raw numerical data transformed into *Semantic Patterns*.

	K-Means												EM										
Par	**Total**	**BW**	**DI**	**GL**	**HS**	**IO**	**IR**	**SE**	**SO**	**VE**	**VO**		**Total**	**BW**	**DI**	**GL**	**HS**	**IO**	**IR**	**SE**	**SO**	**VE**	**VO**
	Raw Data																						
N	0.025	**0.006**	**0.000**	**0.033**	0.040	0.037	**0.061**	**0.020**	0.004	0.023	**0.023**		Not available										
NN	**0.021**	**0.006**	**0.000**	**0.023**	**0.001**	**0.033**	0.078	0.039	**0.003**	**0.004**	**0.023**		**0.005**	**0.000**	**0.000**	**0.018**	**0.008**	**0.000**	**0.000**	**0.001**	**0.001**	**0.000**	**0.026**
	Semantic Patterns																						
	D=0.0 MDL=1.5												**D=0.0 MDL=1.0**										
σ 0.0	0.045	0.003	0.033	**0.030**	0.095	0.050	0.098	0.043	0.048	0.020	0.032		0.028	**0.000**	0.002	0.018	**0.020**	0.007	0.140	0.032	**0.008**	0.011	0.046
σ 0.2	0.043	0.003	0.031	0.059	**0.016**	0.072	0.157	**0.040**	0.007	**0.014**	0.031		0.024	**0.000**	**0.000**	0.031	0.024	0.005	0.109	0.026	0.010	**0.005**	**0.026**
σ 0.4	0.048	0.004	**0.030**	0.035	0.107	0.057	0.147	0.055	**0.005**	**0.014**	0.028		0.024	**0.000**	0.004	0.036	0.023	0.005	0.097	0.022	0.010	0.009	0.035
σ 0.6	**0.031**	**0.002**	0.031	0.035	0.023	0.053	**0.073**	0.038	0.019	**0.014**	0.026		**0.015**	**0.000**	0.003	**0.015**	0.023	0.006	**0.043**	**0.019**	0.009	0.007	0.027
σ 0.8	0.066	0.215	0.021	0.043	0.103	0.036	0.148	0.043	0.013	0.016	**0.023**		0.023	0.004	0.002	0.024	0.023	**0.000**	0.086	0.026	0.013	0.009	0.042
	D=0.5 MDL=1.0												**D=0.7 MDL=1.0**										
σ 0.0	0.022	**0.000**	0.012	0.041	0.007	**0.000**	0.076	**0.026**	0.005	0.036	**0.014**		**0.021**	**0.000**	**0.000**	0.031	**0.010**	0.008	0.093	0.024	0.006	0.014	0.021
σ 0.2	**0.018**	**0.000**	**0.010**	0.037	**0.006**	**0.000**	**0.041**	**0.026**	**0.003**	0.039	0.017		**0.021**	**0.000**	0.011	0.034	**0.010**	0.008	**0.088**	0.025	0.007	0.013	**0.015**
σ 0.4	0.023	**0.000**	0.011	0.066	0.007	**0.000**	**0.041**	0.051	**0.003**	0.038	0.016		**0.021**	**0.000**	**0.000**	0.031	**0.010**	**0.006**	0.092	0.032	0.007	0.013	0.023
σ 0.6	0.024	**0.000**	0.012	0.039	0.007	**0.000**	0.058	0.051	0.005	0.036	0.027		**0.021**	**0.000**	**0.000**	0.028	**0.010**	0.009	0.111	**0.019**	0.008	0.010	0.016
σ 0.8	0.019	**0.000**	**0.010**	**0.031**	0.007	**0.000**	**0.041**	0.041	0.005	**0.033**	0.019		0.022	**0.000**	**0.000**	0.036	**0.010**	**0.000**	0.115	0.025	**0.004**	**0.000**	0.025
	D=0.5 MDL=1.5												**D=0.7 MDL=1.5**										
σ 0.0	0.023	**0.000**	0.008	0.046	0.009	0.053	0.050	0.042	0.007	**0.001**	**0.018**		0.005	**0.000**	**0.000**	0.018	**0.000**	0.003	**0.000**	**0.000**	**0.001**	**0.000**	0.023
σ 0.2	0.021	**0.000**	0.009	0.047	0.009	0.025	0.050	0.040	**0.006**	0.002	0.021		0.006	**0.000**	**0.000**	0.029	**0.000**	**0.001**	**0.000**	**0.000**	0.003	**0.000**	0.029
σ 0.4	0.019	**0.000**	0.007	0.038	0.006	0.020	0.057	0.032	**0.006**	**0.001**	0.021		0.007	**0.000**	0.014	0.024	**0.000**	0.003	**0.000**	0.006	0.002	**0.000**	0.024
σ 0.6	0.021	**0.000**	0.008	0.044	0.009	0.024	0.061	**0.028**	0.013	**0.001**	0.023		**0.004**	**0.000**	**0.000**	**0.013**	**0.000**	0.008	**0.000**	0.004	0.003	**0.000**	**0.012**
σ 0.8	0.014	**0.000**	0.009	0.034	0.009	0.023	**0.000**	0.032	0.008	0.010	**0.018**		0.014	**0.000**	0.017	0.020	**0.000**	0.054	**0.000**	0.004	0.009	0.001	0.030
	D=0.5 MDL=2.0												**D=0.7 MDL=2.0**										
σ 0.0	**0.025**	**0.000**	0.028	**0.020**	**0.007**	**0.052**	0.067	0.036	**0.003**	0.021	0.012		0.018	**0.000**	**0.000**	0.029	**0.000**	0.111	**0.000**	**0.005**	0.004	0.001	0.028
σ 0.2	0.026	**0.000**	0.024	0.035	**0.007**	0.054	0.066	0.029	0.011	0.024	**0.011**		0.016	**0.000**	**0.000**	**0.022**	**0.000**	0.102	**0.000**	0.007	**0.002**	0.001	0.029
σ 0.4	0.030	**0.000**	0.026	0.034	**0.007**	0.082	**0.056**	0.042	0.006	0.023	0.020		0.017	**0.000**	**0.000**	**0.022**	**0.000**	0.088	0.014	0.008	0.005	0.000	0.029
σ 0.6	0.028	**0.000**	0.024	0.025	**0.007**	0.076	**0.056**	0.032	0.012	0.021	0.026		0.020	**0.000**	**0.000**	0.027	**0.000**	0.102	0.028	**0.005**	0.003	0.001	0.032
σ 0.8	**0.025**	**0.000**	**0.018**	0.031	**0.007**	0.065	0.058	**0.024**	0.009	**0.013**	0.021		**0.015**	**0.000**	**0.000**	0.029	**0.000**	**0.066**	0.014	0.007	**0.002**	**0.000**	0.029
	D=0.5 MDL=3.0												**D=0.7 MDL=3.0**										
σ 0.0	0.029	**0.000**	0.015	0.032	**0.000**	**0.088**	0.069	0.027	0.008	0.022	0.025		**0.016**	**0.000**	**0.000**	0.030	**0.000**	**0.008**	0.062	0.027	0.005	0.003	**0.020**
σ 0.2	**0.025**	**0.000**	0.015	0.017	0.008	0.097	**0.039**	**0.019**	**0.007**	**0.021**	0.027		0.020	**0.000**	**0.000**	0.029	**0.000**	0.062	0.039	0.034	**0.003**	**0.000**	0.029
σ 0.4	0.028	**0.000**	**0.012**	**0.014**	0.008	**0.088**	0.060	0.033	0.008	0.022	0.031		0.021	**0.000**	**0.000**	0.031	**0.000**	0.065	**0.038**	0.036	**0.003**	0.001	0.031
σ 0.6	0.028	**0.000**	0.015	0.023	0.008	0.092	0.064	0.034	0.010	0.022	**0.011**		0.018	**0.000**	**0.000**	0.032	0.010	**0.008**	0.075	0.019	0.009	0.001	0.030
σ 0.8	0.030	**0.000**	0.015	0.031	0.011	**0.088**	0.069	0.041	**0.007**	**0.021**	0.021		0.018	**0.000**	**0.000**	**0.028**	**0.000**	0.064	0.050	**0.012**	0.005	0.001	0.022

Table 8.13: Unsupervised V-Measure standard deviation results for the raw numerical data transformed into *Semantic Patterns*.

Summary

Again, the performance of the *K-Means* algorithm can be significantly improved by the *Semantic Patterns*. Similar to the mixed data, the *K-Means* algorithm performance slightly beats the performance of the *EM* algorithm. Together with the results gained for the mixed data, this indicates that the deployment of the *Semantic Patterns* especially makes sense when distance-based features are present. Then, even the simple *K-Means* algorithm outperforms the results achieved by the *EM* algorithm.

8.2.5 Conclusions

By looking at the results for the categorical, mixed and numerical data sets the following conclusions are drawn: By transforming the raw value-centric features into *Semantic Patterns*, significant performance gains are made for the *K-Means* algorithm. In contrast, there are no significant improvements for the

EM algorithm. This is most likely explained by the higher sophisticated model employed by the *EM* algorithm that is already adequate for modeling the raw data, and cannot be improved when used for the *Semantic Patterns*. Still, no significant performance loss can be observed. This indicates that the *Semantic Pattern Transformation* can be safely deployed – in order to gain the benefits of the semantic model for further interpretation and analysis processes – without making compromises in terms of quality. Furthermore, the results also indicate that huge performance gains can be made for the simple distance- and centroid based algorithm family. Although, only the *K-Means* algorithm was evaluated, it is assumed that other algorithms such as the *Neural Gas* algorithm family, or the *Self-Organizing Map* will also benefit from the *Semantic Patterns*. The rationale behind this assumption is that all of these algorithms are based on similar models.

8.3 Supervised Learning Evaluation

For the supervised evaluation, the same data sets and parameters as for unsupervised clustering are used within the evaluation process. Two supervised algorithms, the *Support Vector Machine* (SVM) algorithm and the decision tree based algorithm *J48* from the Weka API are utilized. Thereby, for each possible combination of parameters and data set, the respective supervised algorithm is applied for 10 iterations. For each iteration, a cross-validation operation with 10 folds is applied.

As in the unsupervised evaluation, the algorithms are applied to unprocessed and normed raw data sets, to the baseline *Semantic Patterns* and to the *Semantic Patterns*. Due to a runtime error, the *J48* algorithm could not be evaluated for parts of the numerical data sets. Therefore, all of the *J48* results are skipped for this data. However, these results are not considered as important due to two reasons: **First**, the performance achieved on the categorical and mixed data sets is already lower than that of the SVM algorithm, and significantly drops for the mixed data sets. **Second**, the partial results for the numerical data sets also indicate similar results.

The complete results of the supervised evaluation are listed at the end of this chapter in the Tables 8.16 to 8.25, and summarized as follows: The results gained by applying the *SVM* algorithm to the *Semantic Patterns* are always slightly better than when applied to the raw data sets. This indicates that the additional information about the semantic relations can be utilized by the algorithm.

8.4 Semantic Search Evaluation

One of the key features of the *Semantic Pattern Transformation* is the inclusion of the semantic relations between feature values within the *Semantic Patterns*. This additional information has many benefits that have been explained throughout this thesis. One interesting application is the deployment of a semantic-aware

search algorithm. Due to the semantic information contained within the *Semantic Patterns* this is a straightforward process that just requires the utilization of an appropriate distance measure. The exact procedure is covered in Sections 7.2.1 and 7.4.3 of the previous chapter, and the principle rationale is highlighted again in Example 19.

Example XIX: Semantic-aware search algorithm

Given *Demo Data Set 1*, which contains various instances describing the world's countries, and assuming, there is one group of countries that only export *coffee*, a large group of countries that export *cacao* and *coffee*, and also countries that export *cacao* only: Then, the large group of countries that export both commodities defines a strong semantic relation between *coffee* and *cacao*. This relation is modeled as strong link within the associative network, which is trained on the country instances.
Now, given a search query that contains the feature value *coffee*: Then, when considering a key-word matching search algorithm, only the countries that directly export *coffee* can be retrieved. This corresponds to searching related instances within the raw value-centric data set. This behavior can be improved by deploying a semantic-aware search algorithm, which considers the semantic relation between *coffee* and *cacao* and, thus, is also capable of retrieving those countries that only export the latter.
It turns out that the semantic information contained in the *Semantic Patterns* can simply be utilized for such an algorithm by applying standard distance measures.

The evaluation of the semantic-aware search algorithm is not so straightforward as its application. When large data sets are analyzed that define many different semantic relations between the feature values, the definition of the correctly retrieved results is not a simple task. Thus, another approach for the evaluation of the semantic-aware search capabilities is followed. It is based on the following assumptions and evaluations:

- **Assumption**: *A semantic-aware search algorithm retrieves better results than a keyword-matching algorithm*: This is a reasonable assumption, because a semantic-aware search algorithm utilizes the semantic relations for retrieving more accurate search results.
- **Assumption**: *Instances of the same classes are more similar than instance of different classes*: This is also an obvious and reasonable assumption. Instances of the same classes are considered as similar due to some kind of relation that supervised and unsupervised algorithms aim to discover. Although, the capability to separate different classes strongly depends on the available data, it can be assumed that the employed features and class labels are correlated in some way.

- **Evaluation – Complete Instances**: Based on these two assumptions, the following rationale for the evaluation procedure for a semantic-aware search algorithm can be defined: Given an arbitrary data set that contains instances of different classes, and further given an instance with a class label: Then, when searching for related instances, those that belong to the same class should be retrieved first. Based on the assumption that a semantic-aware search algorithm is better suited for retrieving adequate results, there is the expectation that the number of wrongly retrieved instances is lower for a semantic-search algorithm than for a key-word-matching algorithm. For this evaluation the implementation of the semantic-aware search algorithm corresponds to calculating the similarity (distance) between the *Semantic Patterns*. The key-word-matching algorithm is represented by calculating the similarity (distance) between the raw value-centric instances. The performance of these techniques can be compared by determining the number of correctly retrieved instances. It is important to note that for this evaluation complete instances are used within the search query.
- **Evaluation – Some feature values**: The previous evaluation is based on search queries that contain complete instances. However, this is not the only use case for a semantic-aware search algorithm. In many cases only one or a few feature values (search terms) are specified within the query (as described in Example 19). In order to model this behavior with the previously described evaluation procedure, feature values are removed from the complete instances. The removal of a large percentage of feature values from an instance, and using this modified instance as search query comes closer to the scenario where only one or a few feature values are used within a query. This evaluation can also be seen in the context of the capability to handle missing values, which was also mentioned as a specific capability of the *Semantic Pattern Transformation*.

Based on these discussions, the evaluation used for the evaluation of the different search algorithms is described in Algorithm 16.

This algorithm is described as follows:

- *Input*:
  - The set of instances I and the corresponding class labels contained within an arbitrary data set.
  - The distance function DF, which is used for the semantic search evaluation. For this evaluation the Cosine similarity and the Euclidean distance are employed.
- *Output*: The total performance $TOTALPERF$ of the given distance function DF on the given instance set I.
- *Line 1 to 18*: For each class c_i within the instance labels, the corresponding instances I_c are extracted and used for the performance evaluation.

Algorithm 16 Semantic search performance

Require:
I={i_1,i_2,...,i_n} {set I of instances extracted from an arbitrary data set containing class labels}
DF {distance function DF}
$TOTALPERF$ {total performance of distance measure on data set}

```
 1: for all c_i in getClasses(I) do
 2:    CORRECT(c_i) = 0 {initialize correct counter for class c_i}
 3:    TOTAL(c_i) = 0 {initialize total counter for class c_i}
 4:    I_c = getInstancesWithClass(I,c_i) {get instances with class c_i}
 5:    N_c = |I_c| {get number of instances belonging to class c_i}
 6:    for all i_i in I_c do
 7:       D = getDistances(i_i,I) {calculate distances from i_i to all other instances}
 8:       I_s = sortDistances(D,I) {sort instances according to their distances, starting
          with the smallest}
 9:       for i = 1 → N_c do
10:          TOTAL(c_i)++ {increase total counter for current class c_i}
11:          c_s = getClass(I_s(i)) {get class of instance at position i in I_s}
12:          if c_s = c_i then
13:             CORRECT(c_i)++ {increase correct counter for current class c_i}
14:          end if
15:       end for
16:    end for
17:    PERF(c_i)=CORRECT(c_i)/TOTAL(c_i) {calculate "correct rate" for given class
       c_i}
18: end for
19: TOTALPERF = meanValue(PERF) {calculate total performance on data set}
```

Thereby, for each instance within I_c the distances to all other instances are calculated, and the instance set I is sorted according to these distances. Then the class c_i is compared to the class values of the best matching N_c instances, where N_c is the number of instances of the given class c_i. Thereby, the correct number of retrieved results – when the given class c_i matches the class of the instance – are counted.

- *Line 2 to 3*: The $CORRECT$ and $TOTAL$ counters for the class c_i are initialized.

- *Line 4 to 5*: The instances I_s with the class label c_i are extracted and their count is stored in N_c.

- *Line 6 to 16*: The performance evaluation for the given class c_i and the distance function DF is executed as follows:

- *Line 7 to 8*: Given the instance i_i, then the distances to all other instances in the set I are calculated and then sorted starting at the smallest distance. The result set is stored in I_s.

- *Line 9 to 15*: For the first N_c instances of I_s the class labels are compared to the current class c_i. For each correct match, the *CORRECT* counter is increased.

- *Line 17*: The performance $PERF(c_i)$ of class c_i is calculated by dividing the number of the correctly retrieved instances by the total number of instances within the class c_i.

- *Line 19*: The total performance *TOTALPERF* of the data set is determined by calculating the mean performance value of all classes.

The remainder of this chapter describes the results gained by applying this algorithm to the different numerical, categorical and mixed data sets. Thereby, the Euclidean distance measure and the Cosine similarity are used as distance measures for the search algorithm.

8.4.1 Complete Instances

For this evaluation, the semantic search algorithm is applied to the complete instances of the categorical, mixed and numerical data sets. Thereby, the performance of two distance measures – the Euclidian distance and the Cosine similarity – is determined according to Algorithm 16. The search algorithms are thereby applied to the raw data, to the baseline *Semantic Patterns* and to the *Semantic Patterns*. For the *Semantic Pattern Transformation* the following parameters are used: $D = 0.5$, *Norm=L*, *Comb=E*, and when distance-based features are present, the parameters $C = 2.0$, $\sigma_p = 0.2$, which are required for the pre-spreading technique.

The detailed results are presented in Table 8.14, and the following observations are made:

- **General**: The results gained for the *Semantic Patterns* are significantly better than those gained for the baseline patterns and the raw data. The performance of both distance functions is roughly the same on the *Semantic Patterns* and the baseline patterns. For the raw data, the performance difference is much larger and depends on the type of data: The Euclidean distance function performs better on the categorical data and the mixed data, and the Cosine similarity achieves the better results on the numerical data.

- **Euclidean distance vs. Cosine similarity**: Due to the reasons discussed in Section 7.2.1 of the previous chapter, the Cosine similarity seems to be the best choice when comparing *Semantic Patterns*. However, the most interesting observation here is that the Cosine similarity's performance on the *Semantic Patterns* is more or less equal to that of the Euclidean distance. This is also observed for the baseline *Semantic Patterns*. For the raw data the Cosine similarity's performance is significantly lower than that of the Euclidean distance.

Based on these observations, the following conclusion is drawn: The quality of a search algorithm can be significantly improved when the raw data is transformed into *Semantic Patterns*. This improvement is based on the additional information about the semantic relations that is contained within the patterns. The most interesting observation here are the unexpected results for the Cosine similarity and the Euclidean distance when applied to the *Semantic Patterns*. Although the Cosine similarity seems to be the better choice for the *Semantic Patterns* the same performance as for the Euclidean distance is achieved. This observation will be further discussed in the next section, where search queries with only a few feature values (search terms) will be simulated by introducing a large number of missing feature values for each instance.

Data set	EUC (N)	EUC (NN)	COS (NN)	EUC (NN)	COS (NN)	EUC (NN)	COS (NN)
	RAW			Baseline		Semantic Patterns	
	Categorical						
BC	0.52	0.53	0.53	0.52	0.53	**0.54**	**0.54**
DE	0.68	0.68	0.66	0.67	0.67	**0.81**	0.81
KR	**0.54**	**0.54**	**0.54**	**0.54**	**0.54**	0.52	0.52
LY	**0.63**	**0.63**	0.59	0.60	0.57	**0.63**	0.61
MU	0.64	0.64	0.57	0.64	0.64	**0.68**	0.67
SO	0.65	0.65	0.58	0.69	0.70	**0.75**	0.73
SP	0.48	0.48	0.44	0.48	0.48	**0.62**	0.57
VO	0.80	0.80	0.62	**0.80**	**0.80**	0.78	0.79
ZO	0.84	0.83	0.80	0.85	0.84	**0.86**	**0.86**
Total	0.64	0.64	0.59	0.64	0.64	**0.69**	0.68
	Mixed						
AN	0.64	0.64	0.44	0.64	0.65	0.65	**0.66**
CO	0.59	0.59	0.50	0.59	0.60	0.58	**0.62**
CA	0.62	0.62	0.55	0.61	0.61	0.61	**0.65**
CG	**0.52**	**0.52**	0.51	0.52	**0.52**	**0.52**	**0.52**
HC	0.86	0.86	0.81	0.85	0.85	0.86	**0.87**
HH	0.87	0.87	0.84	0.86	0.86	0.86	**0.88**
HE	0.59	0.59	0.52	0.61	0.60	0.63	**0.65**
Total	0.67	0.67	0.60	0.67	0.67	0.67	**0.69**
	Numerical						
BW	0.86	0.86	0.62	0.74	0.74	0.89	**0.90**
DI	0.55	0.55	0.53	0.54	0.54	0.55	**0.56**
GL	0.49	0.49	0.51	0.51	0.51	**0.53**	**0.53**
HS	0.64	0.64	0.54	0.63	0.63	0.66	**0.69**
IO	0.51	0.51	0.46	0.55	0.55	**0.63**	0.61
IR	0.81	0.81	**0.87**	0.73	0.73	0.81	0.83
SE	**0.61**	**0.61**	0.39	0.54	0.54	0.57	0.57
SO	**0.54**	**0.54**	0.52	**0.54**	**0.54**	**0.54**	**0.54**
VE	0.35	0.35	0.36	**0.37**	**0.37**	0.36	**0.37**
VO	0.15	0.15	0.20	0.21	0.21	**0.22**	0.21
Total	0.55	0.55	0.50	0.54	0.54	**0.58**	**0.58**

Table 8.14: Semantic-aware search results.

8.4.2 Some Feature Values, or the Influence of Missing Values

By evaluating the influence of missing values on the performance of the semantic-aware search algorithms, two questions can be answered: **First**, when missing values are present within the analyzed data, some strategy needs to be applied

that appropriately replaces these values or ignores them in the analysis applied to the data. This topic was covered in Section 4.2 of *Chapter 4 – Towards a Semantic Representation* and it was argued that the transformation of raw data into *Semantic Patterns* makes the application of such strategies obsolete. This was explained due to the semantic relations, which are contained in the *Semantic Patterns*. Thus, it is reasonable to assume that this information could also help to improve the quality when missing values are present. In order to verify these postulations, the algorithm for determining the semantic-aware search performance section is modified to remove feature values for the simulation of missing values. **Second**, when a large percentage of feature values are removed from the instances, the resulting search queries can be compared to typical queries that include only a few feature values (search terms).

The following strategy is used to simulate missing values by removing feature values from the analyzed instances: The raw data sets are taken as input and feature values are removed randomly from the instances contained in the data sets. Thereby, a ratio for the missing values indicates how many of the values should be removed. Thus, a factor of 50% results in the removal of half of the feature values in each instance, which are then marked as missing values. The analyzed data sets are modified according to the selected missing value factors and stored for the further evaluation.

The performance evaluation utilizes Algorithm 16, which is used for the evaluation of the semantic-aware search algorithm. In the described process, a given percentage of feature values is removed from each feature vector, which corresponds to the presence of missing values. These modified instances are then compared to the original instances that are contained in the unmodified data set. The reason for doing this is based on the assumption that a data base is already available, and that search queries for retrieving data from this data base are executed. Here, the modified instances that contain the missing values represent these search queries. The performance is then determined in the same way as for the semantic search evaluation: The class labels of the instances are used to determine the number of the correctly retrieved instances.

For the *Semantic Patterns*, a similar strategy is used: Here, the original raw data sets are transformed into *Semantic Patterns* and stored for the further evaluation. These data sets are later used as reference data sets. Then, the *Semantic Patterns* based on the instances containing missing values are generated by utilizing the associative network trained for the reference data set. The pattern is then used as search query in order to retrieve related patterns from the reference data set. The utilization of the trained associative network corresponds to using the unmodified raw data sets as existing database. The performance evaluation of the *Semantic Patterns* that are generated for the modified instances is based on the same procedure as described before for the raw data sets.

For all evaluations the missing feature value rates of 0%, 10%, 50% and 90% are used, which means that as many feature values are removed from each instance of the analyzed data sets. Obviously the results gained by using the 0% rate are equal to those gained for the semantic search evaluation in the previous

chapter.

The results are presented in Table 8.15. Thereby, the first column describes the analyzed data set, the left part of the table the results for the Euclidean (**Euc**) distance function, and the right part of the table the results gained for the Cosine similarity (**Cos**). For both distance functions, the results are listed for the raw data sets (**Raw**) and the *Semantic Patterns* (**Semantic Patterns**). As for the previous evaluations the results are grouped into categorical, numerical and mixed data sets. The last row for each of these groups lists the total result, which is determined by calculating the mean value over all data sets for the respective distance function, type of data, and missing value rate.

The following observations are made:

- **No missing values (0%)**: This is mentioned for completeness and corresponds to the semantic search analysis in the previous section. In general, both distance functions achieve significantly improved results when applied to the *Semantic Patterns*. In contrast to the application to the raw data, where the Euclidean distance in general outperforms the Cosine similarity, this difference becomes insignificant for the *Semantic Patterns*.

- **Low missing value rate (10%)**: Here the similar behavior as for the 0% rate is observed. Obviously, there is a minor drop in performance due to the information loss that occurs due the removal of 10% of the feature values. Again, the difference between the performance of both distance measures is significant for the raw data, and insignificant for the *Semantic Patterns*.

- **High missing value rate (50% and 90%)**: Here, some interesting observations are made: **First**, for the raw data, the difference between the both distance measures is roughly the same. This is in contrast to the observations made for the low missing value rates, where the Euclidean distance outperformed the Cosine similarity. **Second**, for the *Semantic Patterns* the Cosine similarity now significantly outperforms the Euclidean distance. This is also in contrast to the low missing value rates, where both functions approximately achieved the same results.

Based on the observations the following conclusions are drawn: Due to the structure of the *Semantic Patterns*, the Cosine similarity seems to be the better choice for calculating the semantic similarity between two patterns. However, the observations show that this is only true when a large number of values are missing. When applying spreading activation to instances that contain a large number of missing values, and therefore have a significantly lower amount of feature values, there are fewer activate regions within the network. When this happens, the Cosine similarity is the better choice, because it is at maximum when the compared vectors are orthogonal[2]. These observations also correspond to empirical results that were gained during the application of the *Semantic*

[2]This orthogonality is observed when two patterns that contain activation values of distinct regions are compared.

Pattern Transformation for executing semantic search queries in a wide range of domains (*Chapter 10 – Applications*). In these empirical evaluations, a search algorithm using the Cosine similarity always retrieved the more significant results. When more and more regions within the network become active, which is the case when only a few or none feature values are removed from the instances, the Euclidean distance seems to be the better choice. Here, the strength of the activation values becomes more important, because most of the network's regions are active.

Distance	**Euc**									**Cos**							
Data	**Raw**				**Semantic Patterns**					**Raw**				**Semantic Patterns**			
Missing	**0%**	**10%**	**50%**	**90%**	**0%**	**10%**	**50%**	**90%**		**0%**	**10%**	**50%**	**90%**	**0%**	**10%**	**50%**	**90%**
	Categorical																
BC	0.52	0.52	0.52	**0.52**	**0.54**	**0.54**	**0.53**	0.50		0.53	0.53	**0.53**	**0.51**	**0.54**	**0.54**	**0.53**	**0.51**
DE	0.68	0.66	**0.55**	**0.32**	**0.81**	**0.80**	0.38	0.22		0.66	0.66	0.67	0.36	**0.81**	**0.80**	**0.74**	**0.46**
KR	**0.54**	**0.54**	**0.53**	**0.52**	0.52	0.52	0.51	0.50		**0.54**	**0.54**	**0.53**	**0.51**	0.52	0.52	0.52	**0.51**
LY	**0.63**	**0.68**	0.63	0.30	**0.63**	0.59	**0.64**	**0.48**		0.59	0.53	0.51	0.32	**0.61**	**0.58**	**0.56**	**0.35**
MU	0.64	0.64	**0.62**	0.57	**0.68**	**0.67**	**0.62**	0.53		0.57	0.57	0.56	0.54	**0.67**	**0.67**	**0.67**	**0.62**
SO	0.65	0.63	**0.53**	**0.22**	**0.75**	**0.70**	0.09	0.08		0.58	0.56	0.50	0.18	**0.73**	**0.72**	**0.63**	**0.28**
SP	0.48	**0.47**	**0.44**	0.38	**0.62**	0.46	0.39	**0.39**		0.44	0.44	0.41	0.37	**0.57**	**0.57**	**0.54**	**0.45**
VO	**0.80**	**0.79**	**0.76**	**0.67**	0.78	0.78	0.68	0.51		0.62	0.63	0.67	0.62	**0.79**	**0.79**	**0.78**	**0.72**
ZO	0.83	0.81	**0.72**	**0.31**	**0.86**	**0.85**	0.64	0.24		0.80	0.79	0.71	0.31	**0.86**	**0.84**	**0.76**	**0.41**
Total	0.64	0.64	**0.59**	**0.42**	**0.69**	**0.66**	0.50	0.38		0.59	0.58	0.57	0.41	**0.68**	**0.67**	**0.64**	**0.48**
	Mixed																
AN	0.64	0.63	**0.55**	**0.38**	**0.66**	**0.67**	0.51	**0.38**		0.44	0.46	0.50	0.38	**0.66**	**0.66**	**0.61**	**0.42**
CO	**0.59**	**0.59**	**0.56**	**0.51**	**0.59**	0.58	0.52	0.50		0.50	0.50	0.51	0.51	**0.62**	**0.62**	**0.60**	**0.57**
CA	0.62	0.61	0.59	**0.54**	**0.65**	**0.65**	**0.60**	0.52		0.55	0.55	0.54	0.51	**0.65**	**0.64**	**0.63**	**0.57**
CG	**0.52**	0.52	0.52	0.50	**0.52**	**0.53**	**0.54**	**0.53**		0.51	0.51	**0.52**	0.51	**0.52**	**0.52**	**0.52**	**0.52**
HC	0.86	0.86	**0.85**	**0.81**	**0.87**	**0.87**	**0.85**	**0.81**		0.81	0.81	0.82	0.81	**0.87**	**0.87**	**0.86**	**0.84**
HH	0.87	0.86	**0.85**	**0.82**	0.87	0.87	0.83	0.80		0.84	0.84	0.83	0.81	**0.88**	**0.88**	**0.87**	**0.83**
HE	0.59	0.58	0.56	0.50	**0.64**	**0.64**	**0.58**	**0.55**		0.52	0.51	0.55	0.52	**0.65**	**0.65**	**0.64**	**0.57**
Total	0.67	0.67	**0.64**	**0.58**	**0.69**	**0.69**	0.63	**0.58**		0.60	0.60	0.61	0.58	**0.69**	**0.69**	**0.68**	**0.62**
	Numerical																
BW	0.86	0.86	0.76	0.68	**0.91**	**0.91**	**0.84**	**0.69**		0.62	0.61	0.59	0.50	**0.90**	**0.89**	**0.88**	**0.84**
DI	0.55	0.54	0.53	**0.53**	**0.56**	**0.55**	**0.54**	0.50		0.53	0.53	0.52	0.50	**0.56**	**0.55**	**0.55**	**0.53**
GL	0.49	0.45	0.31	0.30	**0.53**	**0.52**	**0.42**	**0.31**		0.51	0.51	**0.48**	0.29	**0.53**	**0.52**	**0.48**	**0.34**
HS	0.64	0.63	0.59	0.52	**0.69**	**0.69**	**0.61**	**0.53**		0.54	0.54	0.55	0.51	**0.69**	**0.69**	**0.65**	**0.60**
IO	0.51	0.52	0.55	**0.54**	**0.61**	**0.61**	**0.56**	0.46		0.46	0.46	0.47	0.51	**0.61**	**0.61**	**0.60**	**0.57**
IR	0.81	0.60	0.47	0.33	**0.83**	**0.81**	**0.75**	**0.67**		**0.87**	**0.84**	**0.77**	**0.34**	0.84	0.81	0.76	0.75
SE	0.61	0.53	0.21	0.15	**0.57**	**0.57**	**0.43**	**0.17**		0.39	0.40	0.44	0.27	**0.57**	**0.57**	**0.55**	**0.41**
SO	**0.54**	0.53	**0.51**	**0.50**	**0.54**	**0.54**	**0.51**	**0.50**		0.52	0.52	0.52	0.52	**0.54**	**0.54**	**0.54**	**0.53**
VE	0.35	0.33	0.29	0.26	**0.37**	**0.37**	**0.35**	**0.28**		0.36	0.36	**0.36**	0.31	**0.37**	**0.37**	**0.36**	**0.33**
VO	0.15	0.15	0.12	0.09	**0.22**	**0.21**	**0.16**	**0.10**		0.20	0.20	0.17	0.10	**0.21**	**0.21**	**0.20**	**0.13**
Total	0.55	0.51	0.43	0.39	**0.58**	**0.58**	**0.52**	**0.42**		0.50	0.50	0.49	0.38	**0.58**	**0.58**	**0.56**	**0.50**

Table 8.15: Semantic-aware search results when missing values are present.

8.4.3 Conclusions

The *Semantic Patterns* significantly improve the quality of a search algorithm in comparison to the raw data. When only a low number of feature values are

involved in the search query, the Cosine similarity is definitely the better choice. However, for search queries that include complete instances the performance of the Euclidean distance and the Cosine similarity becomes almost equal. This has two significant implications: **First**, for the standard deployment of a semantic-aware search algorithm, the Cosine similarity should be used. In the typical scenarios, the number of feature values within a search query will always remain significantly lower than the number of feature values contained in the instances of the underlying data base. **Second**, when complete instances are compared the Euclidean distance achieves a similar performance as the Cosine similarity. This is especially important for supervised and unsupervised algorithms that employ the Euclidean distance-measure, as it implies that these algorithms can be safely applied to the *Semantic Patterns* without any modification.

Interestingly, the application of fanout strategies for the supervised and unsupervised evaluation did not yield any improved results. On the contrary, in most of the cases the V-Measure results dropped significantly. This might be explained by the large number of connections and normalized weights within the network. Due to the relations created in the analyzed instances the nodes are tightly interconnected, and it seems that there is no significant difference between the different nodes which justifies the application of fanout strategies during spreading activation.

8.5 Empirical Evaluation

During the development of the *Semantic Pattern Transformation* many empirical evaluations for all of the various aspects have been conducted. In this chapter, the results have been extended for supervised and unsupervised learning, semantic search, and the influence of missing values.

For all the empirical aspects of the *Semantic Pattern Transformation* the reader is referred to the published works that will be described in *Chapter 10 – Applications*. In these references other analysis processes including anomaly detection, time-based analysis, feature relevance and visualization will be covered.

8.6 Chapter Conclusions

The results for the individual evaluation procedures are summarized as follows: For unsupervised learning, the simple distance-based algorithms gain a significant boost in the quality of the results when utilizing the *Semantic Patterns*. For the higher sophisticated algorithm *EM*, the performance remains roughly the same. For supervised learning, a slight improvement can be achieved. The final evaluation of the semantic-aware search algorithms reveals that the quality of the results is significantly improved by utilizing the *Semantic Patterns*. Furthermore, when complete instances are analyzed, the results gained by the Euclidean distance are equal to those of the Cosine similarity.

All together this leads to the following conclusions: The quality of the results, when utilizing the *Semantic Patterns*, is at least equal and in most cases significantly better, than when using the raw data. This means, that the huge benefits of the semantic representation can be used without making compromises in terms of quality. Furthermore, due to the results gained for the semantic-aware search algorithm, the conclusion can be drawn that unsupervised algorithms based on the Euclidean distance can be applied to the *Semantic Patterns*.

	SMO											J48									
Par	**Total**	**BC**	**DE**	**KR**	**LY**	**MU**	**SO**	**SP**	**VO**	**ZO**		**Total**	**BC**	**DE**	**KR**	**LY**	**MU**	**SO**	**SP**	**VO**	**ZO**
	Raw Data																				
N	**0.729**	0.033	**0.925**	0.750	**0.530**	**1.000**	0.918	0.712	**0.755**	0.939		Not available									
NN	0.727	0.036	0.924	0.752	0.510	**1.000**	0.918	**0.719**	0.739	0.941		**0.718**	**0.079**	0.838	**0.952**	0.285	**1.000**	0.895	**0.755**	**0.765**	**0.896**
	Semantic Patterns																				
D 0.0	0.723	**0.038**	0.924	**0.816**	0.392	**1.000**	**0.920**	0.718	0.742	**0.954**		0.717	0.045	**0.891**	0.942	**0.289**	**1.000**	**0.918**	**0.755**	0.726	0.886

Table 8.16: Supervised V-Measure results for the raw categorical data sets and the respective baseline *Semantic Patterns*.

	SMO											J48									
Par	**Total**	**BC**	**DE**	**KR**	**LY**	**MU**	**SO**	**SP**	**VO**	**ZO**	**Total**	**BC**	**DE**	**KR**	**LY**	**MU**	**SO**	**SP**	**VO**	**ZO**
	Raw Data																				
N	0.010	0.011	0.005	0.010	0.033	**0.000**	0.004	0.007	0.014	0.007	Not available										
NN	**0.009**	**0.005**	**0.005**	0.009	0.032	**0.000**	**0.003**	0.007	**0.011**	**0.005**	**0.014**	0.030	**0.012**	**0.005**	**0.029**	**0.000**	0.009	0.009	**0.023**	**0.008**
	Semantic Patterns																				
D 0.0	0.010	0.010	**0.005**	**0.006**	**0.028**	**0.000**	0.004	**0.005**	0.021	0.009	**0.014**	**0.015**	0.017	0.006	0.039	**0.000**	**0.003**	**0.008**	0.027	**0.008**

Table 8.17: Supervised V-Measure standard deviation results for the raw categorical data sets and the respective baseline *Semantic Patterns*.

	SMO										J48									
Par	**Total**	**BC**	**DE**	**KR**	**LY**	**MU**	**SO**	**SP**	**VO**	**ZO**	**Total**	**BC**	**DE**	**KR**	**LY**	**MU**	**SO**	**SP**	**VO**	**ZO**
	Raw Data																			
N	**0.729**	0.033	**0.925**	0.750	**0.530**	**1.000**	**0.918**	0.712	**0.755**	0.939	Not available									
NN	0.727	**0.036**	0.924	**0.752**	0.510	**1.000**	**0.918**	**0.719**	0.739	**0.941**	**0.718**	**0.079**	**0.838**	**0.952**	**0.285**	**1.000**	**0.895**	**0.755**	**0.765**	**0.896**
	Semantic Patterns																			
D 0.0	**0.723**	**0.038**	**0.924**	**0.816**	**0.392**	**1.000**	**0.920**	**0.718**	**0.742**	**0.954**	**0.717**	**0.045**	**0.891**	**0.942**	**0.289**	**1.000**	**0.918**	**0.755**	**0.726**	**0.886**
	Comb=E Norm=L																			
D 0.1	0.726	0.036	0.937	**0.798**	0.423	**1.000**	0.920	0.748	**0.732**	0.941	**0.689**	**0.096**	0.895	0.872	0.251	**0.991**	0.873	**0.693**	**0.667**	0.864
D 0.3	**0.730**	**0.041**	0.942	0.745	0.456	0.999	0.923	0.793	0.704	**0.967**	0.688	0.091	0.895	0.873	0.234	**0.991**	0.875	0.688	0.665	0.884
D 0.5	0.727	0.036	**0.946**	0.723	0.480	0.995	**0.927**	**0.799**	0.670	0.966	0.687	0.089	**0.920**	0.873	0.280	**0.991**	**0.880**	0.692	0.571	0.885
D 0.7	0.719	0.036	0.937	0.702	**0.485**	0.994	0.919	0.793	0.647	0.957	0.677	0.063	0.904	**0.883**	**0.291**	0.990	0.862	0.672	0.523	**0.901**
	Comb=S Norm=L																			
D 0.1	**0.731**	**0.039**	0.941	**0.775**	0.462	**1.000**	0.921	0.770	**0.719**	0.950	**0.694**	0.077	0.898	0.875	**0.282**	0.991	**0.874**	0.695	**0.675**	0.881
D 0.3	**0.731**	0.038	**0.946**	0.723	**0.487**	0.995	**0.925**	**0.799**	0.704	**0.966**	0.686	0.075	0.903	**0.882**	0.275	**0.992**	0.871	**0.695**	0.604	0.879
D 0.5	0.723	0.036	0.941	0.719	0.483	0.994	0.921	0.793	0.670	0.950	0.678	0.080	0.906	0.874	0.255	0.990	0.852	0.682	0.593	0.873
D 0.7	0.709	0.036	0.944	0.692	0.464	0.994	0.915	0.784	0.596	0.958	0.679	**0.083**	**0.914**	0.871	0.281	0.987	0.863	0.671	0.555	**0.887**
	Comb=E Norm=S																			
D 0.1	0.720	**0.039**	0.924	**0.816**	0.394	**1.000**	0.921	0.711	0.733	0.940	0.688	0.061	0.880	**0.881**	0.296	0.991	**0.872**	0.708	0.663	**0.838**
D 0.3	**0.721**	0.032	0.923	**0.816**	**0.396**	**1.000**	0.920	**0.718**	**0.747**	0.941	**0.689**	**0.077**	0.890	0.871	0.291	0.991	0.867	**0.715**	**0.666**	0.834
D 0.5	0.717	0.033	0.923	0.814	0.386	**1.000**	**0.922**	0.712	0.727	0.939	0.684	0.062	0.891	0.873	0.268	**0.992**	**0.872**	**0.715**	0.659	0.823
D 0.7	0.719	0.036	**0.925**	0.812	0.384	**1.000**	0.920	**0.718**	0.729	**0.948**	0.688	0.060	**0.904**	0.872	**0.303**	**0.992**	0.870	0.708	0.651	0.830
	Comb=S Norm=S																			
D 0.1	0.721	**0.042**	0.923	0.814	0.384	**1.000**	0.920	0.717	0.740	**0.950**	0.685	0.052	0.878	0.878	0.302	0.991	**0.875**	0.707	**0.656**	0.830
D 0.3	0.722	0.035	0.924	**0.816**	0.405	**1.000**	0.919	0.715	0.741	0.939	0.689	**0.082**	0.894	**0.879**	0.285	**0.992**	0.868	0.710	0.653	**0.842**
D 0.5	**0.725**	0.035	0.926	0.815	0.420	**1.000**	**0.921**	0.720	**0.742**	0.944	0.684	0.071	0.907	0.874	0.275	0.991	0.870	0.711	0.646	0.808
D 0.7	0.723	0.029	**0.931**	0.809	**0.421**	**1.000**	0.919	**0.729**	0.731	0.938	**0.690**	0.065	**0.909**	0.877	**0.308**	0.991	0.865	**0.713**	0.644	0.835

Table 8.18: Supervised V-Measure results for the raw categorical data transformed into *Semantic Patterns*.

	SMO											J48									
Par	**Total**	**BC**	**DE**	**KR**	**LY**	**MU**	**SO**	**SP**	**VO**	**ZO**		**Total**	**BC**	**DE**	**KR**	**LY**	**MU**	**SO**	**SP**	**VO**	**ZO**
	Raw Data																				
N	0.010	0.011	**0.005**	0.010	0.033	**0.000**	0.004	**0.007**	0.014	0.007		Not available									
NN	**0.009**	**0.005**	**0.005**	**0.009**	**0.032**	**0.000**	**0.003**	**0.007**	**0.011**	**0.005**		**0.014**	**0.030**	**0.012**	**0.005**	**0.029**	**0.000**	**0.009**	**0.009**	**0.023**	**0.008**
	Semantic Patterns																				
D 0.0	**0.010**	**0.010**	**0.005**	**0.006**	**0.028**	**0.000**	**0.004**	**0.005**	**0.021**	**0.009**		**0.014**	**0.015**	**0.017**	**0.006**	**0.039**	**0.000**	**0.003**	**0.008**	**0.027**	**0.008**
	Comb=E Norm=L																				
D 0.1	0.010	**0.010**	**0.004**	0.007	0.033	**0.000**	0.004	0.006	0.015	0.007		**0.017**	0.039	**0.011**	0.011	**0.024**	0.002	**0.007**	0.012	**0.023**	0.023
D 0.3	0.009	**0.010**	0.007	0.006	0.032	0.001	0.003	0.008	**0.010**	**0.000**		**0.017**	0.030	0.013	0.009	0.045	0.002	**0.007**	0.013	**0.023**	**0.015**
D 0.5	0.010	**0.010**	0.006	**0.002**	0.045	0.001	0.004	0.005	0.012	0.005		0.018	0.023	0.018	0.013	0.035	**0.001**	**0.007**	**0.008**	0.033	0.021
D 0.7	**0.008**	**0.010**	0.008	0.003	**0.020**	0.001	**0.002**	**0.004**	0.018	0.008		0.016	**0.012**	0.015	**0.005**	0.048	0.002	0.008	0.009	0.024	0.023
	Comb=S Norm=L																				
D 0.1	0.010	0.010	0.006	0.012	0.037	**0.000**	0.005	**0.004**	0.013	**0.000**		0.018	0.027	0.012	0.012	0.038	0.002	**0.007**	0.010	0.032	0.021
D 0.3	0.009	0.011	0.008	**0.004**	0.030	**0.000**	0.003	**0.004**	0.020	0.005		0.020	0.031	**0.011**	**0.008**	0.040	0.002	**0.007**	0.009	0.027	0.041
D 0.5	**0.007**	**0.006**	**0.005**	0.005	0.020	0.001	0.004	0.005	**0.009**	0.007		0.017	0.021	**0.011**	0.010	0.050	0.002	0.011	**0.007**	**0.020**	0.020
D 0.7	**0.007**	0.011	0.007	0.005	**0.015**	0.001	**0.002**	**0.004**	0.014	0.008		**0.015**	**0.022**	0.015	0.010	**0.035**	**0.001**	0.010	0.012	0.023	**0.011**
	Comb=E Norm=S																				
D 0.1	0.012	0.017	0.005	**0.006**	0.050	**0.000**	**0.003**	0.009	**0.013**	0.005		**0.015**	**0.020**	0.012	0.011	**0.037**	0.002	**0.005**	0.011	0.020	**0.014**
D 0.3	**0.010**	**0.008**	0.007	**0.006**	0.034	**0.000**	0.004	**0.007**	0.022	0.006		0.016	0.029	**0.010**	**0.007**	0.042	0.002	0.010	**0.009**	0.020	**0.014**
D 0.5	0.011	0.012	**0.004**	**0.006**	0.031	**0.000**	0.004	0.011	0.026	**0.004**		0.017	0.029	0.014	0.009	0.042	0.002	**0.005**	0.010	**0.019**	0.026
D 0.7	0.011	0.016	0.006	0.008	**0.025**	**0.000**	**0.003**	0.009	0.018	0.015		0.016	0.022	0.015	0.011	0.044	**0.001**	**0.005**	0.010	0.023	0.017
	Comb=S Norm=S																				
D 0.1	0.012	0.012	**0.004**	0.008	0.035	**0.000**	0.005	0.009	0.019	0.014		0.018	0.020	0.016	0.010	0.047	**0.001**	0.007	**0.008**	0.022	0.027
D 0.3	0.011	0.012	0.005	**0.005**	0.039	**0.000**	**0.002**	0.013	0.017	0.005		**0.014**	0.020	0.016	0.007	0.035	0.002	**0.005**	0.009	**0.020**	**0.008**
D 0.5	**0.010**	0.013	0.007	0.007	**0.022**	**0.000**	0.005	0.012	**0.015**	0.010		**0.014**	**0.018**	0.014	**0.004**	**0.030**	**0.001**	0.007	0.013	0.022	0.019
D 0.7	**0.010**	**0.011**	0.006	0.006	0.038	**0.000**	0.004	**0.005**	0.020	**0.000**		0.017	0.020	**0.013**	0.009	0.040	**0.001**	0.008	0.012	0.026	0.022

Table 8.19: Supervised V-Measure standard deviation results for the raw categorical data transformed into *Semantic Patterns*.

	SMO								J48							
Par	**Total**	**AN**	**CO**	**CA**	**CG**	**HC**	**HH**	**HE**	**Total**	**AN**	**CO**	**CA**	**CG**	**HC**	**HH**	**HE**
	Raw Data															
N	0.374	0.857	0.309	0.410	**0.108**	0.359	**0.324**	0.248	Not available							
NN	**0.379**	**0.861**	**0.321**	**0.414**	0.104	**0.362**	0.310	**0.278**	**0.320**	**0.917**	**0.380**	**0.399**	**0.056**	**0.192**	**0.222**	**0.075**
	Semantic Patterns															
	No Spreading/1.0															
σ 0.0	0.375	0.986	0.275	0.417	0.117	0.338	0.313	**0.178**	0.320	**0.929**	**0.322**	**0.417**	0.056	0.223	0.208	**0.088**
σ 0.2	0.374	0.984	0.275	0.422	**0.121**	0.339	0.315	0.163	0.320	**0.929**	0.316	0.408	0.053	0.235	0.210	0.086
σ 0.4	**0.378**	**0.990**	**0.287**	0.423	0.118	**0.352**	0.319	0.157	**0.323**	**0.929**	**0.332**	0.406	0.055	**0.247**	**0.228**	0.061
σ 0.6	0.373	**0.990**	0.282	**0.428**	0.116	0.328	0.300	0.164	0.318	**0.929**	0.305	0.409	0.053	0.236	0.214	0.079
σ 0.8	0.373	0.988	0.275	0.426	0.117	0.331	**0.321**	0.151	0.318	0.927	0.308	0.415	**0.060**	0.221	0.218	0.077
	No Spreading/1.5															
σ 0.0	0.358	0.987	0.280	**0.423**	0.122	0.263	0.315	0.113	0.311	**0.935**	0.277	0.416	**0.088**	0.198	0.201	0.064
σ 0.2	0.361	**0.990**	0.288	0.414	**0.131**	**0.277**	0.317	0.113	0.311	0.927	0.278	0.408	0.076	0.206	**0.216**	**0.069**
σ 0.4	0.356	0.985	**0.295**	0.421	0.125	0.268	0.309	0.092	0.312	**0.935**	0.279	**0.428**	0.080	0.204	0.212	0.045
σ 0.6	**0.363**	0.988	0.278	0.419	0.128	0.272	0.318	**0.136**	**0.316**	0.929	0.279	0.426	0.083	**0.209**	0.212	0.071
σ 0.8	0.360	0.985	0.283	0.411	0.123	0.273	**0.322**	0.123	0.311	0.924	**0.295**	0.417	0.080	0.196	0.212	0.054
	No Spreading/2.0															
σ 0.0	**0.358**	0.990	0.253	0.397	0.110	0.312	**0.321**	0.121	0.308	0.916	0.278	0.397	0.071	0.220	0.207	0.070
σ 0.2	0.357	0.988	0.250	**0.401**	0.105	0.306	0.307	**0.143**	**0.315**	**0.918**	**0.295**	**0.402**	0.073	**0.222**	**0.214**	0.082
σ 0.4	0.349	0.991	0.241	0.395	0.106	0.299	0.299	0.113	0.312	0.912	0.293	0.387	**0.076**	0.215	0.202	**0.102**
σ 0.6	0.355	**0.993**	**0.254**	0.399	0.111	**0.314**	0.302	0.113	0.311	0.907	**0.295**	0.396	0.069	0.207	**0.214**	0.091
σ 0.8	0.353	0.988	0.253	0.397	**0.113**	0.305	0.291	0.127	0.309	0.916	0.288	0.395	0.067	0.208	0.209	0.082
	No Spreading/3.0															
σ 0.0	0.342	0.987	0.281	**0.405**	0.116	0.243	0.235	**0.125**	**0.309**	0.915	0.292	0.377	0.069	0.209	**0.221**	**0.080**
σ 0.2	0.342	0.985	0.277	0.401	0.113	**0.246**	0.250	0.122	0.308	0.916	**0.306**	0.382	0.065	0.208	0.215	0.064
σ 0.4	0.338	**0.991**	0.262	0.400	0.117	**0.246**	0.242	0.108	**0.309**	0.914	**0.306**	0.375	0.069	**0.211**	0.214	0.073
σ 0.6	**0.343**	0.988	**0.284**	0.399	**0.119**	0.245	**0.251**	0.114	0.303	**0.917**	0.292	0.368	**0.077**	0.196	0.210	0.059
σ 0.8	0.341	0.988	0.277	0.395	0.118	0.243	0.250	0.116	0.303	**0.917**	0.284	**0.384**	0.071	0.194	0.210	0.062

Table 8.20: Supervised V-Measure results for the raw mixed data sets and the respective baseline *Semantic Patterns*.

	SMO								J48							
Par	**Total**	**AN**	**CO**	**CA**	**CG**	**HC**	**HH**	**HE**	**Total**	**AN**	**CO**	**CA**	**CG**	**HC**	**HH**	**HE**
	Raw Data															
N	0.018	**0.017**	0.020	0.010	**0.004**	**0.014**	0.018	0.042	Not available							
NN	**0.016**	0.024	**0.015**	**0.005**	0.008	0.018	**0.015**	**0.028**	**0.017**	**0.011**	**0.007**	**0.017**	**0.013**	**0.028**	**0.023**	**0.019**
	Semantic Patterns															
	No Spreading/1.0															
σ 0.0	0.017	0.009	0.026	0.012	**0.006**	0.017	0.018	0.031	0.020	**0.005**	0.029	0.017	0.008	0.026	0.019	0.033
σ 0.2	**0.014**	0.010	**0.016**	**0.006**	0.007	0.018	**0.010**	0.033	**0.017**	0.007	**0.016**	**0.012**	0.010	0.031	**0.014**	0.032
σ 0.4	0.017	**0.006**	0.023	**0.006**	0.008	0.021	0.022	0.034	0.019	0.009	0.022	0.014	0.010	0.023	0.039	**0.017**
σ 0.6	0.016	**0.006**	0.026	0.011	0.008	0.018	0.017	**0.023**	0.020	0.010	0.024	0.020	**0.006**	0.024	0.020	0.035
σ 0.8	0.015	0.010	**0.016**	0.014	0.012	**0.015**	0.015	**0.023**	0.020	0.009	0.020	0.018	0.009	**0.022**	0.023	0.040
	No Spreading/1.5															
σ 0.0	0.017	**0.006**	**0.016**	0.021	0.013	0.013	**0.017**	0.030	0.021	0.007	0.024	0.022	0.013	0.028	0.016	0.037
σ 0.2	0.018	**0.006**	0.022	0.020	**0.008**	0.022	0.021	**0.025**	0.020	0.005	0.029	0.015	0.014	0.021	0.025	0.032
σ 0.4	0.020	0.009	0.024	0.012	0.009	0.025	0.029	0.030	**0.017**	**0.004**	0.021	0.027	**0.009**	**0.020**	0.024	**0.015**
σ 0.6	0.017	**0.006**	0.017	0.016	0.009	0.024	0.019	0.028	**0.017**	0.007	**0.013**	**0.012**	0.013	0.032	0.019	0.026
σ 0.8	**0.015**	**0.006**	0.020	**0.011**	0.012	**0.011**	0.019	0.029	0.021	0.016	0.018	0.027	0.017	0.029	**0.014**	0.023
	No Spreading/2.0															
σ 0.0	0.018	0.006	0.026	0.016	0.010	**0.013**	0.018	0.037	0.018	**0.008**	0.012	0.018	0.013	0.027	0.026	0.019
σ 0.2	**0.017**	0.010	0.026	**0.012**	**0.007**	0.021	**0.013**	**0.027**	**0.016**	**0.008**	**0.010**	0.017	0.010	0.036	**0.008**	0.024
σ 0.4	0.018	0.004	0.021	0.018	0.009	0.031	0.015	0.030	0.021	**0.008**	0.023	0.028	**0.008**	0.032	0.015	0.030
σ 0.6	0.022	**0.000**	0.029	0.019	0.010	0.033	0.030	0.035	0.019	0.010	0.032	0.019	0.011	**0.017**	0.021	0.025
σ 0.8	0.021	0.007	**0.018**	**0.012**	0.011	0.031	0.023	0.046	0.020	**0.008**	0.026	**0.012**	0.013	0.022	0.039	**0.017**
	No Spreading/3.0															
σ 0.0	**0.014**	0.009	**0.016**	**0.008**	**0.006**	**0.017**	0.020	**0.020**	0.018	**0.008**	0.025	0.019	0.010	0.021	0.015	0.028
σ 0.2	0.021	0.008	0.029	0.014	0.011	0.023	0.027	0.034	0.019	0.011	0.020	0.021	0.011	0.025	0.024	**0.021**
σ 0.4	0.016	**0.004**	0.023	0.014	**0.006**	**0.017**	0.022	0.025	0.018	**0.008**	**0.014**	0.018	0.011	0.036	**0.014**	0.023
σ 0.6	0.018	0.010	0.017	0.023	0.008	**0.017**	**0.017**	0.033	**0.017**	0.009	0.018	**0.015**	**0.008**	0.017	0.028	0.026
σ 0.8	0.017	0.007	0.018	0.019	0.008	0.018	0.022	0.027	**0.017**	0.009	0.020	0.017	0.014	**0.016**	0.020	0.023

Table 8.21: Supervised V-Measure standard deviation results for the raw mixed data sets and the respective baseline *Semantic Patterns*.

	SMO								J48							
Par	**Total**	**AN**	**CO**	**CA**	**CG**	**HC**	**HH**	**HE**	**Total**	**AN**	**CO**	**CA**	**CG**	**HC**	**HH**	**HE**
	Raw Data															
N	0.374	0.857	0.309	0.410	**0.108**	0.359	**0.324**	0.248	Not available							
NN	**0.379**	**0.861**	**0.321**	**0.414**	0.104	**0.362**	0.310	**0.278**	**0.320**	**0.917**	**0.380**	**0.399**	**0.056**	**0.192**	**0.222**	**0.075**
	Semantic Patterns															
	D=0.0 MDL=1.0								**D=0.0 MDL=1.0**							
σ 0.0	0.375	0.986	0.275	0.417	0.117	0.338	0.313	**0.178**	0.320	0.929	0.322	0.417	0.056	0.223	0.208	0.088
σ 0.2	0.374	0.984	0.275	0.422	**0.121**	0.339	0.315	0.163	0.320	0.929	0.316	0.408	0.053	0.235	0.210	**0.086**
σ 0.4	**0.378**	**0.990**	**0.287**	0.423	0.118	**0.352**	0.319	0.157	**0.323**	0.929	**0.332**	0.406	**0.055**	**0.247**	0.228	0.061
σ 0.6	0.373	**0.990**	0.282	**0.428**	0.116	0.328	0.300	0.164	0.318	0.929	0.305	**0.409**	0.053	0.236	**0.214**	0.079
σ 0.8	0.373	0.988	0.275	0.426	0.117	0.331	**0.321**	0.151	0.318	**0.927**	0.308	0.415	**0.060**	0.221	0.218	0.077
	D=0.7 MDL=1.0								**D=0.1 MDL=1.0**							
σ 0.0	0.385	0.911	0.335	0.393	0.110	0.371	**0.345**	**0.230**	0.294	0.842	0.226	0.352	0.049	0.233	0.231	0.127
σ 0.2	**0.387**	0.915	**0.340**	**0.400**	0.112	0.376	0.339	0.228	**0.300**	0.851	0.229	0.357	0.043	**0.249**	0.223	**0.149**
σ 0.4	0.380	0.914	0.335	0.393	0.107	0.358	0.329	0.225	0.296	0.851	0.229	0.362	0.046	0.241	0.223	0.122
σ 0.6	0.384	0.910	0.334	0.390	**0.114**	**0.378**	0.332	0.227	0.297	0.839	**0.242**	0.361	**0.051**	0.240	0.224	0.123
σ 0.8	0.383	**0.919**	0.325	0.394	0.112	0.374	0.336	0.218	0.298	**0.857**	0.221	**0.363**	0.046	0.225	**0.234**	0.143
	D=0.7 MDL=1.5								**D=0.1 MDL=1.5**							
σ 0.0	0.363	0.916	0.293	0.374	0.119	0.301	0.323	0.218	0.272	0.847	0.194	0.349	0.053	0.225	0.187	0.051
σ 0.2	**0.371**	**0.919**	**0.304**	0.378	0.116	0.302	**0.348**	0.228	0.279	0.846	0.201	0.353	0.056	0.224	0.201	0.070
σ 0.4	0.370	0.918	0.290	0.378	0.115	0.302	0.338	**0.246**	0.278	**0.855**	0.195	**0.361**	0.058	**0.227**	0.200	0.052
σ 0.6	0.370	0.914	0.298	**0.382**	**0.120**	0.299	0.345	0.233	0.278	0.840	0.195	0.352	0.054	0.226	0.201	**0.075**
σ 0.8	0.370	0.913	0.295	0.377	0.115	**0.312**	0.338	0.240	**0.283**	0.850	**0.203**	0.354	**0.066**	0.232	**0.204**	0.071
	D=0.7 MDL=2.0								**D=0.1 MDL=2.0**							
σ 0.0	0.367	0.932	0.317	0.389	**0.107**	0.320	0.337	**0.166**	0.281	0.856	**0.226**	0.327	**0.038**	0.200	0.174	0.149
σ 0.2	0.366	0.935	**0.318**	0.393	0.106	0.322	**0.338**	0.152	**0.286**	0.857	0.193	**0.344**	0.037	0.218	**0.221**	0.134
σ 0.4	0.363	**0.942**	0.300	0.391	0.102	0.328	0.331	0.149	0.279	0.851	0.195	0.332	0.033	0.216	0.211	0.115
σ 0.6	0.361	0.933	0.303	0.383	0.105	0.317	0.333	0.152	**0.286**	0.854	0.208	0.331	0.037	0.221	0.194	**0.154**
σ 0.8	**0.369**	**0.942**	**0.318**	**0.394**	0.105	**0.336**	0.335	0.153	0.284	**0.858**	0.204	0.332	0.037	**0.223**	0.203	0.128
	D=0.7 MDL=3.0								**D=0.1 MDL=3.0**							
σ 0.0	0.360	0.936	**0.329**	0.389	0.104	**0.273**	**0.285**	0.203	**0.286**	0.864	**0.246**	0.322	0.035	**0.232**	0.217	0.084
σ 0.2	0.356	0.934	0.328	0.384	0.107	0.265	0.274	0.199	0.281	**0.867**	0.229	0.338	**0.043**	0.204	**0.223**	0.065
σ 0.4	0.357	0.934	0.320	0.386	**0.113**	0.255	0.278	0.212	0.278	0.858	0.228	0.303	0.034	0.226	0.201	0.096
σ 0.6	0.360	0.936	0.323	**0.394**	0.108	0.261	0.283	**0.215**	0.284	**0.867**	0.224	**0.347**	0.037	0.214	0.213	0.087
σ 0.8	**0.361**	**0.942**	**0.329**	0.392	0.112	0.271	0.281	0.203	0.279	0.864	0.230	0.316	0.035	0.190	0.221	**0.099**

Table 8.22: Supervised V-Measure results for the raw mixed data transformed into *Semantic Patterns*.

	SMO								J48							
Par	**Total**	**AN**	**CO**	**CA**	**CG**	**HC**	**HH**	**HE**	**Total**	**AN**	**CO**	**CA**	**CG**	**HC**	**HH**	**HE**
	Raw Data															
N	0.018	**0.017**	0.020	0.010	**0.004**	**0.014**	0.018	0.042	Not available							
NN	**0.016**	0.024	**0.015**	**0.005**	0.008	0.018	**0.015**	**0.028**	**0.017**	**0.011**	**0.007**	**0.017**	**0.013**	**0.028**	**0.023**	**0.019**
	Semantic Patterns															
	D=0.0 MDL=1.0								**D=0.0 MDL=1.0**							
σ 0.0	0.017	0.009	0.026	0.012	0.006	**0.017**	0.018	0.031	0.020	**0.005**	0.029	0.017	0.008	0.026	0.019	0.033
σ 0.2	0.014	**0.010**	**0.016**	0.006	0.007	0.018	0.010	0.033	**0.017**	0.007	**0.016**	**0.012**	0.010	0.031	**0.014**	0.032
σ 0.4	**0.017**	0.006	0.023	**0.006**	**0.008**	**0.021**	**0.022**	0.034	0.019	0.009	0.022	0.014	0.010	0.023	0.039	**0.017**
σ 0.6	0.016	0.006	0.026	**0.011**	0.008	0.018	0.017	**0.023**	0.020	0.010	0.024	0.020	**0.006**	0.024	0.020	0.035
σ 0.8	0.015	0.010	0.016	0.014	**0.012**	0.015	**0.015**	0.023	0.020	0.009	0.020	0.018	0.009	**0.022**	0.023	0.040
	D=0.7 MDL=1.0								**D=0.1 MDL=1.0**							
σ 0.0	0.016	0.013	0.013	0.013	**0.006**	0.014	0.019	0.032	0.024	0.016	0.029	0.023	0.010	0.032	**0.024**	**0.033**
σ 0.2	0.015	0.009	**0.011**	**0.006**	0.009	0.011	0.022	0.034	**0.023**	**0.012**	0.023	**0.022**	**0.007**	**0.018**	0.035	0.047
σ 0.4	0.015	0.010	0.017	0.007	0.007	**0.010**	**0.014**	0.040	0.026	0.016	0.022	0.023	0.009	0.027	0.040	0.046
σ 0.6	0.015	0.008	0.021	0.008	0.011	0.017	0.019	**0.022**	0.030	**0.012**	0.032	0.030	0.012	0.039	0.040	0.045
σ 0.8	**0.014**	**0.005**	0.015	0.008	**0.006**	0.020	0.018	0.023	0.028	0.018	**0.017**	0.025	0.017	0.030	0.034	0.056
	D=0.7 MDL=1.5								**D=0.1 MDL=1.5**							
σ 0.0	0.016	0.009	0.017	0.017	0.011	**0.014**	0.025	**0.021**	**0.021**	**0.010**	**0.017**	**0.012**	0.010	0.036	0.032	0.031
σ 0.2	0.015	0.009	0.016	0.014	0.008	0.020	**0.009**	0.027	0.025	0.021	0.019	0.020	**0.009**	0.045	0.028	0.035
σ 0.4	0.017	**0.007**	0.016	0.014	0.010	0.024	0.021	0.027	0.023	0.022	0.026	0.020	0.014	0.030	0.028	**0.022**
σ 0.6	**0.014**	0.010	**0.015**	**0.008**	**0.007**	0.017	0.017	0.022	0.025	**0.010**	0.027	0.023	**0.009**	**0.018**	0.042	0.047
σ 0.8	0.018	0.011	0.020	0.012	0.009	0.020	0.024	0.027	0.025	0.017	0.025	0.033	0.014	0.032	**0.027**	0.028
	D=0.7 MDL=2.0								**D=0.1 MDL=2.0**							
σ 0.0	0.018	0.014	0.014	**0.010**	**0.007**	0.021	0.021	0.040	0.026	**0.009**	0.034	0.029	0.009	0.026	**0.025**	0.047
σ 0.2	0.018	0.022	**0.012**	0.014	0.009	0.017	0.023	0.027	0.026	0.019	0.030	0.018	0.009	0.024	0.039	0.044
σ 0.4	**0.016**	**0.007**	0.028	**0.010**	0.014	0.020	**0.016**	**0.020**	**0.020**	0.012	0.020	**0.017**	0.009	0.024	**0.025**	0.036
σ 0.6	0.018	0.011	0.021	0.018	0.008	**0.015**	0.021	0.031	0.023	0.019	**0.015**	0.029	**0.008**	**0.018**	0.039	**0.033**
σ 0.8	0.022	0.011	0.022	0.014	0.010	0.033	0.018	0.043	0.023	0.018	0.018	0.023	**0.008**	0.024	0.026	0.042
	D=0.7 MDL=3.0								**D=0.1 MDL=3.0**							
σ 0.0	0.019	0.013	0.020	**0.007**	0.010	**0.016**	0.024	0.040	0.025	0.015	**0.019**	0.019	0.013	0.035	0.031	0.046
σ 0.2	0.017	0.014	0.021	0.011	0.011	0.025	**0.007**	0.033	**0.023**	0.019	0.039	0.024	0.011	0.025	**0.020**	**0.022**
σ 0.4	0.020	0.017	0.020	0.019	0.009	0.025	0.013	0.037	0.026	0.019	0.042	0.027	0.014	**0.022**	0.035	0.024
σ 0.6	**0.014**	0.012	**0.012**	0.014	**0.006**	0.019	0.016	**0.018**	**0.023**	0.015	0.023	0.024	**0.010**	0.029	**0.020**	0.039
σ 0.8	**0.014**	**0.010**	0.016	0.011	0.009	**0.016**	0.012	0.027	0.026	**0.013**	0.030	**0.018**	0.011	0.032	0.034	0.043

Table 8.23: Supervised V-Measure standard deviation results for the raw mixed data transformed into *Semantic Patterns*.

SMO											
Par	Total	BW	DI	GL	HS	IO	IR	SE	SO	VE	VO
	Raw Data										
N	**0.521**	**0.781**	**0.178**	**0.296**	0.355	**0.484**	**0.872**	0.879	0.225	**0.514**	0.625
NN	**0.521**	0.777	0.177	0.295	**0.358**	0.477	0.870	**0.880**	**0.230**	**0.514**	**0.628**
	Semantic Patterns										
	No Spreading/1.0										
σ **0.0**	0.538	0.731	0.141	0.424	0.372	**0.489**	**0.871**	0.896	0.280	**0.513**	0.666
σ **0.2**	**0.541**	**0.734**	0.135	**0.432**	0.378	0.483	**0.871**	0.898	**0.301**	0.503	0.670
σ **0.4**	0.540	0.729	**0.145**	**0.432**	0.369	0.475	**0.871**	**0.897**	0.299	**0.513**	0.669
σ **0.6**	0.538	0.738	0.138	0.427	0.371	0.471	**0.871**	**0.897**	0.282	0.510	**0.671**
σ **0.8**	0.540	0.728	0.137	0.431	**0.382**	0.486	**0.871**	0.896	0.286	0.512	0.669
	No Spreading/1.5										
σ **0.0**	0.544	0.730	0.133	0.475	0.346	0.488	0.838	0.901	0.243	**0.507**	0.783
σ **0.2**	0.544	0.734	0.138	0.461	**0.349**	0.507	**0.843**	0.902	0.217	0.506	**0.785**
σ **0.4**	**0.545**	**0.738**	0.134	**0.477**	**0.349**	0.478	0.839	**0.903**	**0.248**	0.502	0.783
σ **0.6**	0.544	0.737	0.135	0.472	0.328	0.495	0.833	**0.903**	0.252	0.506	0.783
σ **0.8**	0.544	0.730	**0.143**	0.470	0.337	**0.508**	0.833	**0.903**	0.233	0.502	0.784
	No Spreading/2.0										
σ **0.0**	0.554	0.736	0.139	0.473	0.262	**0.537**	**0.884**	0.900	0.299	0.501	0.813
σ **0.2**	**0.558**	0.738	0.138	0.484	0.272	0.552	0.882	0.904	0.296	0.504	0.813
σ **0.4**	**0.558**	**0.739**	0.139	**0.487**	0.274	0.534	0.879	**0.906**	**0.303**	0.500	0.815
σ **0.6**	0.557	0.727	0.148	0.483	**0.276**	0.530	0.881	0.902	0.295	**0.510**	0.814
σ **0.8**	0.554	0.732	**0.144**	0.481	0.266	0.520	0.882	0.902	0.291	0.507	**0.816**
	No Spreading/3.0										
σ **0.0**	0.540	0.733	**0.134**	0.466	0.280	0.528	0.864	0.907	0.171	0.527	0.787
σ **0.2**	0.542	0.731	0.130	**0.475**	0.281	0.532	**0.887**	0.905	0.157	0.522	**0.795**
σ **0.4**	**0.545**	0.728	0.132	**0.475**	**0.289**	0.529	0.879	**0.909**	**0.191**	0.532	0.788
σ **0.6**	0.539	0.729	0.128	0.470	**0.289**	**0.535**	0.827	0.908	0.175	0.538	0.789
σ **0.8**	0.537	**0.735**	0.128	**0.475**	0.271	0.533	0.801	0.906	0.184	**0.548**	0.791

SMO											
Par	Total	BW	DI	GL	HS	IO	IR	SE	SO	VE	VO
	Raw Data										
N	**0.012**	**0.007**	**0.005**	0.018	0.015	0.019	**0.012**	**0.001**	0.029	0.008	**0.007**
NN	**0.012**	0.011	0.007	**0.013**	**0.014**	**0.013**	0.018	0.002	**0.021**	**0.007**	0.009
	Semantic Patterns										
	No Spreading/1.0										
σ **0.0**	**0.013**	0.013	**0.010**	0.013	0.018	**0.023**	**0.000**	0.003	0.038	0.009	0.005
σ **0.2**	0.015	0.021	0.011	0.011	0.024	0.025	**0.000**	**0.002**	0.040	0.008	0.007
σ **0.4**	0.014	**0.010**	0.011	0.012	**0.017**	0.028	**0.000**	0.004	0.040	0.007	0.010
σ **0.6**	0.015	0.016	0.011	0.017	0.021	0.036	**0.000**	0.004	**0.030**	0.009	**0.004**
σ **0.8**	**0.013**	0.016	**0.010**	**0.009**	0.018	0.025	**0.000**	0.003	**0.030**	**0.006**	0.009
	No Spreading/1.5										
σ **0.0**	0.018	0.021	0.009	0.016	0.031	0.023	0.014	**0.003**	0.041	0.015	0.009
σ **0.2**	**0.015**	0.023	0.013	**0.012**	0.026	**0.021**	**0.009**	**0.003**	0.027	0.013	**0.005**
σ **0.4**	0.016	**0.019**	0.009	0.015	0.036	0.026	**0.009**	**0.003**	0.029	0.011	0.006
σ **0.6**	0.016	0.021	0.009	0.013	**0.020**	0.027	0.012	0.005	0.030	0.010	0.014
σ **0.8**	0.017	0.022	**0.005**	0.016	0.034	0.037	0.012	0.004	**0.023**	**0.009**	0.006
	No Spreading/2.0										
σ **0.0**	0.016	**0.014**	0.011	**0.017**	0.022	0.030	**0.004**	0.005	0.040	**0.011**	**0.006**
σ **0.2**	0.017	0.018	0.013	0.018	0.027	0.023	0.008	**0.004**	0.038	0.014	0.009
σ **0.4**	**0.015**	0.015	0.013	**0.017**	**0.014**	0.022	0.011	0.005	0.034	0.013	**0.006**
σ **0.6**	0.017	**0.014**	**0.008**	0.019	0.027	0.023	0.013	0.006	**0.033**	0.015	0.007
σ **0.8**	0.016	0.017	0.011	0.023	0.021	**0.019**	0.008	**0.004**	**0.033**	0.014	**0.006**
	No Spreading/3.0										
σ **0.0**	0.016	0.014	0.010	0.013	0.031	0.022	0.022	0.004	**0.016**	0.016	0.009
σ **0.2**	**0.011**	0.018	0.014	**0.009**	**0.013**	**0.015**	**0.008**	**0.003**	0.019	0.011	**0.004**
σ **0.4**	0.017	0.022	0.009	0.020	0.029	0.029	0.014	0.005	0.021	**0.008**	0.009
σ **0.6**	0.016	0.016	0.011	0.022	0.023	0.025	0.014	0.004	0.020	0.012	0.008
σ **0.8**	0.016	**0.012**	**0.006**	0.011	0.017	0.019	0.032	0.005	0.030	0.014	0.010

Table 8.24: Supervised V-Measure results (left) and standard deviation results (right) for the raw numerical data sets and the respective baseline *Semantic Patterns*.

SMO											
Par	Total	BW	DI	GL	HS	IO	IR	SE	SO	VE	VO
	Raw Data										
N	**0.521**	**0.781**	**0.178**	**0.296**	0.355	**0.484**	**0.872**	0.879	0.225	**0.514**	0.625
NN	**0.521**	0.777	0.177	0.295	**0.358**	0.477	0.870	**0.880**	**0.230**	**0.514**	**0.628**
	Semantic Patterns										
	D=0.0 MDL=2.0										
σ **0.0**	0.554	0.736	**0.139**	0.473	0.262	0.537	0.884	0.900	**0.299**	0.501	0.813
σ **0.2**	**0.558**	0.738	0.138	**0.484**	0.272	**0.552**	**0.882**	0.904	0.296	0.504	0.813
σ **0.4**	0.558	0.739	0.139	0.487	0.274	0.534	0.879	0.906	0.303	**0.500**	**0.815**
σ **0.6**	0.557	0.727	0.148	0.483	**0.276**	0.530	0.881	**0.902**	0.295	**0.510**	0.814
σ **0.8**	0.554	0.732	0.144	0.481	0.266	0.520	0.882	0.902	0.291	0.507	0.816
	D=0.5 MDL=1.0										
σ **0.0**	0.537	0.764	**0.144**	0.412	0.383	**0.606**	**0.871**	0.900	**0.229**	0.448	0.609
σ **0.2**	0.536	0.765	0.142	**0.422**	0.386	0.589	**0.871**	0.900	0.221	0.454	**0.612**
σ **0.4**	0.535	0.763	0.135	0.411	0.389	0.603	**0.871**	0.898	0.219	0.454	0.606
σ **0.6**	0.537	0.766	0.139	0.413	**0.400**	0.604	**0.871**	0.899	0.219	0.450	0.609
σ **0.8**	**0.538**	**0.769**	0.139	0.416	0.390	**0.606**	**0.871**	**0.901**	0.227	**0.458**	0.604
	D=0.5 MDL=1.5										
σ **0.0**	**0.552**	0.767	0.149	0.496	0.373	**0.516**	**0.867**	0.900	0.213	**0.489**	0.753
σ **0.2**	**0.552**	**0.772**	**0.151**	0.498	**0.374**	0.497	**0.867**	0.900	0.228	0.481	0.757
σ **0.4**	0.549	0.759	0.144	0.502	0.370	0.500	0.863	**0.901**	0.215	0.479	0.754
σ **0.6**	0.551	0.758	0.142	0.503	0.372	0.493	0.863	**0.901**	**0.239**	0.483	**0.758**
σ **0.8**	**0.552**	0.762	0.140	**0.508**	0.373	0.509	0.859	0.898	0.226	0.488	0.753
	D=0.5 MDL=2.0										
σ **0.0**	0.564	0.754	0.134	0.499	0.326	0.544	**0.901**	0.897	0.307	0.479	0.794
σ **0.2**	0.564	**0.757**	0.140	0.503	**0.330**	0.550	**0.901**	0.897	0.297	0.477	0.791
σ **0.4**	**0.568**	**0.757**	**0.146**	**0.504**	0.327	0.554	**0.901**	**0.898**	**0.318**	0.487	0.792
σ **0.6**	**0.568**	**0.757**	0.144	0.503	0.319	0.554	**0.901**	0.895	**0.318**	**0.495**	**0.795**
σ **0.8**	0.563	0.755	0.141	0.498	0.318	**0.557**	**0.901**	**0.898**	0.283	0.491	0.789
	D=0.5 MDL=3.0										
σ **0.0**	0.536	0.762	0.137	0.469	0.297	0.529	**0.883**	**0.907**	0.146	0.441	0.786
σ **0.2**	**0.539**	0.762	0.137	**0.483**	**0.314**	**0.533**	0.861	0.906	**0.162**	0.451	0.787
σ **0.4**	0.534	**0.763**	0.130	0.471	0.285	0.527	0.856	0.906	**0.162**	0.451	0.786
σ **0.6**	0.535	0.760	0.136	0.467	0.290	**0.533**	0.856	**0.907**	0.151	0.459	**0.791**
σ **0.8**	**0.539**	0.759	**0.141**	0.475	0.305	**0.553**	0.860	0.904	0.149	**0.460**	0.784

SMO											
Par	Total	BW	DI	GL	HS	IO	IR	SE	SO	VE	VO
	Raw Data										
N	**0.012**	**0.007**	**0.005**	0.018	0.015	0.019	**0.012**	**0.001**	0.029	0.008	**0.007**
NN	**0.012**	0.011	0.007	**0.013**	**0.014**	**0.013**	0.018	0.002	**0.021**	**0.007**	0.009
	Semantic Patterns										
	D=0.0 MDL=2.0										
σ **0.0**	0.016	**0.014**	0.011	**0.017**	0.022	0.030	**0.004**	0.005	0.040	**0.011**	**0.006**
σ **0.2**	0.017	0.018	0.013	0.018	0.027	0.023	0.008	**0.004**	0.038	0.014	0.009
σ **0.4**	**0.015**	0.015	0.013	**0.017**	**0.014**	0.022	0.011	0.005	0.034	0.013	**0.006**
σ **0.6**	0.017	**0.014**	**0.008**	0.019	0.027	0.023	0.013	0.006	**0.033**	0.015	0.007
σ **0.8**	0.016	0.017	0.011	0.023	0.021	**0.019**	0.008	**0.004**	**0.033**	0.014	**0.006**
	D=0.5 MDL=1.0										
σ **0.0**	0.012	0.016	0.008	0.007	0.020	**0.018**	**0.000**	0.003	0.027	0.011	0.014
σ **0.2**	0.015	0.019	0.009	0.023	0.019	0.023	**0.000**	**0.002**	0.036	**0.009**	0.008
σ **0.4**	**0.011**	0.014	**0.005**	0.012	**0.014**	0.021	**0.000**	0.004	0.025	0.011	**0.005**
σ **0.6**	**0.011**	**0.011**	0.012	**0.006**	0.018	0.021	**0.000**	**0.002**	**0.018**	0.010	0.012
σ **0.8**	0.012	0.014	0.012	0.013	0.015	0.024	**0.000**	0.003	0.023	0.010	**0.005**
	D=0.5 MDL=1.5										
σ **0.0**	0.014	0.017	**0.006**	0.022	**0.019**	0.024	**0.011**	**0.002**	0.025	**0.007**	**0.008**
σ **0.2**	0.014	0.015	**0.006**	0.018	**0.019**	0.026	**0.011**	0.003	0.023	0.011	**0.008**
σ **0.4**	**0.013**	**0.012**	**0.006**	0.012	0.024	0.022	0.015	0.004	**0.017**	0.011	0.009
σ **0.6**	**0.013**	0.017	0.008	**0.010**	0.024	**0.008**	0.015	0.003	0.021	0.012	**0.008**
σ **0.8**	0.016	0.013	0.011	0.020	0.031	0.019	0.018	0.004	0.023	0.010	0.009
	D=0.5 MDL=2.0										
σ **0.0**	0.014	0.012	0.009	0.019	0.019	0.028	**0.000**	0.004	0.033	0.010	0.009
σ **0.2**	**0.012**	0.014	**0.007**	**0.011**	0.025	**0.016**	**0.000**	**0.003**	0.032	0.010	**0.006**
σ **0.4**	0.013	0.015	0.012	0.017	0.019	0.023	**0.000**	0.004	**0.026**	**0.003**	**0.006**
σ **0.6**	0.015	0.019	0.012	0.019	**0.018**	0.021	**0.000**	0.004	0.034	0.013	**0.006**
σ **0.8**	0.013	**0.009**	0.009	0.012	0.023	0.026	**0.000**	0.005	0.027	0.012	**0.006**
	D=0.5 MDL=3.0										
σ **0.0**	0.014	0.013	0.010	**0.012**	0.024	0.026	0.006	0.005	0.025	0.007	**0.007**
σ **0.2**	0.014	0.015	**0.006**	0.013	**0.022**	0.023	0.012	0.004	0.031	**0.005**	0.009
σ **0.4**	0.014	0.018	0.011	0.024	0.023	0.026	**0.000**	0.004	0.018	0.009	0.011
σ **0.6**	**0.013**	**0.010**	**0.006**	0.016	0.023	**0.017**	0.007	**0.003**	0.030	0.010	0.009
σ **0.8**	0.015	0.014	0.015	0.022	0.023	0.018	0.010	**0.003**	**0.015**	0.017	0.009

Table 8.25: Supervised V-Measure results (left) and standard deviation results (right) for the raw numerical data transformed into *Semantic Patterns*.

9 Related Work

The *Semantic Pattern Transformation* relies on many well-known methods from the areas of machine learning, semantic or associative networks, discretization and spreading activation. These techniques have already been discussed in *Chapter 5 – Techniques*. However, they are not considered as related work, but merely as tools or elements that are required to implement the *Semantic Pattern Transformation*.

In terms of related work, the focus is not so much placed on the required building blocks, but on work that identifies similar problems as discussed in *Chapter 4 – Towards a Semantic Representation*, or work that is based on similar core ideas. The concepts and methods discussed in this chapter within the context of related work include:

- **Machine Learning**: The *Semantic Pattern Transformation* aims to solve many problems that are discussed in the literature whenever machine learning algorithms are employed. The first section of this chapter discusses other work that describes these problems and provides solutions based on various techniques.

- **Latent Semantic Indexing (LSI)**: Similar to the *Semantic Pattern Transformation*, LSI transforms textual data into a semantic representation that can then be analyzed by clustering or semantic-aware search algorithms. This section discusses the main differences between the models employed by LSI and the *Semantic Pattern Transformation*.

- **Measuring node similarities**: Within the context of associative networks, the calculation of the similarities between different network nodes

plays an important role. Similar to the extraction of the *Semantic Patterns* from the trained associative network, the application of spreading activation plays a vital role within these calculations.

- **Calculating the similarity between words**: Finally, the two most important references to related concepts describe the similarity calculation of different English terms by employing associative networks and spreading activation. This work was the initial spark for the idea behind the *Semantic Pattern Transformation* and its application to arbitrary data.

9.1 Machine Learning

The problems related to the value-centric feature vector representation and the required preprocessing operations, which were discussed in the Chapters 4 and 3, are well-known in the literature and typically mentioned whenever machine learning algorithms are applied to an arbitrary problem. A good summary on this topic is given by Kotsiantis et al. [42].

In order to solve these problems, various approaches have been developed that include the application of various preprocessing steps, the development of specific distance-measures capable of handling different data-types, or the transformation of data into another representation that avoids the problems associated with the value-centric feature vector representation. The last approach is the main rationale behind the *Semantic Pattern Transformation*.

The calculation of the similarity between two feature-vectors plays an important role within machine learning algorithms. The usability of such distance-measures depends on the data types contained in the analyzed data set and is discussed in the literature [27]. By taking information theory into account, more sophisticated distance measures based on compression algorithms can be devised [89], [72].

The problem of combining arbitrary data types is also highlighted when looking at machine learning algorithms focused on unsupervised learning. There is a wide range or specific algorithms for symbolic or distance-based data available. Examples for algorithms based on distance-based data are algorithms such as Self-Organizing Maps (SOM) [39], Hierarchical Agglomerative Clustering (HAC), Expectation Maximization (EM), Neural Gas based algorithms [57], [29], [65] or K-Means. In order to cope with symbolic data, existing algorithms have been modified or newly developed: e.g., ROCK [33], COOLCAT [4], or Kernel K-Means [14]. Such techniques typically analyze the co-occurrences of feature values and use this information for unsupervised clustering. A good summary on the different algorithms can be found in [14], where Cuoto et al. introduce the Kernel K-Means algorithm for symbolic data.

In summary, the availability of a wide range of different machine learning algorithms, similarity measures and the in-depth discussion of various preprocessing steps highlight the problems of the value-centric feature vector representation, which the *Semantic Pattern Transformation* aims to solve.

9.2 Latent Semantic Indexing (LSI)

In contrast to the adaptation or creation of distance-measures or new machine learning algorithms, the proposed *Semantic Pattern Transformation* technique focuses on the transformation of the data, before applying standard distance measures and algorithms. Thereby, the transformation process is based on the semantic analysis of the relations between feature values within instances.

This idea is also used by *Latent Semantic Indexing* (LSI) [36], which is widely employed for text related machine learning tasks and is based on the analysis of the semantic relations between terms and documents. Although, the idea behind LSI could be extended and applied to arbitrary data, there are only a few examples in the literature where LSI has been used in other information retrieval domains (e.g., for intrusion detection in a paper by Lassez et al. [51]).

Relation to the *Semantic Pattern Transformation*

LSI employs the mathematical concept of Singular Value Decomposition (SVD) (e.g., [53]) to transform the term-document matrix into a semantic-aware concept space. The dimensionality of this concept space is then reduced by extracting the most important components, which are identified by looking at the largest singular values. In the new representation, the terms contained within the documents are represented as concept space vectors which can be directly used for semantic-aware search algorithms or clustering.

In contrast, the *Semantic Pattern Transformation* first creates the nodes of an associative network that represent the feature values (e.g., terms) of a given data set. The nodes are then linked according to the co-occurrence within the data set instances (e.g., documents). Thereby, symbolic feature values are directly mapped to network nodes, whereas for distance-based feature values a discretization operation is required that maps multiple values to single nodes. Information about feature values (e.g., terms) or instances (e.g., documents) can be extracted by applying spreading activation and extracting the activation values for each node. This information is arranged in a vector – the *Semantic Pattern*. Thus, the elements of a *Semantic Pattern* represent the network activation values for all symbolic and mapped distance-based feature values.

LSI and the *Semantic Pattern Transformation* have both the advantage that not only the feature values, but also their semantic context is taken into consideration. This significantly improves the quality of search and cluster algorithms.

On an abstract level the most significant difference between LSI and the *Semantic Pattern Transformation* is in the employed model. The interpretation of the concept space of LSI is not intuitive and cannot be directly used for extracting the typical characteristics of the analyzed semantic relations between terms and documents. In contrast, the interpretation of a *Semantic Pattern* is trivial: High activation values of network nodes (representing feature values) indicate their high significance within the pattern. Thereby, by sorting the feature values according to their activation values, a quick impression of the pattern can be gained. Furthermore, the intuitive nature of the pattern allows

the simple adaptation for other analysis processes or the extraction of data for further processing. Examples are the incorporation of timestamps and the subsequent extraction of semantic time series of the generated patterns. For an in-depth discussion on the processing and interpretation of the *Semantic Patterns* the reader is referred to *Chapter 7 – Semantic Pattern Analysis*.

Complexity

While the simple model employed by the *Semantic Patterns* enables a significant improvement in terms of flexibility and interpretation, when compared to the LSI concept space, the LSI model has an advantage in terms of complexity. The SVD method employed by LSI offers an intrinsic way of reducing the dimensionality of the semantic concept space by taking the most significant singular values. This dimensionality reduction is highly beneficial for the application of sophisticated procedures like unsupervised clustering.

The dimensionality of the vectors employed by the two different methods is based on the feature values present within the analyzed data. In terms of dimensionality, this corresponds to the bag-of-world model which uses one vector element for each term (or feature value) within the analyzed documents. However, in contrast to LSI, the reduction of these typically high-dimensional vectors is not an intrinsic property of the *Semantic Pattern Transformation* model, and needs to be added by external methods such as the Principal Component Analysis, or other methods that are based on the analysis of the underlying associative networks. However, these procedures have not been applied within the scope of this thesis and will be one of the key areas of future research.

Applications

In current literature, the application of LSI has mostly been limited to the area of text analysis. In contrast, the *Semantic Pattern Transformation* was designed for the application to arbitrary data sets comprised of symbolic and distance-based features. However, there is no theoretical barrier which limits LSI to the application within the text analysis domain. On the contrary, the feature value mapping operations required for the *Semantic Pattern Transformation* could also be utilized to adapt the raw feature vectors, in order to apply LSI for the analysis of arbitrary data sets. In fact, this will be another key area targeted by further research. However, the interpretability of the LSI concept space would still remain a problem and could even become worse due to the combination of arbitrary features.

9.3 Semantic Similarity Between Words

Regarding the application of spreading activation to associative networks and the subsequent extraction of the *Semantic Patterns*, there are two essential papers by Kozima et al. that provided the initial idea for the development of the *Semantic Pattern Transformation*: [43], [44]. In the first paper the authors calculate the

semantic similarity of terms by using the Longman Dictionary of Contemporary English (LDOCE) as knowledge base for extracting semantic relations of English terms. The terms and their relations are represented as nodes and links with a semantic network. In order to calculate the similarity of two terms, the corresponding nodes are activated and the activation is spread to neighboring nodes (terms) via spreading activation techniques. The activation values of other nodes (terms) depend on the strength of the semantic relations between the terms and can be represented in a vector. By calculating the distance between the vectors of two terms, their semantic similarity can be determined. This method is then further refined in their second paper about this topic.

The methods presented in these papers, which were published in 1993 and 1995, are based on similar principles as used within the *Semantic Pattern Transformation*, but their application was limited to textual analysis. The *Semantic Pattern Transformation* significantly extends the initial idea in many ways by adding the capability to analyze data of arbitrary nature, extending the method in order to cover single, multiple feature values, or complete instances, describing multiple operations for the analysis of the generated patterns, and by including advanced spreading activation techniques in order to gain better results. Due to these extensions the *Semantic Pattern Transformation* can be deployed in a wide range of knowledge discovery tasks, which is not possible for the initial application to term similarity presented by the two referenced papers.

9.4 Measuring Nodes Similarities

In order to extract knowledge from large information databases, which often can be organized as networks consisting of nodes and links, analysis methods that explore and query the network structure are needed. Thereby, the network nodes carry some kind of information, whereas the links model the associations or relations between the nodes. Thereby, the emerging and incoming links of the different nodes allow to extract information about the relations between the various concepts represented by the network, and to discover hitherto unknown relations. This knowledge extraction is highlighted by Example 20, which models the relations between words used within different news articles.

Example XX: Is there a similarity between "Tahrir" and "protest"?

When analyzing Tweets related to the early 2011 Egyptian revolution (*Demo Data Set 2*), the extracted terms and their co-occurrences are represented as nodes and links within a network. By looking at the neighborhoods of the two nodes *protest* and *Tahrir*, one can observe that these terms are connected to similar other terms such as *demonstration*, *place*, *Egypt*, or *Mubarak*. This is due to the fact that a large part of the demonstrations of the Egyptian revolution occurred on the Tahrir Place in Cairo. Therefore, the terms *protest* and *Tahrir* are used in a similar context and thus have semantic relations to common terms.

The calculation of such node similarities is discussed in [85], where Thiel et. al investigate the utilization of spreading activation in order to measure the similarity of different nodes within a network. They propose two different similarity measures: The first one is based on the direct and indirect neighbors of nodes, whereas the second one also includes distanced neighborhoods for calculating the similarity.

Definitions

In order to describe the two similarity measures the following definitions (according to [85]) are required: Given a semantic network, which contains a set U of network nodes, an activated node u, and the application of the spreading activation algorithm, then the activation of an arbitrary network node v for iteration k can be calculated via the following equation:

$$a_v(k) = \sum_{u \in N(v)} w(u,v) \cdot a_u^{k-1}$$

Thereby, $w(u,v)$ represents the weighted link between node u and v. The activation values of all nodes v within the network represent the activation vector (pattern) of the network. By arranging the weighted links in a squared matrix W that has a dimension equal to $|U|$, the calculation of an activation vector can be defined as:

$$a^{(k)} = \alpha W a^{k-1}$$

The initial activation of arbitrary nodes in iteration $k = 0$ is represented by $a^{(0)}$ and is the input for the spreading iteration process, which is then applied for various iterations (similar to the processes described in Section 7.3 of *Chapter 7 – Semantic Pattern Analysis*). By summing up the activation vectors of all iterations one *Activation Pattern* is generated that can be compared to patterns generated by other nodes.

$$a = \sum_{k=0}^{k_{max}} \alpha^k \cdot a^{(k)}(v)$$

The changes between the different spreading activation iterations are represented by the velocity vector (for $k > 1$).

$$\delta^{(k)}(v) = a^{(k)}(v) - a^{(k-1)}(v)$$

For $k = 0$ the velocity vector is a null vector. Therefore, the velocity vectors indicate the change of activations during each iteration. The amount of this change can be expressed by applying a norm such as the L2 norm to the velocity vector. According to [6], in each iteration the activation vectors change into the direction of the principal eigenvector of the weight matrix and become equal regardless of the initial activation after a sufficient number of iterations. This means that the amount of change of the velocity vectors will converge to zero after enough iterations.

Activation Similarity

The first node similarity measure introduced by Thiel et. al is based on the similarity of the activation vectors that are caused by the activation of different nodes and the subsequent application of spreading activation.

Given two nodes u and v, their similarity can be calculated by separately activating node u and v in the network, applying spreading activation for k_{max} iterations and extracting the activation vectors $a(v)$ and $a(u)$ from the activated network and then calculating the similarity between $a(v)$ and $a(k)$. Thiel et al. employ the Cosine similarity for this calculation.

Signature Similarity

The second kind of similarity measure is based on the velocity vectors and the amount of change calculated by the L2 norm. Due to convergence to the principal eigenvector of the weight matrix, the amount of change decreases from iteration to iteration. The speed of this process allows to draw conclusions on the position of the activated node and the structure it is embedded. Therefore, Thiel et al. introduce the concept of signature vectors. A signature vector for an activated node v contains the L2 norms for all velocity vectors calculated for each iteration k and thereby gives an indication on the convergence speed of the applied spreading activation process.

$$\tau_k(v) = ||\delta^{(k)}(v)||_2$$

For calculating the similarity between two signature vectors the cosine similarity is employed. According to Thiel et al. the signature vectors reveal structural similarities that cannot be detected via the *Activation Similarity*. However, the concept cannot be used with the *Semantic Pattern Transformation*, because it requires multiple spreading activation iterations.

Relation to the *Semantic Pattern Transformation*

The similarity between *Semantic Patterns* can be calculated by applying the Cosine similarity. This is the same distance measure as employed by Thiel et al. for calculating the similarity of the network nodes' activation and signature vectors.

In respect to the first similarity measure (Activation Similarity), the definitions of the *Semantic Patterns* and the employed activation vectors are practically the same[1]. However, the concept of transforming multiple feature values (e.g instances) into activation vectors and calculating their similarity is not used within the scope of their analysis. Furthermore, for the generation and comparison of *Semantic Patterns* only one spreading activation iteration is employed, whereas Thiel et. al make use of multiple iterations. Multiple iterations have not been utilized for the generation of *Semantic Patterns* due to several reasons: **First**, since the application of spreading activation is a core technique within the *Semantic Pattern Transformation*, the additional computational complexity induced by multiple iterations should be avoided. **Second**, and even more important, due to the high number of interconnections within the networks trained by the *Semantic Pattern Transformation* process, the application of a second iteration typically causes the flooding of the network, which decreases the significance of the semantic information gained during the first iteration.

The second similarity measure discussed by Thiel et al. is based on their concept of node signatures. This concept or a related one is not employed for the *Semantic Pattern Transformation*. Although this method reveals additional information about the network, which cannot be extracted via the Activation Similarity, it requires multiple iterations of the spreading activation process and thus cannot be used within the *Semantic Pattern Transformation*.

9.5 Conclusions

In summary, basic similarities to the ideas and the combination of techniques behind the *Semantic Pattern Transformation* can be found in several references throughout the literature. However, to the best of my knowledge, none of these approaches has been applied as a general transformation process for arbitrary data. Neither does any of these works discuss the integration of a semantic-aware representation within a generic knowledge discovery process, nor the advantages for the interpretation of the results, which are gained when employing a semantic-aware model as represented by the *Semantic Patterns*.

[1] Indeed, the term *Activation Patterns*, which is employed by Thiel et al. was also used during the early development of the *Semantic Pattern Transformation*, because each pattern represents the activation state of the underlying activated associative network. This was later changed to *Semantic Patterns*, which more accurately describes the employed model.

10 Semantic Patterns - Applications

The development of the *Semantic Pattern Transformation* was motivated by the problems associated with the application of knowledge discovery processes based on machine learning in heterogeneous domains. The employed techniques and the choice of their parameters evolved due to the results gained by the empirical evaluations of different applications. Thereby, the initial deployments within text-analysis focused on the basic structure of the transformation process, and the lessons learned have then been used to define more sophisticated analysis processes. This chapter describes the works which were published in the context of the *Semantic Pattern Transformation*, and highlights its evolution process over the last years.

10.1 Analysis Processes

In order to describe the evolution of the *Semantic Pattern Transformation* the following analysis properties, which are also depicted in Figure 10.1 are used for the explanations of the various projects:

- **Distance-based and symbolic features**: This property refers to the nature of the data, which could contain numerical and/or symbolic features. The handling of distance-based features requires the application of discretization procedures that map the feature values to network nodes. This capability was not available in the initial versions of the *Semantic Pattern Transformation*.
- **Unsupervised learning**: Unsupervised clustering was one of the earliest tasks for which the *Semantic Pattern Transformation* was applied. The

first applications included the unsupervised clustering of text-related data consisting of categorical features only. By adding the capability to handle distance-based features, many more application scenarios for unsupervised clustering became available.

- **Supervised learning**: Supervised learning was not applied within the works discussed in this chapter, but played an important role within the performance evaluation conducted in this thesis.

- **Anomaly detection**: The *Semantic Patterns* represent the response of an associative network to arbitrary input-stimuli. Thereby, the amount of the activation energy that is distributed to the neighboring nodes reveals information on whether a given input-stimulus (e.g., an instance describing a country) is considered as an anomaly with respect to the other input-stimuli (e.g., an European country within a subset of Asian countries). The anomaly detection capability was integrated into the later versions of the *Semantic Pattern Transformation*.

- **Semantic search**: Due to the semantic nature of the *Semantic Patterns*, the deployment of a semantic-aware search algorithm is one of the most obvious applications. Since, such an algorithm also relies on the distance-measures used by unsupervised learning algorithms, the capability to execute semantic-aware search queries was already included in the initial versions of the *Semantic Pattern Transformation*.

- **Semantic relations**: Similar to semantic search, the extraction and analysis of semantic relations are also intrinsic capabilities of the *Semantic Patterns* concept, which were already included in the initial deployments. The same concept can always be applied regardless of the analyzed features, and plays an important role for understanding the semantic relations within arbitrary data, especially when no a priori knowledge is available.

- **Feature value relevance**: The weighted links, which connect the different nodes of the associative network enable the extraction of additional information regarding the feature values which were used during the network training. One interesting aspect is the feature value relevance, that can be deduced by measuring the strength of the incoming or outgoing node connections. Similar information also plays an important role for applying advanced spreading activation techniques that employ fanout factors. These advanced strategies were deployed in the later versions of the *Semantic Pattern Transformation*.

- **Time-based analysis**: The inclusion of time-based information shows the high flexibility of the *Semantic Pattern Transformation* and was employed in later stages for the advanced analysis of data extracted from Twitter.

- **Visualization**: The interpretation and visualization of the information contained in the *Semantic Patterns* plays an important part for each

knowledge discovery process. The common model of the *Semantic Patterns*, which remains the same regardless of the analyzed data, is highly beneficial for such processes. In the later applications browser-based visualization and interpretation tools haven been implemented and utilized for the analysis processes.

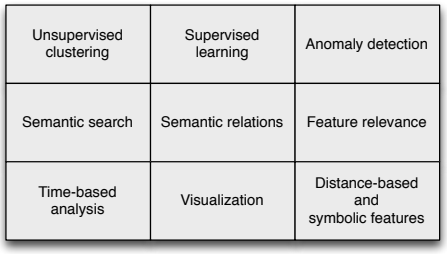

Figure 10.1: Analysis processes and properties of the various applications of the *Semantic Pattern Transformation*.

10.2 Published Works

The *Semantic Pattern Transformation* has been applied in heterogeneous areas and the results have been published in different papers. This section describes the contributions of the different authors, and discusses the different works in the context of the analysis processes discussed in the previous section.

10.2.1 Collaboration

This section focuses on the key authors that were involved in the later described publications.

- **Myself**: The *Semantic Pattern Transformation*, its improvements, and the involved analysis procedures were solely devised and implemented by myself – the author of this thesis.
- **Udo Payer**: Udo Payer was constantly involved in the discussions regarding the security related applications of the *Semantic Pattern Transformation*, and provided vital support and input in the network security related areas, such as event correlation, intrusion detection and shellcode analysis.
- **Günther Lackner**: Guenther Lackner and I collaborated on many WiFi security related issues before the development of the *Semantic Pattern Transformation*. In fact, the topic of his PhD thesis [49] focuses on privacy related issues within WiFi networks, and we found the opportunity apply the *Semantic Pattern Transformation* within the context of his work. Thereby, his focus was placed on the WiFi related issues, while I dealt with

the knowledge discovery related analysis processes. Günther also collaborated on other publications where he provided important feedback on the quality of the results and the utilized methods.

- **Reinhard Fellner:** Reinhard Fellner applied the early version of the *Semantic Pattern Transformation* within his master's thesis [26]. Thereby, among other applications, he analyzed data related to event correlation within intrusion detection, and provided crucial feedback for the improvement of the *Semantic Pattern Transformation*. The results were also published in a paper [83], which is described later.

10.2.2 Works in Direct Relation to the *Semantic Pattern Transformation*

The following publications represent the evolution of the *Semantic Pattern Transformation*. The empirically gained results were used to improve the employed techniques and parameters, and to devise new analysis processes.

Automated Analysis of e-Participation Data by Utilizing Associative Networks, Spreading Activation and Unsupervised Learning [84]

This paper describes the first application of the *Semantic Pattern Transformation* to text-related data extracted from the Austrian e-Participation project Mitmachen.at [84]. The e-Participation process allows citizens to express their opinions or participate in decision processes by electronic means. In many cases, this process involves the creation of text by the participants that reflects their opinions, raises new topics for discussions, or argues for a topic based on various reasons. These documents can then be analyzed by domain experts who extract information or knowledge, which is beneficial for the whole decision process. Unfortunately, this is a time-consuming process that requires the manual interaction of domain experts. Thus, automated analysis methods based on machine learning and clustering play an important role within such processes.

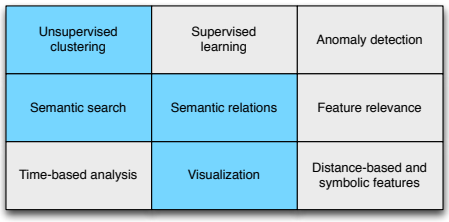

Figure 10.2: Analysis processes and properties within the context of e-Participation related text analysis.

In order to test the first version of the *Semantic Pattern Transformation*, we transformed the available text data into *Semantic Patterns* – called *Activation*

Patterns at this time – and applied various analysis procedures. Together with the environment, this corresponds to the following properties (depicted in Figure 10.2) that were defined in Section 10.1:

- **Symbolic features**: The analyzed data was based on features extracted from documents in German. These features were German words and thus of symbolic or categorical nature. At this time the *Semantic Pattern Transformation* was not capable of analyzing distance-based features due to the lack of the required discretization operations.
- **Unsupervised clustering**: The transformed patterns were then subject to the application of unsupervised clustering algorithms from the Neural Gas family (Section 5.7.3 of *Chapter 5 – Techniques*). Here, the first empirical analysis of the gained results was executed and the influence of various parameters has been determined. The lessons learned were then used to fine-tune the *Semantic Pattern Transformation*.
- **Semantic relations**: One of the major benefits of the *Semantic Pattern Transformation* is the simple interpretation of the *Semantic Patterns*. Due to the well-known features (German terms) the analysis of the e-Participation related documents provided us with a perfect opportunity to test how well knowledge could be extracted from the generated patterns.
- **Semantic search**: Due to the semantic nature of the patterns, the direct application of semantic search algorithms is possible. Here, the capability to perform semantic search queries was tested by using the semantic fingerprints of terms and sentences to retrieve semantically related concepts.

The first results gained by the initial analysis procedures were very promising and caused the further improvement of the *Semantic Pattern Transformation* by including more advanced capabilities.

Event Correlation on the Basis of Activation Patterns [83]

Event correlation plays an important role in intrusion detection in the broad field of network security. Thereby, the data collected by various sensors is used for the detection of attacks or anomalous behavior. The data generated by the sensors is of a heterogeneous nature and includes symbolic as well as distance-based features. In [83], we apply the *Semantic Patterns* concept to the raw feature vectors comprised of different data types, and apply standard unsupervised clustering algorithms for further analysis.

After successfully applying the *Semantic Pattern Transformation* to the symbolic e-Participation data, the focused was placed on the integration of distance-based features. In order to allow this, some kind of discretization algorithms needs to be applied to the raw feature values, which are then mapped to the nodes of the associative network. This procedure was tested by shifting the focus from the text-related data to event correlation data collected by a network-based

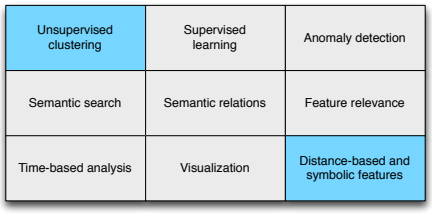

Figure 10.3: Analysis processes and properties within the context of event correlation.

intrusion detection system. Here, symbolic features like protocol identifiers were combined with distance-based features like the number of connections per time frame. After applying the transformation procedure, the focus was based on the unsupervised clustering of the different attacks contained within the data set (Figure 10.3):

- **Symbolic and distance-based features**: The analyzed data set contains both symbolic and distance-based features. Therefore, the necessary discretization procedures have been integrated within the *Semantic Pattern Transformation* and evaluated within the context of this analysis.
- **Unsupervised clustering**: The well-known data set provided us with the opportunity to evaluate the performance of the *Semantic Pattern Transformation* when both symbolic and distance-based features are involved. In order to visualize the clusters representing the different attacks within the analyzed data set, the unsupervised *Self-Organizing Maps* algorithm has been applied. The visualization capabilities of this algorithm allowed us to gain feedback for the various discretization strategies and associated parameters, which are necessary for modeling the distance-based features within the associative network.

Due to the analysis of this heterogeneous data set, the capability to analyze symbolic and/or distance-based features was added to the *Semantic Pattern Transformation*. By evaluating the results of unsupervised clustering, it was possible to selected the best discretization operations.

From NLP (Natural Language Processing) to MLP (Machine Language Processing) [81]

In order to test the *Semantic Pattern Transformation* and the capability for the simple interpretation of the *Semantic Patterns*, the whole concept was applied to a completely different domain where – compared to text analysis – not so much a priori knowledge was available: The analysis of the assembler code of polymorphic shellcodes. Shellcodes are used for the exploitation of buffer overflows and have evolved during the last 20 years from simple machine learning

code that can easily be detected by intrusion detection systems, to very sophisticated polymorphic and metamorphic structures that continuously change by employing encryption and other sophisticated measures. Therefore, machine learning plays an important role for the creation of advanced detection methods and understanding the various categories of such shellcodes. Due to our prior work in the area [64], we came up with the idea to combine the lessons learned form Natural Language Processing within the Mitmachen project, and the capabilities of the *Semantic Pattern Transformation* for the analysis of assembler code extracted from polymorphic shellcodes.

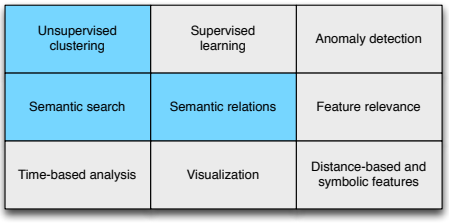

Figure 10.4: Analysis processes and properties within the context of polymorphic shellcode analysis.

In principal, the same procedure as in the Mitmachen project were applied, but here data was analyzed for which only a limited amount of priori knowledge was available. Thus, the capability to interpret the generated patterns could be evaluated within this domain (Figure 10.4):

- **Symbolic features**: The assembler instructions were used as symbolic features, which corresponds to the analysis of terms within text-analysis.
- **Unsupervised clustering**: Again, unsupervised clustering algorithms were applied to automatically categorize the different polymorphic engines, which are used to camouflage the actual shellcode.
- **Semantic relations**: This was the key focus of this analysis. Due to the limited a priori knowledge, the extraction of meaningful semantic relations plays a vital role. Within this context, the relations between the assembler instructions employed by the different polymorphic engines could be extracted and visualized. These results allowed us to gain a better understanding of the methods employed by the various engines.
- **Semantic search**: The semantic search capabilities helped us to find assembler instructions that were used within a similar context – e.g., within similar decryption engines that decrypt the actual polymorphic shellcode. In addition, semantic search queries for semantically related assembler instruction chains were deployed and analyzed.

By focusing on a data set that contains not so well-known data and relations, the alleged capabilities in terms of knowledge extraction and interpretation of the *Semantic Pattern Transformation* could be evaluated.

User Tracking Based on Behavioral Fingerprints [50]

The PhD thesis of Günther Lacker [49] focuses on the privacy aspects of public WiFi networks. Due to prior work, we found another opportunity for the application of the *Semantic Pattern Transformation* here. There are many features and properties that might reveal one's identify and thereby the location when multiple public WiFi hotspots of the same provider are used throughout the time. In order to gain a better understanding of this data, and the relations between the different features, the *Semantic Pattern Transformation* has been applied to information extracted from captured email messages. While this data is not always available, it provided us with a good opportunity to test the knowledge extraction capabilities of the *Semantic Patterns*. As depicted in Figure 10.5, the same analysis processes as for the e-Participation data and the polymorphic shellcodes, were applied.

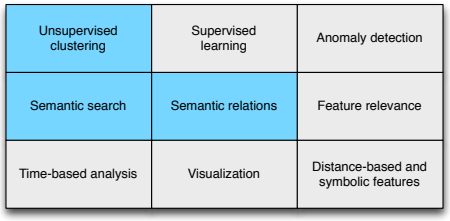

Figure 10.5: Analysis processes and properties within the context of WiFi privacy.

RDF Data Analysis with Activation Patterns [79], and Knowledge Extraction from RDF Data with Activation Patterns [80]

The semantic web – a web of data – utilizes rich semantic links between concepts that are machine-readable. This enables advanced search queries and the possibility to link data from various sources. The SPARQL language enables the extraction of arbitrary aspects of such data sets.

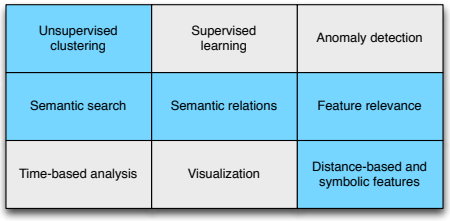

Figure 10.6: The World Factbook investigations

For the analysis presented in [79] and [80], we have used data extracted from an RDF version of *The World Factbook* [11]. This data set was very important

for improving techniques and the selection of parameters, because it contains diverse data types. The following analysis processes were applied (Figure 10.6):

- **Distance-based and symbolic features**: The description of the countries contain many symbolic and distance-based features. Furthermore, the information contained in these features and the semantic relations between the different feature values are well-known. Thus, the gained results can easily be verified.
- **Unsupervised clustering**: Unsupervised clustering algorithms were applied to the country instances and to the single feature values. This allowed us to gain a quick overview of the different countries. Again, a verification of the validity of these clusters could easily be done due to the well-known features and feature values.
- **Semantic relations**: Due to the well-known features and semantic relations, the validity of the information contained in the *Semantic Patterns* could easily be verified.
- **Semantic search**: Here, the semantic search algorithm was improved for two application scenarios: **First**, by using complete country instances as search query, and **second** by executing typical search queries based on one or a few specific feature values.
- **Feature relevance**: This was the first application of the feature relevance analysis process. By analyzing the structure of the associative network, the relevance of the different feature values can be determined. Examples for non-relevant feature values include the different currencies within certain groups of countries, the export commodities of countries located on specific continents, or in certain cases the spoken languages.

The application to this data set allowed us to fine-tune the different aspects of the *Semantic Pattern Transformation*, and test how the relevance of feature values could be determined by analyzing the structure of the associative network.

Android Market Analysis With Activation Patterns [78]

In [78] we have analyzed the security permissions along other metadata of roughly 130.000 apps of the Android Market. By applying the *Semantic Pattern Transformation* to the categorical data, we were able to gain a new understanding of the applications through extracting knowledge about security permissions, their relations and possible anomalies, executing semantic search queries, finding relations between the description and the employed security permissions, or identifying clusters of similar apps.

Thereby, the following analysis processes have been conducted (Figure 10.7):

- **Symbolic features**: The relevant features were of symbolic nature and included application permissions, decryption terms, application categories and symbolic download count ranges.

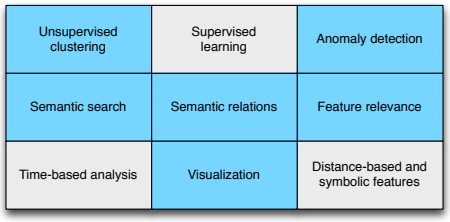

Figure 10.7: Analysis processes and properties within the context of the Android Market.

- **Unsupervised clustering**: Unsupervised clustering was applied to gain a quick understanding of the analyzed applications, and the different feature values. The latter refers to the clustering of single permissions and terms in order to determine in which semantic context they are used.

- **Semantic relations**: Here, the semantic relations and permissions were of special interest, because they allowed us to gain a detailed understanding of the typical permission usage within Android applications.

- **Semantic search**: Here, semantic search queries containing complete application instances and single feature values were executed. In the first case, semantically similar applications could be retrieved. The second application allowed us to specify certain feature values within the search query (e.g., terms) and retrieve semantically related concepts (e.g., permissions).

- **Feature relevance**: Here, the relevance of certain feature values was determined. E.g., the permission to allow the utilization of the GPS component is not relevant in applications that are related to car navigation systems.

- **Anomaly detection**: This analysis played a very important role, since it allows the extraction of applications that are described by non-typical features within a given application group. E.g., an application that describes itself as a simple wallpaper, but has permissions that allow the determination of the user's location, or reading her address book, is considered as anomaly within the wallpaper application group.

This is a very important practical application of the *Semantic Pattern Transformation* that has a high relevance to the current malware developments on the smartphone market. Therefore, it will also be considered as a hot topic for future work.

Extracting Semantic Knowledge From Twitter [77]

Twitter – also called the SMS of the Internet – currently has more than 200 million users and processes roughly 140 million tweets a day[1] – meaning there is a huge amount of arbitrary information available for knowledge discovery processes. Since a tweet is limited to 140 characters, the information a user wants to convey must be compressed, which reduces the computational effort required by automated analysis processes. For the analysis processes within the *Semantic Pattern Transformation*, we have extracted Tweets related to the early 2011 revolution in Egypt. Here, also the timestamps of the tweets were modeled within the associative network, which allowed us to automatically extract relevant events, analyze their semantic evolvement over time and automatically find semantic similar events that occur on other timestamps. In addition, a user interface was developed that demonstrates the interpretation of the *Semantic Patterns*, and can easily be extended for arbitrary knowledge discovery processes.

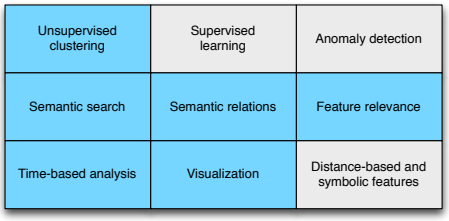

Figure 10.8: Analysis processes and properties within the context of Twitter.

Thereby, the following analysis processes have been conducted:

- **Symbolic features**: The features were based on the terms extracted from Tweets related to the Egyptian revolution, the retrieved hashtags and the timestamps of the Tweets. Especially, the combination of timestamps with the terms and hashtags features allows the extraction of information that is related to semantic relations between events that occur at different times. The inclusion of the timestamps also highlights the flexibility of the model employed by the *Semantic Patterns*.

- **Unsupervised Clustering**: Here, unsupervised clustering was applied to automatically find categories for the analyzed tweets and their features.

- **Semantic relations**: Similar to the other applications, the semantic relations between tweets and the extracted features could be extracted. However, due to the integration of time-based information, we were also able to analyze the semantic relations between timestamps and terms that were used within the different Tweets.

[1] As of March 2011, based on http://blog.twitter.com/2011/03/happy-birthday-twitter.html, accessed on December 15th 2011.

- **Semantic search**: Based on the semantic relations between terms, hashtags and timestamps, we were able to execute search queries similar to the previous examples, but also search queries that contained time-based information. By including this information, it was possible to retrieve events that occur at different timestamps but have a semantic similarity.
- **Time-based analysis**: For this analysis, the timestamps of the tweets have been included as features used within the *Semantic Pattern Transformation*. This allows for further analysis processes that analyze the semantic relations between timestamps and other feature values, or instances (Tweets). Examples for such analysis processes are: semantic-aware search queries that retrieve semantically related timestamps, the clustering of timestamps that co-occur with similar events and the semantic development of events over time.
- **Visualization**: Here, a browser-based user interface was developed that allows the graphical analysis and interpretation of the *Semantic Patterns*. Due to the nature of the employed model, that is independent of the analyzed data, a generic interface can be deployed for arbitrary knowledge discovery problems and data.

The integration of the time-based information and the development of a user interface that allows to interpret arbitrary *Semantic Patterns* present the latest important mile stones within the *Semantic Pattern Transformation*, and form the basis for future work.

10.2.3 Precursors to the *Semantic Pattern Transformation*

The works discussed in this section precedes the publications related to the *Semantic Pattern Transformation*. We have worked on multiple knowledge discovery domains primarily related to e-Participation and IT security. The requirements for specific algorithms and preprocessing operations for each domain-specific knowledge discovery problems were the main reasons for developing the *Semantic Pattern Transformation*. The foundations for this development were laid in the following prior works:

- **Hybrid Engine for Polymorphic Shellcode Detection [64]**: This work summarizes the first part of my master's thesis [76] and describes how continuously changing polymorphic or metamorphic shellcodes used to exploit buffer overflows can be detected in the network stream by applying artificial neural networks. Thus, the knowledge discovery relevant part of this work is related to the training and deployment of supervised machine learning algorithms.
- **Traffic Classification Using Self-Organizing Maps [62]**: This paper describes the second part of my master's thesis, which focused on the machine learning based identification of application layer protocols such as

HTTP, HTTPS, or TELNET by inspecting the transmitted data. Here the knowledge discovery relevant part was related to protocol classifiers implemented via supervised machine learning algorithms and the application of unsupervised algorithms for discovering unknown relations within the analyzed data.

- **Massive Data Mining for Polymorphic Code Detection [63]**: Here, additional machine learning based methods related to polymorphic shellcode detection are discussed.
- **WiFi Chipset Fingerprinting [46]**: The lessons learned from polymorphic shellcode detection and network traffic analysis were then employed for the identification of WiFi chipsets based on timing characteristics.
- **InFeCT - Network Traffic Classification [82]**: The previously described network traffic classification algorithms were extended in this work by further machine learning methods such as the *Support Vector Machines* algorithm.

10.2.4 This Thesis

The previous mentioned publications have applied the *Semantic Pattern Transformation* within a wide range of knowledge discovery domains. By empirically validating the results and utilizing the learned lessons for the improvement of the *Semantic Pattern Transformation*, the method has been constantly improved. However, the detailed explanation of the concept, the relations to and between knowledge discovery and machine learning, and the thorough evaluation have not been described within these applications, and therefore are the main subject of this thesis.

10.3 Chapter Conclusions

This chapter describes the scientific publications that are directly related to the *Semantic Pattern Transformation*, or are considered as preliminary work that provided the initial motivation for its development. The evolution of the new method has been described by explaining how each publication helped to improve the transformation process and the subsequent analysis of the generated patterns. Thereby, the new concept has been applied within heterogeneous knowledge discovery domains. This thesis is considered as the final building block, that first gives an in-depth description of the whole concept and presents a thorough evaluation of the performance within unsupervised clustering, supervised learning and semantic-aware search algorithms.

11 Conclusions and Outlook

Due to the huge amount of electronically available data, and the wide spread requirement to extract information and relations from this data, knowledge discovery mechanisms are considered as a vital instrument within academic and industrial processes. Thereby, these methods aim to analyze unknown data, extract relevant information and discover hitherto unknown relations. Among the wide variety of different knowledge discovery processes, machine learning plays an important role. However, when such algorithms are deployed in heterogeneous domains, the associated preprocessing steps need to be adapted or even redefined according to the nature of the data and the desired knowledge discovery goals. Furthermore, the interpretation of the gained results strongly depends on the employed machine learning algorithms, and the analyzed data. This leads to the requirement for a time-consuming setup of the deployed procedures, whenever the data or the knowledge discovery goals change.

Due to the lessons learned from the application of machine learning algorithms in a wide range of heterogeneous domains, and the analysis conducted in this thesis, one key reason for the discussed problems was identified: The value-centric feature vector representation, which is typically used within machine learning.

Therefore, this thesis proposes the *Semantic Pattern Transformation* that transforms this value-centric representation into a semantic representation – the *Semantic Patterns*. Due to the analysis of the semantic relations between the feature values, the presented transformation process removes the need for many time-consuming setup procedures. Furthermore, the model employed by the *Semantic Patterns* is independent of the analyzed data and, thus, simplifies the otherwise complex knowledge extraction and interpretation procedures. Another benefit of the common model is the easy extension of existing analysis processes,

or the integration of additional procedures without the need for specific adaptations.

For the development of the *Semantic Pattern Transformation* a wide range of empirical evaluations have been conducted, and the process has been constantly improved by studying the lessons learned. This is shown by the works published in relation to the *Semantic Pattern Transformation*, which represent the evolution of the concept. These publications range form the initial clustering of categorical data, over the integration of semantic search algorithms, to the application of sophisticated analysis processes applicable to an arbitrary combination of categorical and numerical features.

This thesis extends the empirical evaluations by conducting an in-depth analysis of the *Semantic Pattern Transformation* within the domains of supervised and unsupervised machine learning, and semantic-aware search algorithms. Thereby, the following observations are made: The quality of the results gained by simple distance-based unsupervised learning algorithms like *K-Means* can be significantly improved when the value-centric feature vectors are transformed into *Semantic Patterns*. Also, in the area of supervised learning the results of the *Support Vector Machine* algorithm indicate slight quality improvements. For the semantic-search aware algorithm significant quality improvements can be achieved, when using the *Semantic Patterns*. These observations lead to the following conclusions: **First**, the quality of the results gained by simple distance-based supervised and unsupervised algorithms can be significantly improved by employing the *Semantic Pattern Transformation*. **Second**, regardless of the applied supervised or unsupervised algorithm, the gained results are at minimum at the same level, and in most of the cases even better as for the raw data. This means, that the huge benefits gained by the semantic model can be utilized for machine learning without making compromises in terms of quality. **Finally**, the results gained for the semantic-aware search algorithms show the Euclidean distance and the Cosine similarity achieve the same results when complete instances are analyzed. However, the performance of the Euclidean distance significantly drops in comparison to the Cosine similarity, when only a few feature values are used within the search queries.

The results gained by this evaluation process and the nature of the *Semantic Patterns* lead to the following main contributions of this thesis:

1. **Improvements for machine learning algorithms**: The quality of the results gained by simple unsupervised machine learning algorithms can be significantly improved, while it is approximately the same for more sophisticated ones. For the supervised algorithms a slight quality increase can be observed, and huge gains are made for the search algorithms.

2. **Deployment in heterogeneous domains**: The *Semantic Pattern Transformation* analyzes the semantic relations between the feature values and transforms the value-centric feature vectors into *Semantic Patterns*. This transformation leads to a significant decrease in the complexity of the adaptation processes required when knowledge discovery processes are deployed in heterogeneous domains.

3. **Improved analysis processes**: The model employed by the *Semantic Patterns* can easily be interpreted and visualized. This plays an important role when analyzing data for which no, or only limited a priori knowledge is available. Furthermore, the common model and its simple interpretation allows for the application of simple techniques, such as addition or subtraction, as well as highly sophisticated analysis, such as machine learning. Furthermore, due to the employed model that is independent of the analyzed data, new analysis processes can easily be defined and deployed.

Although, many aspects haven been analyzed and explained within this thesis and the published works, there remain several questions and research directions for the further improvement and extension of the *Semantic Pattern Transformation*. These include the application of mechanisms for reducing the patterns' dimensionality in order to decrease the required computational complexity, the development of improved visualization and interpretation techniques based on the *Semantic Patterns*, and the focus on further analysis methods such as anomaly detection, or time-based analysis. Another advanced and interesting topic is how the *Semantic Patterns* could be used in multi-hierarchical learning scenarios, where the *Semantic Patterns* gained by different analysis for one level are considered as feature values that are used as input for the *Semantic Pattern Transformation* of the next level.

A

Demonstration Data Sets

Throughout this thesis, examples have been presented that are based on the two data sets described in this appendix. The first one contains the descriptions of the world's countries as extracted from an RDF representation of the "The World Factbook" published by the CIA [11]. The second data set consists of Tweets extracted from Twitter that are related to the Egyptian revolution in early 2011. While the first data set plays the most important role within this thesis, the second one is especially used in the context of time-based analysis of *Semantic Patterns*.

A.1 Demo Data Set 1 - The World Factbook

The first data is based on an RDF representation of the "The World Factbook" [11]. This representation contains the descriptions of 261 countries and territories, and was generated in 2005 by Ben Humphreys[1], who provides various Perl scripts to parse the data from HTML pages and convert it into an RDF presentation. Thereby, only a subset of the original features have been extracted. These features are listed in Table A.1.

The data set played an important role during the development and improvement of the *Semantic Pattern Transformation* for the following reasons: **First**, the data set contains a good mixture of symbolic and distance-based features. **Second**, the distance-based features are based on different value ranges and value types: ratios, percentage values, absolute counts etc., which was important for improving the *Semantic Pattern Transformation*. **Finally**, and probably

[1] http://www.aktors.org/interns/2005/cia/CIA World Factbook to RDF.htm, accessed on December 15th 2011.

Feature	Type	Description
Export	sym	export commodities
Import	sym	import commodities
EnvAgree	sym	environmental agreements
Agree	sym	international agreements
AgrProd	sym	agricultural products
Language	sym	spoken language
Resource	sym	natural resources
Feature	**Type**	**Description**
UnemploymentRate	%	unemployment rate
LiteracyFemale	%	literacy rate of females
LiteracyMale	%	literacy rate of males
MilitaryGDP	%	percentage of GDP used for military spending
GDP-Agriculture	%	contribution to GDP by agricultural sector
GDP-Industry	%	contribution to GDP by industrial sector
GDP-Services	%	contribution to GDP by service sector
GDP-PerCapita	$	GDP per capita in US dollars
BirthRate	real	births/1000 population
DeathRate	real	deaths/1000 population
PopBelowPovertyLine	%	percentage of population below poverty line
PopGrowthRate	%	population growth rate
InflationRate	%	inflation rate

Table A.1: Features contained within the first data set.

the most important reason: The contained data is well-known and understood without any additional information. Therefore, it played an important part in the empirical evaluation of the initial versions of the *Semantic Pattern Transformation*. The plausibility and correctness of the gained results could easily be checked without deploying complex evaluation schemes.

The data set was analyzed in two works: *RDF Data Analysis with Activation Patterns* [79], and *Knowledge Extraction from RDF Data with Activation Patterns* [80]. A summary on these works is given in *Chapter 10 – Applications*.

A.2 Demo Data Set 2 - The Egyptian Revolution on Twitter

The second data set was extracted from Twitter by using the now shut down Google Realtime service. It contains 3712 Tweets covering the beginning of the Egyptian revolution in 2011. The data set starts at the beginning of the uprising on January 25th and ends with the resignation of then president Hosni Mubarak. The employed features are summarized in Table A.2. In order to avoid noise and limit the amount of information, the following procedure has been applied to the Tweets: Only the raw text was extracted and analyzed by applying well-known techniques from Natural Language Processing such as stop-word removal, Part-

Feature	Type	Description
Term	sym	Nouns, verbs and adjectives of a tweet after applying NLP processing
Hashtag	sym	The hashtags of a tweet
TimeStamp	sym	Timestamp of a tweet

Table A.2: Features contained within the second data set.

Of-Speech tagging and finding the base form on the extracted terms. In addition to the terms, the hashtags encountered in the Tweets and the timestamps of the tweets were also extracted. The timestamps played an important role for the time-based semantic analysis of the data, which was mentioned in *Chapter 7 – Semantic Pattern Analysis*. The data set was thoroughly analyzed within [77] – *Extracting Semantic Knowledge from Twitter*. A summary on this work is given in *Chapter 10 – Applications*.

Bibliography

[1] H. Allende, C. Rogel, S. Moreno, and R. Salas. Robust Neural Gas for the Analysis of Data with Outliers. *Proceedings 24th International Conference of the Chilean Computer Science Society*, pages 149–155, 2004.

[2] S. Anand. A Data Mining Methodology for Cross-Sales. *Knowledge-Based Systems*, 10(7):449–461, May 1998.

[3] S. S. Anand and A. G. Buchner. *Decision Support Using Data Mining*. Trans-Atlantic Publications, 1998.

[4] D. Barbará, Y. Li, and J. Couto. COOLCAT: An Entropy-Based Algorithm for Categorical Clustering. In *Proceedings of the eleventh International Conference on Information and Knowledge Management - CIKM '02*, page 582, New York, New York, USA, Nov. 2002. ACM Press.

[5] A. Barron, J. Rissanen, and B. Yu. The Minimum Description Length Principle in Coding and Modeling. *IEEE Transactions on Information Theory*, 44(6):2743–2760, 1998.

[6] M. R. Berthold, U. Brandes, T. Kötter, M. Mader, U. Nagel, and K. Thiel. Pure Spreading Activation is Pointless. *CIKM 2009*, pages 1915–1918, 2009.

[7] J. C. Bezdek. *Pattern Recognition with Fuzzy Objective Function Algorithms*. Plenum Press, 1981.

[8] C. M. Bishop. *Pattern Recognition and Machine Learning*, volume 4 of *Information Science and Statistics*. Springer, 2006.

[9] M. Browne. Cross-Validation Methods. *Journal of Mathematical Psychology*, 44(1):108–132, 2000.

[10] P. Cabena, Hadjnian, Stadler, Verhees, and Zanasi. *Discovering Data Mining: From Concept to Implementation*. Prentice Hall, 1997.

[11] CIA. The World Factbook, 2011.

[12] K. J. Cios, W. Pedrycz, R. W. Swiniarski, and L. A. Kurgan. *Data Mining: A Knowledge Discovery Approach*, volume 1. Springer, 2007.

[13] C. Cortes and V. Vapnik. Support-Vector Networks. *Machine Learning*, 20(3):273–297, 1995.

[14] J. Couto. Kernel K-Means for Categorical Data. *Lecture Notes in Computer Science*, 3646, 2005.

[15] F. Crestani. Application of Spreading Activation Techniques in Information Retrieval. *Artificial Intelligence Review*, 11(6):453–482, 1997.

[16] R. N. Dave and R. Krishnapuram. Robust Clustering Methods: A Unified View. *IEEE Transactions on Fuzzy Systems*, 5(2):270–293, 1997.

[17] David Deutsch. *The Beginning of Infinity: Explanations That Transform the World*. Viking Adult, 2011.

[18] F. Dellaert. The Expectation Maximization Algorithm. *Technology*, 2(February):1–7, 2002.

[19] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum Likelihood from Incomplete Data Via EM Algorithm. *Journal of the Royal Statistical Society Series BMethodological*, 39(1):1–38, 1977.

[20] Döring Christian, Kruse Rudolf, Timm Heiko, and Borgelt Christian. Fuzzy Cluster Analysis with Cluster Repulsion. In *Proceedings of the 1st International Workshop on Hybrid Methods for Adaptive Systems (EUNITE 01)*, 2001.

[21] J. Dougherty, R. Kohavi, and M. Sahami. Supervised and Unsupervised Discretization of Continuous Features. In *Machine Learning: Proceedings of the Twelfth International Conference*, pages 194–202. Morgan Kaufmann Publishers, Inc., 1995.

[22] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification*, volume 2 of *Pattern Classification and Scene Analysis: Pattern Classification*. Wiley, 2001.

[23] P. Earle. Earthquake Twitter. *Nature Geoscience*, 3(4):221–222, 2010.

[24] U. Fayyad, G. Piatetsky-Shapiro, and P. Smyth. From Data Mining to Knowledge Discovery in Databases. *AI Magazine*, 17(3):37–54, 1996.

[25] C. Fellbaum. *WordNet: An Electronic Lexical Database*. Language, speech, and communication series. MIT Press, 1998.

[26] R. Fellner. *Ereignisskorrelation mit Assoziativen Netzen*. Master's thesis, Graz University of Technology, 2008.

[27] H. Finch. Comparison of Distance Measures in Cluster Analysis with Dichotomous Data. *Journal of Data Science*, 3(1):85–100, 2005.

[28] A. Frank and A. Asuncion. UCI Machine Learning Repository, 2010.

[29] B. Fritzke. A Growing Neural Gas Learns Topologies. *Advances in Neural Information Processing Systems*, (1):1211–1216, 2005.

[30] Q. Gao, J. Yan, and M. Liu. A Semantic Approach to Recommendation System Based on User Ontology and Spreading Activation Model. In *Proceedings of the 2008 IFIP International Conference on Network and Parallel Computing*, pages 488–492. IEEE Computer Society, 2008.

[31] R. M. Gray. Vector Quantization. *IEEE ASSP Magazine*, 1(2):4–29, 1984.

[32] M. Grinberg, V. Haltakov, and H. Stefanov. Approximate Spreading Activation for Efficient Knowledge Retrieval from Large Datasets. In *Proceedings of the 2011 conference on Neural Nets WIRN10, Proceedings of the 20th Italian Workshop on Neural Nets*, pages 326–333. IOS Press, 2011.

[33] S. Guha, R. Rastogi, and K. Shim. Rock: A Robust Clustering Algorithm for Categorical Attributes. *Information Systems Journal*, 25(5):345–366, 2000.

[34] G. Hamerly and C. Elkan. Learning the K in K-Means. *Engineering*, 17:1–8, 2003.

[35] K. Heller and Z. Ghahramani. Bayesian Hierarchical Clustering. *Neuroscience*, 22(section 2):297–304, 2005.

[36] T. Hofmann. Probabilistic Latent Semantic Indexing. In *Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '99, pages 50–57. ACM, 1999.

[37] Y.-J. Hu and D. Kibler. Generation of Attributes for Learning Algorithms. pages 806–811, Aug. 1996.

[38] P. J. Huber. *Robust Statistics*, volume 82 of *Wiley Series in Probability and Statistics*. Wiley, 1981.

[39] T. Kohonen. *Self-Organizing Maps*, volume 30 of *Springer Series in Information Sciences*. Springer, 2001.

[40] S. Kotsiantis and D. Kanellopoulos. Discretization Techniques: A Recent Survey. *GESTS International Transactions on*, 32(1):47–58, 2006.

[41] S. B. Kotsiantis. Supervised Machine Learning: A Review of Classification Techniques. *Informatica*, 31:249–268, 2007.

[42] S. B. Kotsiantis, D. Kanellopoulos, and P. E. Pintelas. Data Preprocessing for Supervised Leaning. *Journal of Computer Science*, 1(2):111–117, 2006.

[43] H. Kozima and T. Furugori. Similarity Between Words Computed by Spreading Activation on an English Dictionary. *Proceedings of the sixth Conference on European Chapter of the Association for Computational Linguistics*, pages 232–239, 1993.

[44] H. Kozima and A. Ito. Context-Sensitive Measurement of Word Distance by Adaptive Scaling of a Semantic Space. In *Proceedings of RANLP-95*, number 52, pages 161–168. John Benjamins Publishing Company, 1995.

[45] L. A. Kurgan and P. Musilek. A Survey of Knowledge Discovery and Data Mining Process Models. *The Knowledge Engineering Review*, 21(01):1, 2006.

[46] G. Lackner, M. Lamberger, U. Payer, and P. Teufl. WiFi Chipset Fingerprinting. In P. Horster, editor, *DACH Mobility 2006*, pages 1–13, 2006.

[47] G. Lackner, U. Payer, and P. Teufl. Combating Wireless LAN MAC-Layer Address Spoofing with Fingerprinting Methods. *International Journal of Network Security*, 9(2):164–172, 2009.

[48] G. Lackner and P. Teufl. IEEE 802.11 Chipset Fingerprinting by the Measurement of Timing Characteristics. In *Proceedings of the Australasian Information Security Conference 2011 AISC11*, 2011.

[49] G. Lackner and P. Teufl. *Security and Privacy Aspects of Wireless Computer Networks*. PhD thesis, University of Technology Graz, Austria, 2011.

[50] G. Lackner, P. Teufl, and R. Weinberger. User Tracking based on Behavioral Fingerprints. In *Proceedings of the The Ninth International Conference on Cryptology And Network Security CANS 2010*, 2010.

[51] J.-L. Lassez, R. Rossi, S. Sheel, and S. Mukkamala. Signature Based Intrusion Detection Using Latent Semantic Analysis. In J. Wang, editor, *2008 IEEE World Congress on Computational Intelligence*, pages 1068–1074. IEEE Computational Intelligence Society, IEEE Press, 2008.

[52] D. Li. Understanding Latent Semantic Indexing : A Topological Structure Analysis Using Q-Analysis. *Journal of the American Society for Information Science*, 61(2005):592–608, 2010.

[53] S. Lipovetsky. PCA and SVD with Nonnegative Loadings. *Pattern Recognition*, 42(1):68–76, 2009.

[54] J. MacQueen. Some Methods for Classification and Analysis of Multivariate Observations. In L. M. L. Cam and J. Neyman, editors, *Proceedings of the fifth Berkeley Symposium on Mathematical Statistics and Probability*, volume 233, pages 281–297. California, USA, University of California Press, 1967.

[55] N. Malik. Artificial Neural Networks and their Applications. *IEEE PES special publication*, page 6, 2005.

[56] S. Markovitch and D. Rosenstein. Feature Generation Using General Constructor Functions. *Machine Learning*, 49(1):59–98, 2002.

[57] T. Martinetz and K. Schulten. A "Neural Gas" Network Learns Topologies. In T. Kohonen, K. Mäkisara, O. Simula, and J. Kangas, editors, *Artificial Neural Networks*, pages 397–402. Elsevier Science Publishers B.V., 1991.

[58] J. Morato, M. A. Marzal, J. Lloréns, and J. Moreiro. WordNet Applications. *Processing*, pages 270–278, 2004.

[59] M. Needleman. RDF - The Resource Description Framework. *Serials Review*, 27(1):58–61, 2001.

[60] C. Orthacker, P. Teufl, S. Kraxberger, A. Marsalek, J. Leibetseder, and O. Prevenhueber. Android Security Permissions Can we trust them? In *MobiSEC 2011*, 2011.

[61] N. R. Pal, L. Jain, K. J. Cios, and L. A. Kurgan. *Advanced Techniques in Knowledge Discovery and Data Mining*. Advanced Information and Knowledge Processing. Springer London, London, 2005.

[62] U. Payer, M. Lamberger, and P. Teufl. Traffic Classification using Self-Organizing Maps. In *INC 2005 5th International Networking Conference Workshops Samos Island Greece*, 2005.

[63] U. Payer, P. Teufl, S. Kraxberger, and M. Lamberger. Massive Data Mining for Polymorphic Code Detection. *Third International Workshop on Mathematical Methods Models and Architectures for Computer Network Security MMMACNS 2005*, 3685:448–453, 2005.

[64] U. Payer, P. Teufl, and M. Lamberger. Hybrid Engine for Polymorphic Shellcode Detection. In *Lecture Notes in Computer Science*, volume 3548, pages 19–31, 2005.

[65] A. K. Qin and P. N. Suganthan. Robust Growing Neural Gas Algorithm with Application in Cluster Analysis. *Neural Networks*, 17(8-9):1135–1148, Oct. 2004.

[66] M. R. Quillian. Semantic Memory. In M. Minsky, editor, *Semantic Information Processing*, volume 2, chapter 10, pages 227–270. MIT Press, 1968.

[67] J. R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann series in {M}achine {L}earning. Morgan Kaufmann, 1993.

[68] J. Rissanen. *Stochastic Complexity in Statistical Inquiry*, volume 15 of *Series in computer science ; vol. 15*. World Scientific, 1989.

[69] A. Rosenberg and J. Hirschberg. V-measure: A Conditional Entropy-Based External Cluster Evaluation Measure. *Computational Linguistics*, 1(June):410–420, 2007.

[70] G. K. Saha. OWL Web Ontology Language. *Ubiquity*, 2007(September):1–1, 2004.

[71] D. Scanfeld, V. Scanfeld, and E. L. Larson. Dissemination of Health Information Through Social Networks: Twitter and Antibiotics. *American Journal of Infection Control*, 38(3):182–8, 2010.

[72] D. Sculley and C. E. Brodley. Compression and Machine Learning: A New Perspective on Feature Space Vectors. *Data Compression Conference DCC06*, 0:332–332, 2006.

[73] C. Shearer. The CRISP-DM Model: The New Blueprint for Data Mining. *Journal of Data Warehousing*, 5(4):13–22, 2000.

[74] J. Shlens. A Tutorial on Principal Component Analysis. *Measurement*, 51(10003):52, 2005.

[75] J. F. Sowa. Semantic Networks. *Encyclopedia of Artificial Intelligence*, 5(3):291–9, 1992.

[76] P. Teufl. *Intrusion Detection and Traffic Classification based on Machine Learning Techniques*. Master's thesis, Graz University of Technology, 2005.

[77] P. Teufl and S. Kraxberger. Extracting Semantic Knowledge from Twitter. *IFIP International Federation For Information Processing*, pages 48–59, 2011.

[78] P. Teufl, S. Kraxberger, C. Orthacker, G. Lackner, A. Marsalek, J. Leibetseder, and O. Prevenhueber. Android Market Analysis with Activation Patterns. In *MobiSEC 2011*, Aalborg, 2011.

[79] P. Teufl and G. Lackner. RDF Data Analysis with Activation Patterns. In H. Maurer and K. Tochtermann, editors, *Proceedings of the 10th International Conference on Knowledge Management and Knowledge Technologies iKNOW 2010 Graz Austria*, Journal of Computer Science, 2010.

[80] P. Teufl and G. Lackner. Knowledge Extraction from RDF Data with Activation Patterns. *J. UCS*, 17(7):983–1004, 2011.

[81] P. Teufl, G. Lackner, and U. Payer. From NLP (Natural Language Processing) to MLP (Machine Language Processing). In I. Kotenko and V. Skormin, editors, *Proceedings of the Mathematical Methods Models and Architectures for Computer Networks Security Conference 2010 MMMACNS 2010 St Petersburg Russia*, volume 6258 of *Lecture Notes in Computer Science*, pages 256–269. Springer Berlin Heidelberg, 2010.

[82] P. Teufl, U. Payer, M. Amling, M. Godec, S. Ruff, G. Scheikl, and G. Walzl. InFeCT - Network Traffic Classification. *Proceedings of the seventh International Conference on Networking ICN 2008*, pages 439–444, 2008.

[83] P. Teufl, U. Payer, and R. Fellner. Event Correlation on the Basis of Activation Patterns. In *Proceedings of the 18th Euromicro Conference on Parallel Distributed and NetworkBased Processing PDP 2010*, pages 631–640, 2010.

[84] P. Teufl, U. Payer, and P. Parycek. Automated Analysis of e-Participation Data by Utilizing Associative Networks, Spreading Activation and Unsupervised Learning. In *Proceedings of the 1st International Conference on Electronic Participation (EPART 09)*, volume 5694, pages 139–150. Springer-Verlag, 2009.

[85] K. Thiel and M. R. Berthold. Node Similarities from Spreading Activation. *2010 IEEE International Conference on Data Mining*, pages 1085–1090, 2010.

[86] A. Troussov, D. Parra, and P. Brusilovsky. Spreading Activation Approach to Tag-aware Recommenders: Modeling Similarity on Multidimensional Networks. In *Proceedings of the ACM Conference on Recommender Systems*, 2009.

[87] A. Troussov, M. Sogrin, J. Judge, and D. Botvich. Mining Socio-Semantic Networks Using Spreading Activation Technique. In *International Workshop on Knowledge Acquisition from the Social Web KASW08*, 2008.

[88] G. Tsatsaronis, M. Vazirgiannis, and I. Androutsopoulos. Word Sense Disambiguation with Spreading Activation Networks Generated from Thesauri. In M. M. Veloso, editor, *IJCAI*, pages 1725–1730, 2007.

[89] P. M. B. Vitanyi, F. J. Balbach, R. L. Cilibrasi, and M. Li. Normalized Information Distance. *Information Theory and Statistical Learning*, cs.IR:33, 2008.

[90] I. H. Witten, E. Frank, and M. A. Hall. *Data Mining: Practical Machine Learning Tools and Techniques, Third Edition (The Morgan Kaufmann Series in Data Management Systems)*. Morgan Kaufmann, 2011.

[91] Y. Yang and G. I. Webb. A Comparative Study of Discretization Methods for Naive-Bayes Classifiers. In *Proceedings of PKAW*, volume 2002, pages 159–173, 2002.

[92] T. Zesch, C. Müller, and I. Gurevych. Extracting Lexical Semantic Knowledge from Wikipedia and Wiktionary. In N. Calzolari, editor, *Linguistics*, pages 1646–1652, 2008.

[93] T. Zesch, C. Müller, and I. Gurevych. Using Wiktionary for Computing Semantic Relatedness. In *Proceedings of AAAI*, volume 8, pages 861–866. AAAI Press, 2008.

[94] C. N. Ziegler and G. Lausen. Spreading Activation Models for Trust Propagation. In *Proceedings of the IEEE International Conference on e-Technology, e-Commerce, and e-Service*, EEE '04, pages 83–97. IEEE, 2004.

Deutsche Fassung:
Beschluss der Curricula-Kommission für Bachelor-, Master- und Diplomstudien vom 10.11.2008
Genehmigung des Senates am 1.12.2008

EIDESSTATTLICHE ERKLÄRUNG

Ich erkläre an Eides statt, dass ich die vorliegende Arbeit selbstständig verfasst, andere als die angegebenen Quellen/Hilfsmittel nicht benutzt, und die den benutzten Quellen wörtlich und inhaltlich entnommene Stellen als solche kenntlich gemacht habe.

Graz, am …… ……..
(Unterschrift)

Englische Fassung:

STATUTORY DECLARATION

I declare that I have authored this thesis independently, that I have not used other than the declared sources / resources, and that I have explicitly marked all material which has been quoted either literally or by content from the used sources.

…… ……..
date (signature)