Knowledge Discovery from Time Series

Inaugural-Dissertation

zur

Erlangung des Doktorgrades der

Mathematisch-Naturwissenschaftlichen Fakultät

der Heinrich-Heine-Universität Düsseldorf

vorgelegt von

Tim Schlüter

aus Hilden

Düsseldorf, April 2012

Aus dem Institut für Informatik
der Heinrich-Heine Universität Düsseldorf

Gedruckt mit der Genehmigung der
Mathematisch-Naturwissenschaftlichen Fakultät der
Heinrich-Heine-Universität Düsseldorf

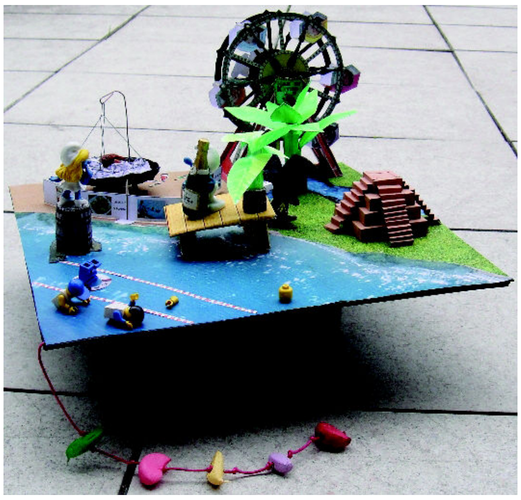
Referent: Prof. Dr. Stefan Conrad
Koreferent: Priv.-Doz. Dr. Frank Gurski

Tag der mündlichen Prüfung: 26.4.2012

Danksagung

Als erstes möchte ich meinem Doktorvater Prof. Dr. Stefan Conrad danken. In den fünf Jahren, in denen ich an seinem Lehrstuhl für Datenbanken und Informationssysteme an der Heinrich-Heine-Universität in Düsseldorf als wissenschaftlicher Mitarbeiter gearbeitet habe, hat er mir ein überaus angenehmes Klima für meine Tätigkeiten in Forschung und Lehre geboten. Unter seiner fachkundigen Betreuung habe ich sehr viel gelernt und ist letztendlich diese Doktorarbeit entstanden.

Mit „Schuld“ an diesem angenehmen Klima waren natürlich auch meine lieben Kollegen, denen ich an dieser Stelle danken möchte: Katrin Zaiß und Sadet Alcic, die die längste Zeit meine Weggefährten auf dem Weg zur Promotion waren, Guido Königstein, unseren spitzen Admin, der sich so oft um meine Pflanzen kümmern musste, Sabine Freese und Marga Potthoff, ohne die ich mich nur sehr schwer im administrativen Dschungel der Universität zurecht gefunden hätte, meine beiden temporären „Mitbewohner“ Thomas Scholz und Ludmila Himmelspach, Magdalena Rischka, Robin Küppers, Jiwu Zhao, Daniel Braun, Michael Singhof und – last but not least – unserem „Externen“ Oliver Daute, der mich zum Kauf meiner tollen neuen GPS-Uhr überzeugt hat (... läuft). Das Bild auf der Rechten zeigt den wunderschönen Hut, den mir meine Kollegen zur bestandenen Prüfung gebastelt habt... vielen Dank auch dafür!



Selbstverständlich möchte ich auch meinen Eltern Klaus und Margret danken, die mir das Studium ermöglicht und mich jederzeit unterstützt haben. Für ihre Geduld und ihr Verständnis insbesondere in den letzten Tagen danke ich auch meiner Freundin Janina.

Allen anderen, die sich an dieser Stelle nicht wiederfinden, sei gesagt, dass ich sie nicht vergessen habe; allerdings ist jetzt auch gerade die Seite voll...

Düsseldorf, Germany
April, 2012 *Tim Schlüter*

Abstract

Nowadays, organizations of diverse areas are collecting several kinds of data, which results in a huge bulk of data that possibly contains useful information. Since this amount of data is far too big for manual analysis, algorithms for semi-automatically discovering potential useful information within this data are developed, which is the main subject of the research area *Knowledge Discovery in Databases.*

Among the different kinds of data, from which knowledge can be discovered, *Time Series* represent an especially challenging kind, since they contain interesting temporal particularities which have to be regarded separately. Analyzing time series with respect to these temporal particularities can analogously be denoted as *Knowledge Discovery from Time Series*, which is the main issue of this work.

In order to provide the background for this thesis, we first introduce and provide a detailed review of knowledge discovery in databases in general and time series analysis in particular. After that, we introduce our contributions and integrate them into the area of *Knowledge Discovery from Time Series.*

The first two contributions concern the subarea temporal association rule mining, which aims at analyzing transactional data with temporal information (which can be regarded as complex time series) in order to find associations within this data. Here, we introduce TARGEN, a market basket dataset generator which models several temporal coherences (which is thus ideal for testing new temporal association rule mining algorithms), and a tree-based approach for mining several kinds of temporal association rules at once.

We transfer standard and temporal association rule mining techniques, which were originally designed for transactional data, to the analysis of elementary time series, and present a concrete approach for mining such standard and temporal association rules from a time series database, which for instance can be used for predicting future values of time series.

In addition to that, we present two further approaches for time series analysis and prediction, which use a Hidden Markov Model basing on inter-time-serial correlations discovered by using derivative dynamic time warping and a novel motifs-based time series representation.

Finally, we present two approaches applying time series analysis to concrete problems in linguistics and medicine, namely approaches for measuring text similarity and automatic sleep stages scoring.

Zusammenfassung

In fast allen Bereichen des Lebens fallen heutzutage große Datenmengen an, die von Organisationen gesammelt werden, um darin potentiell nützliches, bisher jedoch noch unbekanntes Wissen zu finden, das man möglicherweise gewinnbringend einsetzen kann. Da die Menge an Daten zu groß für eine manuelle Analyse ist, werden Algorithmen entwickelt, die semi-automatisch versuchen sollen, potentiell nützliches Wissen aus diesen Daten zu extrahieren. Die Entwicklung dieser Algorithmen ist Hauptgegenstand des Forschungsgebiets *Knowledge Discovery in Databases* (zu Deutsch: *Wissensentdeckung in Datenbanken*). In der Vielzahl der möglichen Daten, anhand derer konkretes Wissen gewonnen werden kann, stellen *Zeitreihen* eine besondere Herausforderung dar, da sie interessante zeitliche Besonderheiten aufweisen, die gesondert betrachtet werden müssen. Die Analyse von Zeitreihen im Hinblick auf diese zeitlichen Besonderheiten kann analog als *Knowledge Discovery from Time Series*, also als *Wissensentdeckung aus Zeitreihen*, bezeichnet werden, was das Thema dieser Arbeit ist.

Zu Beginn behandelt diese Dissertation die erforderlichen Grundlagen aus den Bereichen Wissensentdeckung in Datenbanken im Allgemeinen und der Analyse von Zeitreihen im Speziellen. Daraufhin werden die einzelnen Beiträge dieser Arbeit vorgestellt und in das Gebiet der *Wissensentdeckung aus Zeitreihen* eingeordnet. Im Einzelnen zählen die ersten beiden Ansätze zum Teilbereich der zeitlichen Assoziationsanalyse, bei der transaktionale Daten mit zeitlichen Informationen (welche als komplexe Zeitreihen aufgefasst werden können) im Hinblick auf zeitliche Assoziationen untersucht werden. Die beiden konkret vorgestellten Ansätze sind TARGEN, ein Generator für Warenkorbdaten, der verschiedene zeitliche Zusammenhänge innerhalb der Daten erzeugen kann (weshalb er sich hervorragend zum Testen von neuen Algorithmen zur zeitlichen Assoziationsanalyse eignet), und ein baumbasierter Ansatz, der verschiedene Arten von zeitlichen Assoziationsregeln auf einmal entdecken kann.

Im Weiteren zeigt diese Arbeit, wie Techniken der normalen und zeitlichen Assoziationsanalyse, die ursprünglich nur für transaktionale Daten entwickelt wurden, auf normale Zeitreihen übertragen und angewendet werden können. Mit Hilfe der so gefunden normalen und zeitlichen Assoziationen kann z.B. Zeitreihenvorhersage betrieben werden. Zur Analyse und Vorhersage von Zeitreihen werden zwei weitere Ansätze vorgestellt, die beide ein Hidden Markov Model benutzen, das Zusammenhänge zwischen Zeitreihen modelliert, und Derivative Dynamic Time Warping bzw. eine neuartige Motiv-basierte Zeitreihenrepräsentation um korrelierte Zeitreihen zu finden.

Schließlich werden noch zwei Ansätze präsentiert, die Techniken der Zeitreihenanalyse anwenden, um konkrete Probleme aus den Bereichen Linguistik und Medizin zu lösen; bei dem einen Ansatz handelt es sich um das Messen von Textähnlichkeiten, bei dem anderen um automatische Schlafphasenanalyse.

Contents

1 Introduction

This introductory chapter motivates *Knowledge Discovery in Databases* in general and *Knowledge Discovery from Time Series*, which is the main issue of this PhD thesis, in particular. The contributions of this PhD thesis to these areas are listed in section 1.2, followed by section 1.3, which presents the organization of this work.

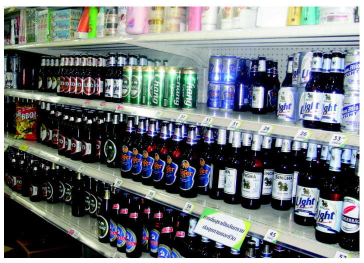
1.1 Motivation

Advances in storage and media technology achieved in the past decades have led to rapidly increasing sizes of storage media and falling memory prices. The resulting increased storage availability together with advances in data collection techniques enable organizations of many different areas to collect huge amounts of data, either for processing, archiving, or other purposes. These amounts of data possibly contain previously unknown information and coherences, of which the usage could be very valuable for these organizations. Since the amounts of data are far too big for manual analysis, algorithms for (semi-)automatically discovering such potentially useful knowledge in large data repositories are developed. This very active research area is called *Knowledge Discovery in Databases* (KDD).

KDD can be defined as "non-trivial process of identifying valid, novel, potentially useful, and ultimately understandable patterns in data" [FPSS96]. It is an iterative and interactive process, which involves several steps and decisions being made by a user according to the underlying data and the purpose of the knowledge discovery. Naturally, the first step is the identification of the concrete knowledge discovery objective. After that, steps like data selection, preprocessing and transformation are applied, before concrete patterns are mined from the data. This central step of the whole KDD process is called *Data Mining*. Since in other literature data mining is also defined as "process of automatically discovering useful information in large data repositories" [TSK05] or similar, data mining and KDD are often used synonymously. The last step of the KDD process is the evaluation and interpretation of the discovered patterns/knowledge, where the user is involved. Decisions like whether derived patterns are the desired ones can lead to a complete or partial restart of the KDD process with adapted parameters in order to reveal knowledge useful for the certain domains.

A concrete knowledge discovery objective and data mining task is the discovery of association rules from transactional data, which is the main issue of the subarea *Association Rule Mining*. An association rule captures associations between sets of items in form of an implication expression $\{antecedent\} \Rightarrow \{consequent\}$, which expresses that the occurrence of a certain set of items in a transaction denoted by $\{antecedent\}$ also implies the occurrence of the second set $\{consequent\}$ in that particular transaction. Association rules can be used for descriptive and predictive purposes in several areas, e.g. in weblog analysis (for determining user groups and maximizing user satisfaction [NES+11]), in utilization analysis (providing sufficient electrical power supply or selecting larger aircraft types at certain times or days [BLT+10]), and in market basket analysis. Since market basket analysis is one of the most popular applications of association rule mining, example 1 clarifies the discovery and possible usage of association rules from a supermarket scenario. Two other very popular knowledge discovery objectives are *Clustering* and *Classification*, which aim at assigning data objects to certain clusters or classes. (More information about steps involved in the KDD process and concrete objectives can be found in chapter 2, which introduces and explains KDD in detail.)

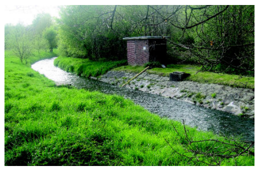
Example 1. Association Rules in Market Basket Analysis



As indicated by the name, market basket analysis analyzes market baskets with the goal of discovering associations between items. In a typical supermarket scenario, these items are products, which can be bought by a customer. At the end of his shopping tour, the customer has to pay for all products in his shopping basket at the cash register, which stores several pieces of information about all transaction (like bought items, prices, timestamp of transaction, and so on). The analysis of these transactions can reveal interesting coherences; an example for such a coherences could be expressed by the statement "customers, who buy beer, often buy potatoes crisps as well". In market basket analysis, such coherences are typically captured in form of association rules; for the example mentioned above as $\{beer\} \Rightarrow \{potato\ crisps\}$ *with* *a certain probability. The knowledge derived from a particular association rule can be used in order to describe customer behavior or to predict future purchase behavior, and it can be applied in several ways; one of them is product placement: in order to make shopping more comfortable for the customers, the owner of a supermarket can place beer and potato crisp directly next to each other. Or on the contrary, he can place these products far apart from each other: since he knows that many people tend to buy both beer and potato crisp, they will either walk from beer to the crisps section or vice versa, and on their way they will pass and see several other products or even cleverly placed (maybe pretended) "special offers", which they normally would not notice. By choosing such market layout, customers can be seduced to buy products, which they normally would not buy, what increases the profit of the supermarket owner.*

Among the huge amounts of data, which are collected nearly everywhere and every day, *Time Series* represent a special kind of data with very interesting temporal particularities. In the simplest case a time series is a sequence of values, which are measured at certain (typically successive) points in time. These values are either stored in sequences alone (e.g. in case of equidistant points in time when the domain is clear) or together with timestamps (which denote the time of measurement/occurrence). Examples for the occurrences of elementary time series can be found in geology (e.g. derived from sensor measurements [Lan12]), in medical diagnosis (from EEG and other measurements [GAG$^+$00, Fre12]), in finance (stock prices [Yah12]), in a more complex form in marketing (transactional data[1] [BSVW99, KBF$^+$00, ZKM01]), and in diverse other areas [CC08]. The analysis of time series is the main subject of *Time Series Analysis*, which is also a subarea of KDD. Deriving knowledge from this analysis can be denoted as *Knowledge Discovery from Time Series*, which is the main topic of this thesis. The following example 2 illustrates time series derived from river level measurements and their analysis with regard to possible objectives of knowledge discovery from these time series. (Further objectives and more details about the analysis of time series in general can be found in chapter 3.)

Example 2. Time Series derived from River Levels and their Analysis



Water levels are measured by different organizations for different purposes. The environmental agency of the German federal state North Rhine-Westphalia (NRW) for instance measures river levels of most of the major rivers in and around NRW, and publishes these measurements online [Lan12]. These river levels are of interest for several domains, e.g. for floodwater warning and custody, boat transportation, and sports like canoeing (every river has a minimal water

[1]A transactional dataset can be seen as sequence of high dimensional data points, where every transaction corresponds to one data points. The components of these data points store all pieces of information of the transactions (e.g. timestamps, customer IDs and bought items).

level, under which traveling on this river is forbidden - among others due to environmental protection). River level measurements naturally induce a time series for each river level station, which consists of river levels measured at this station, times of measurements and potentially further information. Several analyses can be conducted on these time series in order to derive certain knowledge; among others these time series can be clustered (in order to find similar behaving rivers), they can be classified into predefined river classes (e.g. according to flood danger categories or boat transportation suitability), and models can be constructed in order to predict future river levels (for instance using Hidden Markov Models). Here, the particular denotable part is the fact, that all these analyses are merely performed on the time series derived from the river level measurements without using additional domain knowledge (such as geographic location or special river characteristics).

The whole area of KDD can be categorized in several ways. One of them is according to the underlying data, thus time series analysis can be seen as subarea of KDD, which merely deals with elementary low dimensional time series data. From another point of view, where time series are not limited to elementary low dimensional time series, but to all kinds of data with temporal information, time series analysis can be regarded as synonym for *Temporal Data Mining* [LS06]. Temporal data mining denotes data mining with special regard to the temporal component, i.e. it deals with data containing temporal information. For completely revealing all occurring phenomenons, this kind of data has to be viewed as sequence of points or events [AO01], thus it can be seen as time series. A concrete temporal data mining or time series analysis task is *Temporal Association Rule Mining* (TARM), which aims at discovering special temporal coherences in timestamped transactional data. An example of a temporal association rule from market basket analysis is the rule $\{\text{coffee}\} \Rightarrow \{\text{newspaper}\}$, which could be valid at a kiosk or gas station only in the morning hours. The knowledge derived from this rule could be used for instance in order to boost the sales volume by offering special coffee and newspapers bundles in these hours. (Details about temporal association rule mining can be found in chapter 4.)

This PhD thesis integrates and explicates several approaches for time series analysis in the broader sense, which can be summarized under the title “Knowledge Discovery from Time Series” (where time series refer to all datasets containing temporal information as discussed before). Most of these approaches, which are listed in detail in the following section 1.2 as own contributions to the area of time series analysis, have been developed at the Institute for Databases and Information Systems of the Department of Computer Science at the Heinrich-Heine-University in Düsseldorf, where the author has been working five years from April 2007 until March 2012 as research assistant.

1.2 Contributions

After introducing and providing a detailed review of KDD in general and time series analysis in particular in order to provide the background for this thesis, the following chapters present our contributions to the area of time series analysis. This includes two concrete approaches for TARM, an approach for applying TARM to the analysis of elementary time series, two approaches for predicting future values of time series containing simple values, and two practical applications of time series analysis to problems in linguistics and medicine (measuring text similarity and automatic sleep stages scoring).

In detail, the main contributions are

- TARGEN, a dataset generator which creates timestamped transactional datasets that contain temporal coherences (TARGEN provides an excellent basis for testing and evaluating temporal association rule mining algorithms; it has already been proposed in [SC09].),
- a tree-based approach for mining several kinds of temporal association rules from transactional data (presented in [Sch08, SC10b]),
- the transfer from standard and temporal association rule mining techniques for transactional data to time series (containing simple values) and a concrete approach for mining these rules in time series databases (which can be used e.g. for predicting future values of these time series; published in [SC11a, SC11b]),
- approaches for predicting elementary time series using a Hidden Markov Model and inter-time-serial correlations (basing on time series distances [SC10c] and a novel motifs-based time series representation [SC12b]), and
- two approaches for solving practical problems by applying time series analysis to linguistics and medicine, namely an approach for measuring text similarity [MSC08] (e.g. for detecting plagiarism in different languages) and an approach for automatic sleep stages scoring[2] [SKC10, SC10a, SC12a].

The following section outlines the concrete organization of the whole thesis.

[2]Sleep stage scoring is an essential step in sleep analysis; up to now it is mostly done manually by medical experts.

1.3 Organization

This work is organized as follows. Chapter 2 presents the background of knowledge discovery in databases, i.e. basics, different steps towards the knowledge discovery, an overview of data mining tasks and objectives, and in more detail the subarea association rule mining. Time series, their analysis and the knowledge discovery from time series, which is in the main focus of this thesis, are introduced in chapter 3.

Chapter 4 introduces temporal association rule mining and presents one of our first contributions, namely TARGEN, a dataset generator for testing temporal association rule mining algorithms, and a tree-based approach for mining different kinds of temporal association rules. The application of temporal association rule mining to time series analysis, which can be used for instance for predicting future value of time series, is presented in chapter 5. Chapter 6 contains further approaches for time series analysis with the purpose of time series prediction (using a Hidden Markov Model basing on time series distances and time series motifs), followed by chapter 7, which applies time series analysis to concrete problems in linguistics and medicine (measuring text similarity and automatic sleep stages scoring). Conclusions and directions for future work of our approaches presented in chapters 4 to 7 are given at the end of the according sections and chapters.

Finally, after section 7 a list of all our publications in chronological order is given at the end of this thesis, followed by references, and lists of figures and tables.

2 Knowledge Discovery in Databases

Nowadays, organizations of diverse areas are collecting several kinds of data, which results in a huge bulk of data that possibly contains usual information. Since this amount of data is far too big for manual analysis, algorithms for semi-automatically discovering potential usual information within this data are developed. The area of research dealing with the discovery of such information from large databases is called *Knowledge Discovery in Databases* (for short: KDD), which will be introduced and explicated in this chapter.

KDD is an interdisciplinary area settled at the intersection of several research fields, which provide different techniques and influences: techniques from statistics for instance have their main focus on numerical data and model-based interferences. Artificial intelligence, pattern recognition and machine learning are mainly dealing with the extraction of patterns, search procedures and symbolic data. The accessibility on large data sources, on which the analyses are performed, is handled and provided by techniques from database systems. KDD can be seen as the branch of information science, which is mainly focusing on the automated discovery of knowledge (instead of focusing on the manual discovery), although user interaction still is absolutely necessary (e.g. to determine whether extracted information is subjectively useful).

The basic process of KDD is presented in section 2.1. Section 2.2 gives an overview of the concrete data mining techniques, which are used in order to extract knowledge for the different purposes. *Association Rule Mining*, which is one of the two most interesting data mining tasks for this thesis, is presented in section 2.3 in detail. (Due to its size and meaning for this thesis, the second data mining of great interest for this thesis is presented in an extra chapter: details about *Time Series Analysis* can be found in chapter 3.)

2.1 The Process of KDD

In their seminal work from 1996, Fayyad, Piatetsky-Shapiro, and Smyth have introduced a unifying framework for knowledge discovery and data mining, which will be presented in the following. According to [FPSS96], *Knowledge Discovery in Databases* can be defined as "non-trivial process of identifying valid, novel, potentially useful, and ultimately understandable patterns in data". Here, *data* refers to any kind of data, which accrues somewhere and which is suspected to contain some information. In order to be beneficial, this information should be valid, novel, potentially useful, and ultimately understandable (at least after some postprocessing). *Patterns* describe this information in an arbitrary language, e.g in SQL for describing a subset of the input data, which is of special interest for the domain, as model applicable to the data, in form of association rules (cf. section 2.3), or by other means appropriate for the different purposes of the knowledge discovery (see subsection 2.2). The goal of the whole KDD process is the extraction and application of knowledge derived from these patterns.

KDD is an iterative and interactive process, which involves several steps and decisions being made by a user according to the underlying data and the purpose of knowledge discovery. An overview of the different purposes and appropriate data mining techniques is given in the next section. An example for a required user decision is for instance the determination, whether derived patterns are the desired ones, eventually in order to restart parts or even the whole process with varying parameters (e.g. if the patterns do not meet the expectations). The different steps of the KDD process are presented in the following, the whole process is illustrated in figure 2.1.

Basic Steps of the KDD Process

1. **Understanding the data and identifying the goal of the KDD process** according to the concrete purposes (involves concrete domain knowledge)

2. **Data Selection**: selecting the corresponding data, which is required for achieving the defined goal ("target data")

3. **Preprocessing**: cleaning and other preprocessing of the target data (among others removing noise, checking consistency, and determining how to deal with missing/wrong data entries)

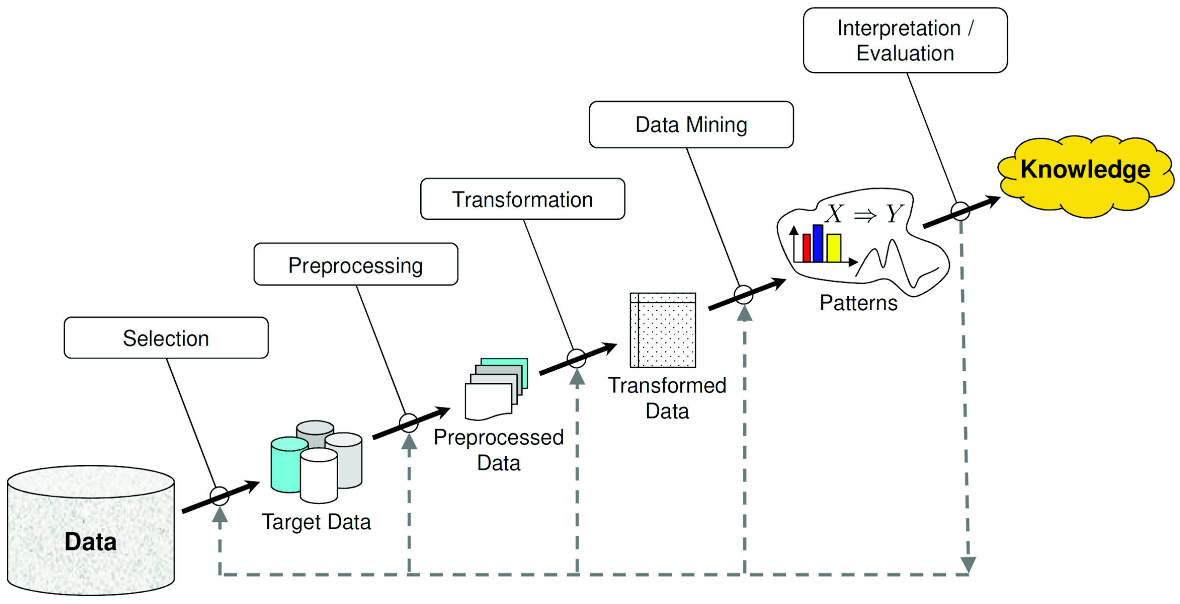


Figure 2.1: Basic flow of the KDD process (adapted from [FPSS96]). Eventually necessary user decisions and iterations are denoted by dotted arrows.

4. **Transformation**: selection of relevant features, data integration, dimensionality reduction (in order to reduce complexity or number of considered variables)

5. **Data Mining**: performing the concrete mining task on the transformed data in order to reveal the patterns of desire

6. **Evaluation/interpretation** of the discovered patterns: validating and interpreting mined patterns (eventually restarting the KDD process from one of the preceding steps); visualizing and documenting mined knowledge

The fifth step, *Data Mining*, which will be presented in the next section in more detail, is the central step in the whole KDD process. From the point of view of the KDD process, it is just a step towards the discovery of useful knowledge. However, other approaches like [TSK05] define data mining as "process of automatically discovering useful information in large data repositories", thus it is also often used synonymously for the whole KDD process.

An area related to KDD and databases systems in general is *Data Warehousing* [Pon10, CD97], where huge amounts of data are stored and aggregated in a *Data Warehouse* in order to support business reports and analyses. By using OLAP (online analytical processing) tools, data warehouses allow multidimensional data analyses, which are more efficient than e.g. using standard SQL for calculating data summaries and breakdowns. Here, the main focus lies on supporting and simplifying interactive data analyses, whereas KDD focuses on automating as much as possible of the whole process [FPSS96].

2.2 Data Mining

Depending on the underlying data and the different intentions, there are several data mining tasks which can be applied in the KDD process in order to mine concrete patterns, from which domain specific knowledge can be derived. Here we give a short overview about the main areas, the basic ideas and intentions of data mining.

Cluster Analysis *Clustering* [XW05, JMF99] is one of the major data mining task. Its goal is to partition data objects from a database into a finite number of groups or classes, the so called *clusters*, so that objects from one cluster are very similar to each other, but objects from different clusters very dissimilar to each other. Since clusters of a given database D can vary in size, form and density, several clustering algorithms have been proposed in literature. Examples for the main clustering strategies are

- partitioning-based clustering (which partitions the database D into a fixed number of k clusters; a very popular algorithm is k-means [HW79], where each data object is assigned to the cluster with the nearest mean),
- density-based clustering (which regards clusters as dense areas in the space of the data objects divided by less dense areas; cf. for instance DBSCAN [EKSX96]),
- hierarchical clustering (which creates cluster hierarchies in form of dendrograms by merging clusters with minimal distance; an example is the single link approach [JD88]), and
- combinations of these approaches (for instance CURE [GRS98], which is a middle ground between partitioning-based and hierarchical clustering, or the density-based hierarchical clustering approach OPTICS [ABKS99]).

Outlier- or Anomality Detection A tasks very closely related to clustering is *Outlier*- or *Anomality Detection* [Hod11, HA04]. According to Hawkins' definition [Haw80], an *Outlier* is "an observation that deviates so much from other observations as to arouse suspicion that it was generated by a different mechanism". Outlier and anomality detection is often applied by credit card or telephone companies in order to detect fraud, or in medicine to detect certain diseases. Transfered to clustering, an outlier (an anomalous data object) can be defined as object, which does not belong to any cluster. According to this definition, not all clustering strategies mentioned before are able to find such outlier, since e.g. partitioning-based clustering assigns every data object to one of the k cluster. A strategy more suitable for detecting outliers is density-based clustering, where an objects is only assigned to a cluster, if sufficient other data objects lie within an ϵ-neighborhood of this object. Further strategies and general overviews of outlier and anomality detection can be found in [TSK05] and [Hod11].

Before starting the clustering process (or the detection of outliers or anomalities), the concrete number of clusters is unknown, i.e. the different algorithms perform so-called *Unsupervised Learning* of the structures from the data objects. Clustering concepts are used in several places in this thesis, among others in chapters 3, 5 (in order to obtain and apply a clustering-based time series discretization) and 6 (for organizing time series representations according to their similarity).

Classification and Prediction The goal of *Classification* and *Prediction* [DHS01] is to *predict* the "class" of previously unknown data objects basing on a training set, which consists of objects where the "class" membership is known. In contrast to clustering, where the number of clusters is unknown apriori, the concrete "classes" are known before. Traditionally, the term "classification" is used for predicting the class c_i of a data object, where c_i is a class from a finite set of classes, i.e. $c_i \in C = \{c_1, ..., c_k\}$. The term "prediction" is used, if the predicted "class" is a numeric value, e.g. for predicting values of a time series (cf. chapter 3 for more details about time series and their analysis in general, and chapters 5 and 6 about time series prediction).

The task of classifying or predicting new data objects is to learn a classifier or predictor on base of a set of (labeled) training data in order to estimate the real class (or value) of these objects. Classification and prediction are counted to the area of *Supervised Learning*; they are mainly applied in chapter 3 (where several time series analysis approaches are discussed) and chapter 7 (for classifying sleep stages).

Association Rule Mining As indicated by the name, *Association Rule Mining* [KK06a, HGN00] considers mining associations between items, which are captured in form of *Association Rules.* Association rules are logical implications of the form $\{antecedent\} \Rightarrow \{consequent\}$, which express that the occurrence of a set $\{antecedent\}$ implies the occurrence of $\{consequent\}$ in a transaction with a certain probability. Since major parts of this thesis (chapters 4 and 5) are dealing with association rules, association rule mining is presented in the next section (2.3) in more detail.

Time Series Analysis and Temporal Data Mining *Time Series* are sequences of data points, which are typically measured at certain successive points in time. These data points simply may represent numeric values (for instance river levels measured at certain time points), or more complex attributes (e.g. whole customers' transactions like in market basket analysis in a supermarket from which association rules can be mined). The analysis of these data points with respect to their temporal coherence is the main subject of *Time Series Analysis* [SS06]. Since time series analysis is the main issue of this thesis, it is handled in an extra chapter (3) in detail.

In general, *Temporal Data Mining* [LS06, AO01] denotes the knowledge discovery from data, which contains temporal information. A complete understanding of all phenomenons requires to view this kind of data as sequence of points or events. Thus temporal data mining and time series analysis are very similar, and depending on the point of view, these two terms can be used synonymously, or time series analysis can be regarded as subtask of temporal data mining. Since time is the designated attribute in temporal data mining, all kinds of temporal data can be seen as time series (consisting of a timestamp and further attributes); for this reason we are basically considering time series analysis and temporal data mining as equivalent. However, if time series are limited to the simplest kind, which contains only two attributes in every data point (timestamp and value, e.g. of a measurement), it is also reasonable to regard time series analysis as subtask of temporal data mining.

Since the following data mining tasks play a minor role in this work, they are only mentioned very briefly due to their general importance.

Spatial Data Mining In contrast to temporal data mining, where time is the designated attribute, *Spatial Data Mining* [RHS01, KAH96] has its focus on space-related data. Thus the main issue is the analysis of spatial relationships, for instance of single spatial distributions of certain attributes or dependences between such spatial distributions. Applications of spatial data mining are among others geo marketing (marketing with respect to local conditions) and environment protection.

Text- & Web-Mining The analysis of texts and the usage of these texts is the main issue of *Text* and *Web Mining* [SW08]. From the text mining point of view, the web is interpreted as structured text with hyper links, where other media contents are neglected (of course, the analysis of images or videos is another large area of data mining, but since it is not related to this thesis, we are omitting these areas here). Chapter 7 is related to text mining (since it presents an approach for measuring text similarity with temporal data mining methods).

Summarization The task of *Summarizing Data* [Mie05] is to create a shortened version which still has the characteristic features of this data. Examples can be found in text mining (creating summaries of larger texts) or time series analysis (creating an abstract description or a shorter approximated version of a time series).

2.3 Association Rule Mining

In the last two decades there has been an intense research interest in *Association Rule Mining* (ARM). ARM is one of the major tasks in data mining, and since large parts of the research work presented in this thesis concern ARM, it is presented in this section in detail.

In 1993, Agrawal, Imielinski, and Swami published one of the first papers about mining association rules in large transactional databases [AIS93], which introduced the basic notions of ARM. Intuitively, an *Association Rule* (AR) is an implication of the form $X \Rightarrow Y$, where X and Y are two disjoint sets of items. As the name indicates, an AR captures associations between itemsets. A typical application of ARM is for instance *Market Basket Analysis* in a super market, where customer transactions are recorded in the cash register for later analysis. An itemset in this example could be $\{\text{beer}, \text{potato crisps}\}$, which means that a customer bought together the two items beer and potato crisps. If sufficient transactions support this itemset, the itemset is called *Frequent Itemset* (FI) and an AR $\{\text{beer}\} \Rightarrow \{\text{potato crisps}\}$ can be generated, which expresses that customers, who buy beer, also likely buy potato crisps. Two measures for the interestingness of ARs are *Support*, which states how often a rule is applicable to a given database, and *Confidence*, which is stated in our case by the ratio of transactions that contain potato crisps in the transaction that contain beer.

In their work from 1994, Agrawal and Srikant proposed the *Apriori* algorithm [AS94], which is one of the fundamental algorithms for mining FIs. It uses the inversion of the monotonicity property of FIs (i.e. subsets of frequently occurring itemsets are also frequent) in order to generate FIs more efficiently. The following subsection introduces the basic notions of ARM, followed by subsection 2.3.2 which presents the Apriori algorithm and shortly mentions other approaches for discovering FIs.

2.3.1 Basic Notions

In the following, we introduce the basic notions of ARM formally according to [TSK05]. Let $I = \{i_1, i_2, ..., i_d\}$ be the set of all items in a market basket database and $D = \{t_1, t_2, ..., t_N\}$ the transactional database itself, containing N transactions. Every transaction t_i consists of a subset of items $X_i \subseteq I$, called *itemset* (in the case of market basket analysis, e.g. in a supermarket, these items represent the products that are bought by a customer in one transaction) and optional further pieces of information like timestamps (which state when the transactions have occurred) and customer IDs. An itemset with exactly k elements is called *k-itemset*. Itemsets are supposed to be ordered according to a lexicographic ordering $\leq_{lex}$, i.e. a itemset k-itemset can be written as $X = (i_{x_1}, i_{x_2}, ..., i_{x_k})$, where $i_{x_1} \leq_{lex} i_{x_2} \leq_{lex} ... \leq_{lex} i_{x_k}$. (An example for the naturally induced lexicographic ordering of words is for instance *butter*$\leq_{lex}$ *milk*, since the word *butter* starts with *b*, which precedes *m* from *milk* in the Latin alphabet.)

Note, that a database of timestamped transactions could also be presented as high dimensional time series $D = (t_1, t_2, ..., t_N)$ (cf. chapter 3 for more information about time series in general), but here we prefer to use the traditional way as introduced above and in the seminal paper [AIS93], since time is not in the main focus of ARM.

A measure for the *interestingness* of an itemset X is its *support count*. The (absolute) support count σ of an itemset X is the number of transactions in D, which contain itemset X. Thus, it can be stated as

$$\sigma(X) = |\{t_i \mid X \subseteq t_i, t_i \in D\}|.$$

Accordingly, the relative *support* of an itemset X is the number of occurrences of X divided by the number of all transactions in D, or formally as

$$sup(X) = \frac{\sigma(X)}{N}.$$

Association Rule An *Association Rule* (AR) is an implication expression of the form $X \Rightarrow Y$, where X and Y are two disjoint itemsets. Semantically considered, an AR $X \Rightarrow Y$ means that if itemset X occurs in transaction t_i, itemset Y will most likely be in that transaction too. (Note, that "most likely" will be specified in the following using interestingness measures, e.g. support and confidence.) The following example concretizes the meaning of an AR.

Example 3. *Semantics of an AR in Market Basket Analysis*
In market basket analysis, an AR $X \Rightarrow Y$ states that customers, who buy items from itemset X, tend to buy the items from itemset Y too. An example for such an association rule might be {computer} $\Rightarrow$ {internet security package}, which expresses that people tend to buy an internet security package in order to secure their new purchased computer. The knowledge derived from discovered ARs can be used in many ways, for instance for product placement (in order to make customer's shopping more comfortable or to increase sales volume).

Note, that the given definition of $X \Rightarrow Y$ is the standard definition of an AR, without special regard to the temporal component. Several kinds of *Temporal Association Rules* (TARs), which especially capture temporal coherences, can be defined through modifications of the standard definitions, as will be introduced and explained in chapter 4.

The strength of an AR $X \Rightarrow Y$ can be measured in several terms, two typical ones are *support* and *confidence* (cf. section 5.2 for the J measure [SG92], which is another interestingness measure, and [Han05] for further measures and more details).

Analogously to the support of an itemset, the support of an AR determines how often this rule is applicable to the given database. Thus the (absolute) support count of $X \Rightarrow Y$ equals the support count of itemset $X \cup Y$, i.e. $\sigma(X \Rightarrow Y) = \sigma(X \cup Y)$. The relative support again is given by dividing the support count by the number of transaction:

$$sup(X \Rightarrow Y) = \frac{\sigma(X \cup Y)}{N}$$

An AR with a very low support can simply occur per coincidence and thus likely be uninteresting for the outcome of the market basked analysis. To avoid such kinds of rules a threshold `minsup` for the minimal support of a rule is given by the user. Itemsets that have a support higher than `minsup` are called *large* or *frequent itemsets* (FIs), non-frequent itemsets are also called *small.* A *maximal large itemset* is a frequent itemset of maximal size, i.e. a frequent k-itemset X_k for which applies that there is no frequent l-itemset X_l with $l > k$ and $X_k \subset X_l$.

The confidence of an AR $X \Rightarrow Y$ determines how frequently items in Y appear in transactions that contain X, i.e. it is given by

$$\text{confidence}(X \Rightarrow Y) = \frac{\sigma(X \cup Y)}{\sigma(X)} \left(= \frac{sup(X \cup Y)}{sup(X)} \right).$$

Confidence is a measure for the reliability of an AR, and again a threshold named `minconf` is defined for the minimal confidence a rule must have in order to be discovered. An AR (or an itemset) which satisfies both the `minsup` and `minconf` constraint is called *strong.*

Association Rule Mining The task of ARM in a given market basket database D is the discovery of every AR, which support is larger than `minsup` and which confidence is larger than `minconf`. This task can be composed in the following two steps:

1) The **discovery of frequent itemsets** from database D, which supports exceed `minsup`, and

2) the **generation of association rules** from these frequent itemsets, which confidences exceed `minconf`.

In contrast to the first step (the discovery of FIs), step 2 (the generation of association rules from frequent itemsets) is a task easy to solve: given frequent itemset Z, generate every two possible disjoint subsets X and Y of Z, with $X \cup Y = Z$, and check whether confidence$(X \Rightarrow Y) \geq$ `minconf`. Due to this rather easy solution, for simplicity we sometimes use the terms itemset and AR analogously in this work. (More information about the generation of ARs can be found in [AS94] or [Han05].)

2.3.2 Frequent Itemset Discovery

The discovery of frequent itemsets is the main challenge in ARM, thus here we focus on the first step mentioned in the previous subsection. One of the first and probably best-known algorithm for discovering frequent itemsets is the *Apriori Algorithm* [AS94]. In order to mine ARs efficiently, the Apriori algorithm exploits the monotonicity (or rather the anti-monotonicity) property of frequent itemsets.

(Anti-) Monotonicity Property[1] A property $\pi : \mathcal{P}(R) \to \{0, 1\}$ is said to be monotonic, if $(\pi(Y) = 1) \Rightarrow (\pi(X) = 1)$ for all sets X and subsets Y with $Y \subseteq X$, and anti-monotonic, if $(\pi(X) = 1) \Rightarrow (\pi(Y) = 1)$.

When π is anti-monotonic $(\pi(Y) = 0) \Rightarrow (\pi(X) = 0)$ for all $Y \subseteq X$.

Obviously, the frequency of an itemset is an anti-monotonic property, since an itemset X can only be frequent if all subsets $Y \subseteq X$ are frequent too. Thus, the `minsup` constraint can be used in order to prune the search space for discovering all frequent itemsets more efficiently. If an itemset Y is not frequent, no superset X with $Y \subseteq X$ can be frequent, thus we do not have to regard (and calculate the support) for any superset X.

Apriori The *Apriori Algorithm* [AS94] uses the anti-monotonic property of frequency for discovering all frequent itemsets from a given database D efficiently. The basic idea is to "grow" k-itemsets from $(k-1)$-itemsets level by level, but by only regarding itemsets which can be frequent. A k-itemset X can only be frequent, if every subset of X (i.e. every j-itemsets with $j \leq k$, which only contains items from X) is frequent. Thus, in contrast to the naive approach, which would regards every possible itemset and calculate its support in order to check its frequency, the Apriori algorithm only regards k-itemset, which can be combined from smaller frequent itemsets[2].

The smallest possible itemsets, 1-itemsets, are easy to calculate by simply scanning the database once and counting the occurring items. Thus, in a first step the Apriori algorithm generates L_1, the set of all frequent 1-itemsets, by doing this. A candidate set C_2 of possible frequent 2-itemsets is created from L_1 (by combining all frequent 1-itemsets). Due to the anti-monotonicity property, this candidate set is an upper-bound for all frequent 2-itemsets, and in normal applications its size is by far smaller than the number of all possible 2-itemsets (since normally only a small portion of all

[1] adapted from [Häm10]

[2] Due to the requested lexicographic itemset ordering, combining $(k-1)$-itemsets in order to "grow" valid k-itemsets is equivalent to joining all $(k-1)$-itemsets pairwise, which agree in the first $k-2$ items. Here, joining refers to the binary natural join operator $\bowtie$ of the relational algebra from database theory [Cod70]; the join of the two 3-itemsets $(1, 2, 3)$ and $(1, 2, 4)$ for instance results in $(1, 2, 3) \bowtie (1, 2, 4) = (1, 2, 3, 4)$.

items is frequent). Thus only the support values of the candidates in C_2 have to be calculated in order to discover all frequent 2-itemsets. The candidates which support values exceed `minsup` are saved as L_2, which is the complete set of all frequent 2-itemsets. The basic idea of this procedure is repeated iteratively, i.e. for every level $k = 3, 4, ...$ the candidate set C_k of all possible k-itemsets is composed from L_{k-1}. After that, candidates with infrequent subsets are removed, the support values of the remaining k-itemsets are calculated and frequent k-itemsets are stored in L_k, until a k is reached for which $L_k = \emptyset$ (no more frequent k-itemsets are discovered). After that, the union of all created L_k is the complete set all frequent itemsets from database D.

A further improvement of the Apriori algorithm is the usage of hash trees for implementing the quite often used subset function, which is explained in [AS94] in detail.

The Apriori algorithm is the quasi-standard algorithm for mining ARs; its principle is used as basis for many ARM papers from different authors. An approach, which uses the Apriori principle on basis of two tree structures is [GCL00], where the transactional database is reorganize in form of the P and T tree. These tree structures can be uses in order to calculate the support with less database accesses; since they can be extended to find interesting temporal intervals in the database, in which certain temporal association rules are applicable, it will be presented in the according chapter about TARM in detail (cf. chapter 4).

An approach which avoids the frequent itemset candidate generation, which is quite costly for the Apriori and similar algorithms, is FP (frequent pattern) Growth [HPY00]. Experiments showed that FP growth, which is basing on a prefix tree structure, is quite comparable in efficiency to Apriori [HGN00].

Overviews of several other ARM approaches and further details can be found in [TSK05], [Han05] and [KK06a].

3 Time Series Analysis

Time series occur in various domains in great number and heterogeneity. In the simplest and most frequent case a time series contains successive measurements of certain quantities, for instance measurements of EEG in medicine (as appearing in sleep data analysis [SC10a] and epilepsy seizure prediction [MAEL06]), of sun rays in astronomy (for sunspot analysis and prediction [XWWL08]) and of geological data (in order to predict river levels [SC10c] or the El Niño phenomenon [CDK^{+}99]). Naturally, the discovery of special coherences, the understanding of certain characteristics and the possibility of predicting future values of time series are of special interest for these domains, which led to an intense research interest in *Time Series Analysis.*

Corresponding to the plurality of time series and their different origins, there are several actions and tasks which can be performed on the different types of time series. Time series can either be continuous or discrete, they can contain numeric or symbolic values, or even events or transactional data with temporal information. Thus, time series representation, transformation from one representation into another, and the selection of an appropriate distance or similarity measure are very important before conducting concrete actions and analyses on time series.

This chapter presents details about time series and their analysis which will be used throughout this thesis, namely different time series representations, discretization methods (section 3.1), distance and similarity measures (section 3.2), and concrete data mining tasks which can be performed on time series (section 3.3).

3.1 Representation, Dimensionality Reduction and Discretization

Basic Time Series Definition A *Time Series* $s = (x_1, ..., x_n)$ is a sequence of n data points x_i. In the simplest case, these data points can consist of real numbers (typically measured at certain equidistant points in time), e.g. derived from stock prices [FCLN07] or the examples mentioned at the beginning of this chapter. In a more complex case, they can be multidimensional, for example in market basket analysis [BSVW99], where x_i corresponds to a customer's transaction (containing time of the transaction, customer ID, bought items, and so on). The latter is an example of a very high dimensional discrete time series, whereas the time series derived from the first examples are continuous[1]. Classically, the area of time series analysis considers time series of the simplest case, but as already mentioned in section 2.2, it is reasonable to extend this notion and consider all kinds of data with temporal information as time series. From this point of view, time series analysis corresponds to *Temporal Data Mining*, thus these two areas are handled together in this chapter.

The first issue dealing with time series is the selection of an appropriate *Time Series Representation.* The easiest and most "natural" way to represent a time series, is using its original representation without prior modifications (apart from depending on the data possibly necessary preprocessing steps like smoothing or interpolation), as done for instance in [ALSS95, LR98]. Time series of length n can be regarded as object in the n-dimensional vector space [GAIM00], which is only a modification of the point of view.

Dimensionality Reduction and Discretization Due to fine-grained measurements or simply due to a long period of time captured, time series can be very large, either too large to handle efficiently or simply unnecessarily complex for certain purposes. Thus, sometimes time series are transformed into simpler approximate representations, which is also denoted as *Complexity* or *Dimensionality Reduction.* Examples for this kind of representation are piecewise linear functions [DGM97, KS97, GS99] or representations through diverse other basis functions (piecewise constant approximation [FJMM97], piecewise polynomial approximation [OWT$^+$00], and so on). Time series can also be transformed from one domain into another, e.g. from time to frequency domain by using the discrete Fourier [AFS93] or Wavelet transform [CF99], where only

[1]Since time series derived from real world phenomenons have to be measured and stored somehow for further processing, they cannot be continuous in the mathematical sense (even if the original signal is continuous, e.g. like the voltage of the myocardial muscle tension measured in EEG). But in spite of that, we uses the term *continuous* for time series derived from continuous signals (with numeric values) in order to distinguish them from time series derived from discrete signals or artificially created time series with a limited range of possibly symbolic values.

some parts are used as simpler approximative representation (e.g. the first k Fourier coefficients). Another way of representing time series is to describe and represent them by special characteristics, e.g. by time series motifs (cf. section section-rel-ts-dm-tasks), which is done in [SC12b] (see section 6.3.1).

For some reasons a discretized or symbolic representation of a former continuous time series might be desirable, e.g. for applying string manipulation or other appropriate techniques (for instance from bioinformatics). In the past years, a vast number of time series discretization and quantization methods have been proposed in literature (cf. e.g. [DFT03, KK06b] and [LKWL07] for an overview). As first step in this *discretization* process the time series have to be segmented applicatively, either in segments of the same or of different lengths (cf. for instance [KCHP93, MGG$^+$09] for different segmentation procedures). After that a symbol (or a “state” as denote in [Höp01]) is assigned to each segment, resulting in a discretized representation $D(s) = a_1a_2...a_j$. The discretized representation can either be seen as sequence of points a_i, where each point contains start and end time (or optionally, one of these two time points, or the mean) and the symbol/state assigned to this segment, or simply as string containing the assigned symbols in the accordant order (if the segments are of the same length).

In the following, we present two methods in detail, which discretize continuous time series with equidistant points in time, namely SAX [LKLcC03] and the clustering-based method proposed in [DLM$^+$98]. Both discretization methods result in segments of the same length, i.e. every symbol derived from a segment has the same duration. The clustering-based method [DLM$^+$98] is an example for an inductive approach, since symbols (or shapes) are derived directly from the time series (e.g. like in [DLM$^+$98] by clustering all subsequences, which are derived by a sliding window). Deductive approaches fix the shapes of interest in advance; time series are represented as sequences of previously defined basic shapes (for instance of the shape definition language [APWZ95]) or other terms (e.g. like “constant”, “linearly increasing” or “convexly increasing”, which can be derived by checking all possible combinations of zero/positive/negative first and second derivative, given that the signal from which the time series was derived is almost everywhere twice differentiable [Höp02b]).

SAX The *S*ymbolic *A*ggregate appro*X*imation *SAX* [LKLcC03] is used in several time series problems as representation [Eam12]. As preprocessing step, SAX uses piecewise aggregate approximation (PAA) in order to reduce the complexity and dimensionality of a time series s. PAA is obtained by divided s into j equally sized frames, where the mean value of the data points falling in a frame is used as value for representing it. After that, s is normalized, which results in Gaussian distributed data points[2]. These data points are discretized into equiprobable symbols of a chosen

[2]Empirical experiments show that nearly all time series show a Gaussian distribution in their data

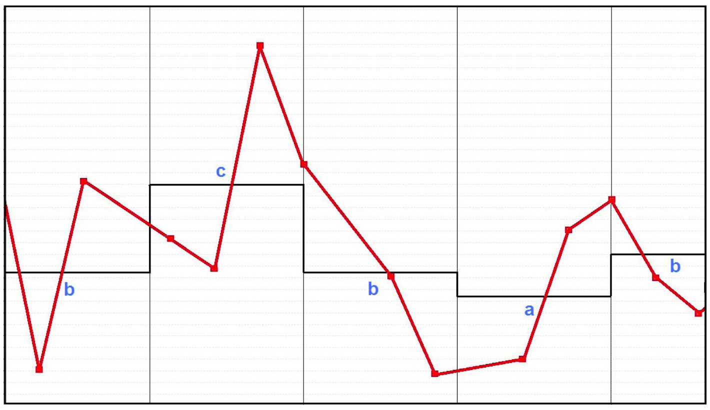


Figure 3.1: Simplified illustration of SAX on a cut-out of an example time series: the time series (red with rectangles), its PAA representation (black), and the symbols assigned to the time series in each frame. With $\Sigma = \{\mathrm{a}, \mathrm{b}, \mathrm{c}\}$, the discretization of the cut-out would be "bcbab".

alphabet Σ. Figure 3.1 illustrates the procedure on a cut-out of an example time series.

The use of PAA as preprocessing step yields a smoothing effect, which might distort dynamic characteristics of certain time series with the effect, that some movement shapes cannot be captured by SAX anymore. A variant of SAX, which claims to resolve this issue, is [PLCS10]; another variant for huge amounts of data is iSAX [SK08].

Clustering-Based Discretization The *Clustering-Based Discretization* method proposed in [DLM+98] is another interesting and often adapted approach, where the alphabet Σ is derived directly from the data. A time series s is discretized by first segmenting the time series using a sliding window mechanism, where a window of fixed size l is running with a fixed step size (in the following, assumed to be 1 for simplicity in presentation, although it can be set larger) over the data points from s, so that each window forms a subsequence $s_i = (x_i, ..., x_{i+l-1})$. The set of all subsequences $W(s) = \{s_i \mid 1 \leq i \leq (n - l + 1)\}$ is clustered using a partitioning-based approach (e.g. k-means [HW79]) in order to obtain a fixed number of clusters. After clustering, each subsequence s_i belongs to a certain cluster, thus a certain symbol representing the cluster can be assigned to every subsequence s_i. By using k-means as clustering algorithm, we obtain a clustering with exactly k clusters, thus an alphabet Σ of size k can be used, where each symbol represents a cluster. The discretized time series $D(s)$ consists of the symbol representation for each subsequence in the respective order. Thus a data-derived representation of time series s is obtained, where, according to window and step size, each symbol represents a primitive shape. Figure 3.2 illustrates the whole procedure on a simple example time series. (Note, that if the subsequences overlap more, they will be correlated more. Thus it makes sense to allow a step size of > 1 for

points after being normalized [LKWL07].

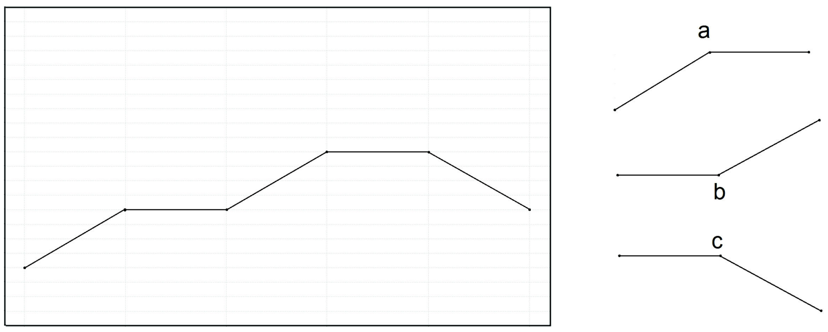


Figure 3.2: Illustration of the clustering-based discretization for a simple time series $s = (1, 2, 2, 3, 3, 2)$, window size $l = 2$ and step size 1 (adapted from [DLM+98]). The primitive shapes derived through k-means clustering (with $k = 3$) are shown on the right of the plot; with $\Sigma = \{\mathrm{a}, \mathrm{b}, \mathrm{c}\}$ the discretized time series would be $D(s) = \mathrm{abac}$.

the sliding window movement in order to prevent too much correlation.)

A completely different possibility for representing time series is offered by generative models. Here, the main goal is to find a specific model, which most likely generated a given time series. Probabilistic generators, which are commonly used in order to generate time series, include Semi-Markov and Hidden Markov Models [GS00, LOW01, SC10c], neural networks [FDH01], and grammars [dlH05].

Time series derived from transactional databases with temporal information, e.g. transactional supermarket records containing timestamps and lists of items bought by customers, count to a very high-dimensional kind of time series, where only very few alternative representations are known besides their transactional representation. For the search of sequential patterns [AS95], the transactional database (or the time series containing the transactions) is transformed by introducing new symbols for certain sets of items to make the discovery of correlations in the data more efficient. In the area of standard and temporal association rule mining (cf. chapter 4) special tree structures are used for representing the original database in order to mine several kinds of association rules more efficiently [HPYM04, CGL04, SC10b].

3.2 Distance and Similarity Measures

After selecting an appropriate representation, the *distance* (or *similarity*) between two (parts of) time series has to be defined, where the property of dealing with noise, outliers, different amplitude scalings, missing values, and shifts along the time axis is particularly important.

Euclidean and Minkowski Distance One of the most used approaches to measure the distance between two time series is the *Euclidean Distance*. Time series are regarded as points in the n-dimensional vector space, where the i-th point of one time series is compared linearly with the i-th point of the other time series (e.g. as applied in [AFS93] and [CF99]). The Euclidean distance between two time series $a = (a_1, a_2, ..., a_n)$ and $b = (b_1, b_2, ..., b_n)$ of equal length n is defined as

$$dist_{\text{Euclid}}(a, b) = \sqrt{\sum_{i=1}^{n}(a_i - b_i)^2}.$$

The Euclidean distance is the special case $p = 2$ of the *Minkowski distance*, which regards the differences of every two points a_i and b_i, raises it to a given power p, sums up these results, and raises the sum to the power $\frac{1}{p}$. Formally, it is given by

$$dist_{\text{Minkowski}}(a, b) = \left(\sum_{i=1}^{n} |a_i - b_i|^p\right)^{\frac{1}{p}}.$$

Further common cases of the Minkowski distance include the *Manhattan* ($p = 1$) and the *Maximum* or *Chebyshev distance* ($p = \infty$).

The computation of the Euclidean distance (and all other Minkowski variants) is very easy and fast, but it has several drawbacks: in many cases it may be inappropriate to compare the i-th point of time series a to the i-th point of time series b, especially when dealing with time series of different lengths, scalings, or translations. Small shifts in the x-axis might quickly result in huge distances, although the curves of the two time series still might look very similar. The left plot in figure 3.3 illustrates two time series, that look very similar to the human eye, but due to small shifts in the x-axis the alignments made by the Euclidean distance results in a huge distance.

The problem of dealing with time series of different lengths, scalings, or translations can be compensated by using linear transformations. [ALSS95] for instance uses sliding windows of fixed size to find matching pairs of subsequences in two time series a and b. Before a subsequence from time series a is compared with a subsequence from time series b, they are normalized in order to solve the scaling problem; the search for matching subsequences solves the translation problem. Finally, non-overlapping matching subsequences are merged in order to find the longest match length of the two time series. A similar approach is [DGM97], where a linear function f has to be found, so that a long subsequence of time series a approximately matches a long subsequence of time series b after applying f. These subsequences do not have to consist of contiguous data point, which can also compensates gaps or outliers (the data points just have to appear in the same order).

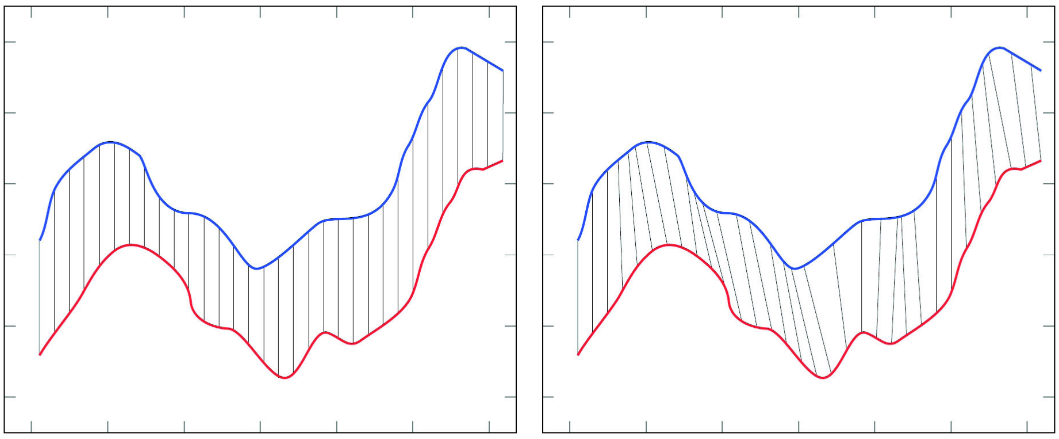


Figure 3.3: Illustration of two similar time series, which points are aligned by the Euclidean distance (left) and the DTW distance (right) for comparison. Note, that the time series are shifted in the y-axis for clarification.

(Derivative) Dynamic Time Warping For compensating shifts along the time axis, which are problematic for the Euclidean distance and several other distances, a pseudo-distance measure based on *Dynamic Time Warping* (DTW) has been proposed in the literature [BK59, SC78, BC94]. The basic idea of DTW, which was first proposed in the field of speech recognition, is to find a non-linear matching between the points of the two time series a an b, which is intuitively equivalent to locally (or even globally) stretching or compressing the time series in the x-axis. Since DTW can handle time series of different length, we describe this process for the two time series $a = (a_1, a_2, ..., a_n)$ and $b = (b_1, b_2, ..., b_m)$ (of length n and m respectively) according to [KP01]. For aligning these two time series, a $n \times m$-matrix is constructed, where each element (i, j) contains the distance $d(a_i, b_j)$ between the points a_i and b_j. Thus, each matrix element corresponds to an alignment between the points a_i and b_j. A warping path W is a set of contiguous matrix elements which defines a mapping between the whole sequences a and b. Formally, $W = (w_1, w_2, ..., w_K)$, where $w_k = (i, j)_k$ and $max(m, n) \leq K \leq m+n-1$. The goal of Dynamic Time Warping is to find an optimal warping path W, i.e. the path that minimizes the warping cost $DTW(a, b)$ between a and b, which is given by

$$DTW(a, b) = \frac{1}{K}\sqrt{\sum_{k=1}^{K} w_k}.$$

The result of this process is a distance measure which captures the human perception of similarity much better than the Euclidean distance. The right plot in figure 3.3 gives an example for this observation, peaks are aligned with peaks and valleys with valleys, where it is appropriate.

The naive way of calculating an optimal alignment between two time series results in a high computational effort, which has been improved by many approaches in the literature including constraining the warping path (e.g. with the Itakura parallelogram [Ita75] or the Sakoe-Chiba band [SC90]), using lower bounding techniques (like [YJF98], LB_Keogh [Keo02], or Anticipatory DTW [AWK^{+}09]) and approximating the optimal warping path (e.g. using FastDTW [SC04]).

A modification of the DTW distance is the *Derivative Dynamic Time Warping* (DDTW) distance [KP01]. While the DTW algorithm is considering the absolute values of two time series at certain points, the algorithm for calculating the DDTW distance considers the derivation at these points. One consequence of this is, that two time series, which have a very similar shape, but are shifted along the y-axis, will be regarded as very similar using DDTW (the distance will be close to 0), whereas the distance calculated by DTW will be much higher due to the shifts along the y-axis and the comparison of the absolute values. Thus DTW will probably not reveal that the two time series have a very similar shape, but due to its definition DDTW will.

The definition of the DDTW distance is quite similar to the DTW distance, thus here we just state the difference: two time series a and b are aligned by constructing a $n \times m$-matrix, which elements (i, j) contain the distance $d(a'_i, b'_j)$, where a'_i and b'_j denote the local derivation at the points a_i and b_j of the two time series. The optimal warping path is calculated analogously to DTW (for more details see above and cf. [KP01]), which can be done by dynamic programming.

The DDTW distance is a very robust distance measure suitable for many applications, since it reduces singularities (i.e. several point of time series a are aligned to only one point of b, which is one problem of DTW) and other wrong warpings (for instance when two time series are compared, which contain few - or no - warpings of the x-axis) [KP01].

An approach which does not regard every point for comparing two time series is [AKK^{+}06]. The authors assume that it is sufficient for many applications to regard only partial information (significant parts) of the time series and consider only intervals of which the values exceed a given threshold. Another way of calculating the distance between time series is the use of probabilistic models like in [KS97], where a time series is deformed according to some probability distribution in order to match another time series.

Distance Functions for Symbolic Time Series Time series with discrete non-numerical values can be compared by checking whether symbols (or sequences of symbols) match or not, or by using string techniques like the *Edit Distance* (also known as *Levenshtein Distance*, cf. e.g. [MVM09]), which regards the minimum number of edits (insertion, deletion or substitution of a symbol) needed to transform one string (representing the time series) into the other. The edit distance between two different symbols from the discretization alphabet Σ is at most 1, because at most one substitution is needed to transform the one symbol into the other. In situations, where symbolic time series are derived from real valued time series (e.g. by discretization), it is reasonably to assume that the distance between two symbols can be more than 1 (since it represents the distance between two arbitrary real values). A more appropriate distance measure for this situations is the *MINDIST Function*, which was especially proposed for SAX. Assuming a lexicographical ordering of the discretization alphabet Σ, MINDIST makes uses of a symbols distance function $dist_m$, which outputs different values for the distance between two symbols according to their lexicographical distance and the size of the discretization alphabet Σ. The distances can be retrieved from lookup tables, which contain breakpoints dividing a Gaussian distribution in an arbitrary number of equiprobable regions (equal to $|\Sigma|$) [LKLcC03] (cf. table 3.1 for an example of such a lookup table). With $dist_m$ defined like this, the (normalized) MINDIST function between two time series x and y can be defined as:

$$MINDIST(x,y) = \frac{1}{n}\sqrt{\sum_{i=1}^{n}(dist_m(x_i,y_i))^2}$$

"Blurry" (approximate) matches between two time series can be found for instance by using operators of the shape definition language [APWZ95], which describe the coarse form of a time series. The distance for time series represented by generative models can be calculated by regarding the distance between a time series and the time series generated by a given model (which leads for instance for deterministic models to a discrete distance measure resulting in 1 or 0 - the time series is generated by a given model or not - [MTV95, SM00]), or by directly interpreting the probability that a time series was generated by a given model as distance, as applied amongst others for stochastic generative model like grammars [Smy99] and Markov Models [GS00].

	a	b	c	d
a	0	0	0.67	1.34
b	0	0	0	0.67
c	0.67	0	0	0
d	1.34	0.67	0	0

Table 3.1: Example of a lookup table displaying symbol distances $dist_m$ used by the MINDIST functions for an alphabet $\Sigma = \{a, b, c, d\}$.

3.3 Data Mining Tasks on Time Series

Depending on domain, intention, time series representation, and several other factors, there are many actions and analyses which can be performed on time series in order to mine concrete knowledge. This section provides an overview about these data mining tasks, starting with the general tasks from section 2.2, which can be transfered to and conducted on time series.

Clustering, Classification, Segmentation, Change Point Detection and ARM
The common denominator of the two machine learning tasks *Clustering* and *Classification* is the assignment of objects to certain clusters or classes, either with additional information provided apriori (e.g. class membership) or not. In time series analysis, objects can either be parts or entire time series [Smy99, WK06, OV10], depending on the intention of the analysis. Applications of classification are the detection and classification of certain waves in time series derived from EEG measurements [HKY02] and sleep stage scoring [SC10a], where epochs of sleep are classified into predefined sleep stages for later analysis. The size of the regarded time series parts can either be fixed (for instance to 30 seconds lasting epochs, which are typically regarded in sleep stage scoring) or determined dynamically.

The subarea dealing with the dynamic segmentation of time series is known as *Time Series Segmentation* or *Change Point Detection*, where a time series has to be segmented reasonably according to significant points, special properties or behavior on certain intervals [GS00, KCHP04, KYM07, AA10].

Another subarea is the analysis of associations in or between time series. *Association Rule Mining* for instance can be applied directly to time series, if the transactional database is regarded as time series (where every data point consists of customer transactions an possibly further pieces of information) as already mentioned in section 2.3. Of course, for “normal” time series with simpler data points (containing e.g. measurements), ARM cannot be applied directly. The application of ARM and especially temporal ARM to “normal” time series [SC11a, SC11b] is one contribution of this thesis, it is examined in detail in chapter 5, which introduces the required concepts and steps in order to reveal interesting associations in and between time series.

Prediction *Predicting* future values of time series is a very important data mining task for many applications [WG94][3]. Plenty of approaches have been proposed in the literature, where many reformulate the problem as other data mining tasks, e.g. as

[3]Time series prediction is surprisingly often applied to the prediction of financial data, although it is commonly known and widely accepted that time series from this area can only be predicted very limited [Fam70].

classification (a classifier applied to a new object "predicts" the class of this object), using association rules (cf. chapter 5 or [LHF98]), clustering (possibly in combination with other techniques like association rule mining [DLM+98], cf. also chapter 5) or applying generative models (a model that generates a given time series can generate some more points in order to continue and thus predict the time series with a certain degree of accuracy).

In addition to that, plenty of concrete methods for predicting time series have been proposed, among others basing on statistical analysis (e.g. regression analysis [Wei87, KF02], which again could be seen as reformulation of another data mining task, namely numerical classification), constructing linear and non-linear models for the time series [CD07, PRF+09], neural networks [FDH01, GLT01, BK07], fuzzy models [CG05] or complex physical models, which have to be adapted to the specific application and domains in detail.

Physical models are non-learning, mainly used for simulation purposes, and thus only of minor interest for data mining. Due to their nature, statistical methods like regression analysis only reveal general trends in time series, which allow only an approximative prediction. Rule- and pattern-based approaches like [LHF98, CWY+11, MRBD11] yield quantitative predictions (tendencies) and only in special cases concrete values. Since time series derived from natural phenomenons usually consist of several components like general trends, economic cycles, saisonal components and irregular remaining variations, an ideal system would have to be a combination of diverse approaches integrating these components. However, in order to identify all components and design a system combining them appropriately, much domain knowledge has to be integrated in the process, which is again in contrast to our general goal of discovering knowledge without investing too much specific domain knowledge (as for instance done for constructing and using physical models). In chapter 6, we present two approaches [SC10c, SC12b] for time series prediction, which do not rely on additionally provided domain knowledge. Both approaches analyze a time series database in order to find two correlated time series (by using derivative dynamic time warping [SC10c] and motif-based time series representations [SC12b]), on which base a Hidden Markov Model is constructed, which is used to predict one time series with help of the other. The usage of similar behaving time series could also be seen as kind of domain knowledge, but in contrast to the domain knowledge earlier mentioned, this knowledge is derived automatically and directly from the data.

The *discovery* and *recognition* of certain *patterns* in time series is a important and large subarea in the analysis of time series, where *Query by content* and *Motif Discovery*, which will be explained in the follwoing, are two main tasks. The major goal of association rule mining is the discovery of frequent itemsets, which could also be denotes as frequent patterns, thus ARM could be counted to the area of pattern discovery too.

Query by Content The task of *Query by Content* is to find a subsequence or a whole time series in a larger time series or time series database [FRM94]. Therefore the inquired patterns can be given explicitly or merely by shape [APWZ95], and either exact or approximative matches ("blurry matches") can be desired. Several approaches base on the Euclidean or other variants of the Minkowski distance, which allow fast indexing (cf. section "Dimensionality Reduction and Discretization" above and "Query Support" below). Examples for such approaches are [AFS93, CF99], which use the Fourier or Wavelet transform to extract features in order to find patterns more efficiently. Another distance measured used for query by content is the DTW distance [BC94], possibly in combination with techniques for fast indexing [YJF98, Keo02]. A different approach basing on parameters describing the patterns of desire is the waveform recognition method proposed in [HKY02], or the combinations of such techniques [SC10a].

Motif Discovery *Time Series Motifs* have been introduces in [PKLL02], where a motifs has been defined as previously unknown frequently occurring pattern. *Motif Discovery* aims at identifying these approximately repeated parts in a time series. A common way to solve this task is to define a minimal length L for a motif, scan the database and enumerate all subsequences of given length L [BST09, YKM$^+$07, FASB06]. Since association rule mining [AIS93] is very similar to motif discovery, ideas from this field have been adapted to the discovery of motifs. A more efficient approach for discovering motifs is based on the apriori principle [AIS93], which uses the monotonicity property of frequent itemsets (cf. subsection 2.3.2). Adapted to frequent pattern discovery in time series, this property says that every subsequence of a frequent sequence is frequent too. Several algorithms (e.g. [GST01, Bod05]) exploit this property in order to mine frequent sequences efficiently by first discovering short motifs, from which larger ones are grown together step by step. The efficiency of such algorithm can be increased by using appropriate data structures like tries [Bod05, BST09]. Similar to the usage of association rules for prediction (cf. chapter 5), time series motifs can also be used to predict future values of time series, e.g. for weather [MRBD11] or stock prediction [JLH09].

Several kinds of time series motifs have been proposed in the literature, which can be classified in flat motifs (e.g. [PKLL02]) and motifs with one or more gaps [GST01, BST09]. A generalizations of these continuous and non-continuous motifs has been proposed in [BST09], where the authors also present an efficient algorithm to discover such generalized sequential motifs. A *Semi-Continuous Sequential Motif* is a motif, in which maximal n gaps of a maximal length d are allowed.

Many approaches have been proposed for discovering the different kinds of motifs, but only very few work has been invested in the analysis of the discovered motifs. In most of the approaches this task is simply delegated to a human domain expert, who would have to investigate a huge sum of discovered motifs manually. Recently, the authors of [CA11] proposed and evaluated an approach for calculating the *Statistical Significance* of time series motifs, which bases on ideas from bioinformatics [RSV07] and association rule mining [Web07]. After discovering all time series motifs (or motifs exceeding a minimum support threshold), the probability of each motif is calculated using simple Markov Chain Models (cf. [CA11]), and then statistical hypothesis tests are applied in order to calculate the p-value, or rather an estimation of this value, of each motif. The p-value is the probability of the motif count to be at least as large as the observed motif count, just by chance. A motif is said to be significant, if its p-value equals or is smaller than a predefined threshold value α. The evaluation on several different datasets presented in [CA11] affirms the usefulness of this approach, since the number of motifs to be inspected manually can be reduced significantly (ranging from 66% of the original value down to 0%, where of course discarding all discovered motifs is arguable at least for certain purposes, as will be discuss later in chapter 6.3).

Data Stream Mining Areas with large amounts of emerging data, which contain systems with only low capacity for processing these amounts of data including history (e.g. sensor or computer networks), are typical scenarios for the application of *Data Stream Mining* [GZK05]. Algorithms from this domain only have access to a small window of the recently emerged data. Main issues of data stream mining are simple classification tasks and especially incremental learning. An approach which uses incremental learning for discovering temporal association rules is [GNTA10]. Incremental Learning can also be useful for treating new data, which is added to an already analyzed large dataset, to prevent starting the whole mining procedure from the start.

Query Support Especially when performing data mining tasks on large time series or large time series databases, *Time Series Indexing* becomes an important issue in order to optimize speed and performance, e.g. for clustering and finding similar time series. For answering database queries more efficiently, many approaches transform time series from one domain into another, e.g. from the time to the frequency domain [AFS93, FRM94, CF99]. In the target domain, indexing techniques like the R* tree [4] [BKSS90] are typically used on an approximation of the time series (e.g. derived by only regarding the first few Fourier coefficients) in order to reduce the dimensionality.

[4]Recent experiments showed that the R* tree, which is more or less the standard structure in this area, does not always perform best [KP10], e.g. in comparison to the quad tree [Sam90].

Especially the recognition of certain patterns in large time series databases cannot be realized efficiently without indexing.

4 Temporal Association Rule Mining

Association rule mining (ARM) is one of the major tasks in data mining (cf. chapter 2). In ARM, transactional data is analyzed with the purpose of revealing associations within this data. These associations can be captured as association rules (ARs), which are implications of the form $X \Rightarrow Y$. In market basket analysis, such an association rule could for instance express that customers who buy product X, also tend to buy product Y. The time when a transaction occurred is not important for standard ARM, i.e. the whole database is analyzed at once without specially regarding the temporal component. But since this additional information might reveal additional interesting knowledge within the data, discovering such kind of *Temporal Association Rules* (TARs) is the main goal of *Temporal Association Rule Mining* (TARM).

This chapter introduces TARM in general (section 4.1) and several approaches for capturing TARs in detail (section 4.2). After that, it presents two of our contributions to the area of TARM, namely TARGEN [SC09] (section 4.3), a dataset generator which creates transactional datasets containing certain temporal coherences (i.e. it provides an excellent basis for testing TARM algorithms), and a tree-based approach for mining several kinds of temporal association rules (section 4.4).

4.1 Introduction

In general data mining, time does not play a major role, it is processed like any other component of the given data. But as real life shows, time needs particular attention in many cases, for instance in a supermarket: if the whole day is regarded, e.g. beer and potato crisps are bought together probably relatively seldom, which will be expressed by a low support value for the two itemset. But if only the evening is regarded, for example the time from 7PM-9PM, the amount of transactions that support {beer} $\Rightarrow$ {crisps} in this segment of the database will probably be much higher. Depending on the minimal support value to discover an AR, standard ARM might overlook such a rule, whereas naturally an approach especially regarding the temporal component should find it. ARM with particular attention to the temporal component is called *Temporal Association Rule Mining.*

The knowledge derived from standard and temporal ARs can be applied in several ways. A rule like {beer} $\Rightarrow$ {crisps} could be used in order to make shopping more comfortable for the customer, by placing the two products beer and crisps side by side. Or, more realistically, it could be used to increase the profit of a supermarket by placing several other "offers" between them. And, regarding the temporal component of such rules, shopmen can lure customers by making special "offers" at certain reasonable times, e.g. for beer and crisps in the evening, or for doughnuts and coffee in the morning.

Since standard ARM has been discussed in section 2.3 in detail, this chapter concentrates on TARM. Several approaches for capturing TARs are imaginable, e.g. rules from events that happen sequentially [AS95], cyclically [ORS98], periodically [HDY99] or in special times which can be described by calendar-based patterns [LNWJ01], and researchers from the whole world have been and are still developing new algorithms for discovering TARs efficiently.

Dataset Generators These algorithms have to be tested and evaluated extensively to show their benefits, but: On which data should these algorithms be tested? Tests on real world datasets are indispensable, but they have two major drawbacks: The first one is that appropriate real world datasets are hardly available to the public. Most of the few public datasets available come from research facilities. The effort to make real market basket data irrecognizable in the sense of data privacy is simply too high for most of the enterprises that collect real world data. The only larger real world dataset which is available to the public[1] is the Retail Market Basket Data Set supplied by Brijs [BSVW99], which contains transactions from an anonymous Belgian retail supermarket store. Additionally, up to now it is not custom that datasets contain timestamps,

[1]to the best of our knowledge

which naturally is a precondition for TARM. The second drawback of real life datsets is that real life data contains interferences, e.g. by influences of season, weather, shop location, fashion and others, on which certain algorithms might fail. Since we do not exactly know what is in the data, we do not have a controlled scenario for testing. These two handicaps motivate the development of generators for synthetical market basket data. Generators can produce a vast amount of data, and the user is in total control of everything in this data. In general data generators have their drawbacks too: algorithms can perform in another manner than on real world data, and created datasets do not reflect exactly real world customer behavior. Synthetical datasets can just reflect real world facts that are included in the underlying model of the generator. Several generators have been propose in literature, most of them basing on IBM's Almaden Research Center Data Generator [AS94]. But just as well as it is not commonly for real world datasets to contain timestamps, most of the proposed generators do not create timestamps and are thus useless for TARM. Some generators produce timestamped transactions, e.g. [ORS98], [LNWJ01] and [OLC08], but either no temporal coherences, which could be expressed by TARs, are contained in datasets, or the TARs are limited to one special kind of TARs, which is proposed in most of the cases by the same researchers too.

Motivated by this need for appropriate dataset generators, which contain temporal coherences, we developed *TARGEN* [SC09], a novel *market basket dataset generator integrating temporal coherences*, which will be presented in section 4.3. TARGEN models various temporal and general aspects from real life in order to create datasets that contain interesting and realistic temporal coherences. Besides taking into account general customer behavior, customer group behavior and several other aspects, the created datsets are timestamped in a manner which allows TARM with regard to three major kinds of TARs, namely cyclic, calendar- and lifespan-based association rules (cf. section 4.2). The appearance of all these aspects can be controlled by the use of accordant parameters, which brings the user in total control of every little facet of the emerging datasets. Thus, the generated data provides an excellent basis for developing and testing market basket analysis and TARM algorithms in general.

Mining Several Kinds of TARs The second contribution of this work to the area of TARM is an *approach for mining several kinds of TARs* [SC11a, SC11b], which makes use of two novel tree structures. Most of the existing TARM approaches are limited to the discovery of one specific kind of TARs, whereas our approach considers cyclic, lifespan- and calendar-based ARs. The two novel tree structures used are the EP and the ET tree, which represent a given transactional database and allow to mine TARs very efficiently. As a first step in constructing the ET tree, the database is reorganized in one single pass. The EP tree is an extended version of the P tree [GCL00], which is used to speed up standard ARM by maintaining a partial support counter, extended with additional temporal information. Due to its special structure, the EP tree can be used to calculate the support of itemsets more efficiently and to collect important temporal information with only little additional effort. These pieces of information, namely itemsets, their standard support and lists of their timestamps, are stored in the ET tree, which provides a basis for efficiently discovering several kinds of TARs.

The next section presents the background for TARM, i.e. it gives an overview and explains some of the major approaches in detail. Section 4.3 presents TARGEN, our dataset generator for TARM, followed by section 4.4, which explicates our tree-based approach for mining several kinds of TARs.

4.2 Background on TARM

Comparing the underlying databases for standard ARM and TARM, the only difference is the fact that every transaction is required to be timestamped in order to mine TARs. In the example of market basket analysis in a supermarket, timestamps denote the time when a customer purchases his products, in weblog analysis, these timestamps denote the time, when a user clicks on a link. For TARM, most of the definitions from standard ARM can be used as well (cf. section 2.3), except from the definition of support, which has to be modified. Several kinds of temporal association rules have been proposed in literature, e.g. sequential [AS95] and generalized sequential patterns [SA96] (ARs derived from events that happen sequentially), periodic [HDY99] and cyclic association rules [ORS98], or lifespan- [AR00] and calendar-based association rules [LNWJ01], and also many approaches for mining them efficiently have been presented [HPM+00, VVV05, SC10b]. The approaches mentioned before regard the temporal component as points in time, but of course other representations are also possible, e.g. interval-based representations [KF00a, WR07]. Episode [MTV97] and generalized episode mining [MT96a] are closely related to TARM, but with focus on finding certain events in time. In the following, we explain the approaches of most interest for this thesis in more detail.

Lifespan-Based TARs A straightforward approach for capturing simple temporal particularities for ARM has been proposed in [AR00], where the *lifespan* of an itemset is considered. Intuitively the lifespan of an itemset X is the span of time in which X occurs in the transactional database D. The *Temporal Support* of X is defined as the number of occurrences of X in D divided by the number of transactions which happen in the lifespan of X. This approach is reasonable, since in large databases one could find information related to products that do not necessarily exist throughout the whole time when a database was gathered. With the standard definition of support, products or itemsets could be found which already have been discontinued at the time the mining process was performed, and new products, which were introduced at the end of the time in the database, might not be found due to support restrictions. This problem is solved by the lifespan-based support definition, to what we refer as temporal support, which is stated formally as follows.

We assume that every transaction $t_i \in D$ is associated with a timestamp, denoted by $t(t_i)$, which stands for the time when the transaction occurred. Naturally, a total temporal order $\leq_t$ is defined, where $t(t_1) \leq_t t(t_2)$ denotes that transaction t_1 happens before t_2 (or at the same time). Let $X \subseteq I$ be an itemset. The *lifespan* of the itemset is defined as the interval $\mathbb{I}_X = [t(t_i), t(t_j)]$, where t_i is the first transaction, which contains X, and t_j the last transaction in D, with $t(t_i) \leq_t t(t_j)$. (Note, that for $t(t_i) = t(t_j)$ this interval degenerates to a point in time, e.g. if X occurs only in one transaction in the whole database.) Let $D\left[\mathbb{I}_X\right]$ denote the segment of the database D, which contains all transactions, that happen in the lifespan $\mathbb{I}_X$ of X. Then, the *Temporal Support* of itemset X is defined by

$$\text{tempsup}(X) = \frac{|\sigma(X)|}{|D\left[\mathbb{I}_X\right]|},$$

where $\sigma(X)$ denotes the support count of X as introduced in section 2.3.

Cyclic TARs Another interesting approach is [ORS98], which introduces the notion of *Cyclic TARs*. An AR is called cyclic, if it represents regular cyclic variations over time. A meanwhile quite familiar example for this kind of ARs is the rule $\{\text{beer}\} \Rightarrow \{\text{potato crisps}\}$. Probably, the support of this rule will be relatively low during the whole day, but in certain regular time intervals, e.g. every day from 7-9PM (and in the corresponding database segment, respectively), it will have probably a much higher support. To state this approach formally, some definitions and another perspective of time are needed, which will be explained in the following.

We assume that time is given in a fixed unit τ, and that the i-th time interval, $i \geq 0$, is denoted by τ_i, which corresponds to the time interval $[i \cdot \tau, (i+1) \cdot \tau]$. Let $D\left[i\right]$ denote the set of transactions that were executed in τ_i. We refer to $D\left[i\right]$ as *Time Segment i*. The *cyclic support* of an AR $X \Rightarrow Y$ in this time segment i is the fraction of transactions in $D\left[i\right]$ which contain the itemset $(X \cup Y)$. The confidence measure for $D\left[i\right]$ is defined analogously. A cycle c is a tuple (l, o), which consists of a length l and an offset o, $0 \leq o \leq l$, both given in time units. An AR is said to have a cycle $c = (l, o)$, if it holds in every l-th time unit starting with time unit τ_o. An AR that has a cycle is called *cyclic*. The beer and potato crisps rule for instance would have the two cycles $c_1 = (24, 19)$ and $c_1 = (24, 20)$, that denote “every 24 hours in the 19th hour” and “every 24 hours in the 20th hour”, which is is exactly every day’s time from 7 till 9PM.

Calendar-Based TARs An approach that is even more flexible with regard to the time intervals, in which a TAR holds, is [LNWJ01]. The time intervals are specified by a former defined *Calendar Schema*, e.g. by (year, month, day). In this schema for instance every year’s Christmas Eve could be represented by the tuple $(*, 12, 24)$, where $*$ is a wildcard denoting every arbitrary integer in the domain of the accordant attribute, in this case year. As well as in both approaches mentioned before, support is calculated relatively to the number of the transactions that occur at the specified time intervals. Thus, we omit the formal definitions of calendar-based support, refer to [LNWJ01] and give one more example for clarification instead: In the calendar-based schema (month, day, hour) our beer and potato crisps rule would hold in the time represented by $(*, *, 19) \cup (*, *, 20)$, because this denotes the 19th and 20th hour of every day in every month, which again refers to the time from 7 to 9PM.

Obviously, calendar-based ARs can express many cyclic ARs and vice versa, depending on the calendar schema and time granularity uses.

Sequential Patterns The notion of *Sequential Patterns* has been introduced in [AS95]. Sequential patterns are patterns derived from events that happen sequentially, they can express ARs like the following one, which could be discovered in a video rental store: people who rent the DVD “The Fellowship of the Ring” and then “The Two Towers”, will most likely rent the DVD with the third part of the “Lord of the Rings” trilogy, namely “The Return of the King”, thereafter too. Sequential patterns in this case just state that if someone rents the first and then the second movie, he will most likely rent the third movie afterwards some day, but they do not state when this happens. Sequential patterns have been generalized in [SA96], and their discovery has been made more efficient in several works (e.g. in [SA96] and [PHMA$^+$01]). A concept similar to sequential pattern mining is episodes mining [MT96b, MTIV97], where an episode is a collection of events that occur relatively close to each other.

Basic TARs TARs, which involve specific temporal constraints, can be captured by the simple rule format $A \Rightarrow^T B$ [DLM$^+$98]. $A \Rightarrow^T B$ denotes that if A occurs, B will occur within time span T. We will refer to this kind of TAR as *Basic Temporal Association Rule*. An approach extending this simple rule format from one time span to several is [HDT02], which introduces the notion of “sequential association rules with constraints and time lags”.

A combination of two previously mentioned approaches is [ZSAS02]. It is a rather straightforward extension of [AR00] and [LNWJ01], which results in a system for mining two kinds of TARs (lifespan- and calendar-based) by means of only a small alteration of the primary algorithms.

Another combination approach is [VVV05], which combines calendar-based ARM [LNWJ01] with two tree structures proposed in [CGL04]. These tree structures developed for improving standard ARM are the P and the T tree, on which bases we also developed our EP and ET tree for improving TARM (cf. section 4.4.1). In standard ARM one P and one T tree is created for the whole database, which yields a good speed-up. The temporal version from [VVV05] creates one P and one T tree for every possible calendar-schema and thus yields a speed-up too, but on cost of a huge overhead. (In contrast, our approach for mining several kinds of TARs presented in section 4.4.1 needs only one EP and one ET tree for representing the whole database.)

All these approaches presented so far had their focus on events existing at one or certain points in time. Since for some kind of data the start and end time point of a transaction is available too, there are also approaches for discovering TARs on interval-based data, e.g. [KF00b, Höp01, WR07], which consider the relationships between intervals in terms of Allen's interval logic [All83]. [WR07] for instances generates "richer" interval-based TARs, which intuitively said corresponds to merging "smaller" TARs. In [Höp01], this kind of TARM is also called state sequences mining.

Further information about TARM can be found in the two overview papers [AO01] and [LS06], and in [RHS01], which provides a large bibliography of temporal data mining research.

4.3 TARGEN - A Market Basket Dataset Generator for TARM

In this section we describe TARGEN, a dataset generator, which we presented in [SC09]. TARGEN creates timestamped market basket data, which contains temporal coherences that can be captured as TARs. Its underlying model contains various aspects from real life, which results in synthetical datasets that reflect nearly exactly real world customer behavior. The aspects concerned are general customer behavior including timing information, customer group behavior and temporal features (cyclic, calendar- and lifespan-based TARs; cf. previous section). These aspects can be controlled by the adjustment of the appropriate parameters, so that the user is still in total control of what is in the created dataset. Thus our generator provides an excellent basis for testing and evaluating market basket and TARM algorithms in general.

The next subsection (4.3.1) presents related work on existing dataset generators. TARGEN, our approach for Temporal Association Rule GENeration, is explained in the following subsections: subsection 4.3.2 contains TARGEN's underlying model of the real world, subsection 4.3.3 the detailed description of the algorithm for data generation. An evaluation of our generator can be found in subsection 4.3.4, followed by subsection 4.3.5 with conclusions of this section about TARGEN and directions for future work.

4.3.1 Related Work on Existing Market Basket Dataset Generators

The probably most cited data set generator for market basket data is IBM's Almaden Research Center Data Generator [AS94]. It models a retailing environment and produces ordinary market basket data without timestamps. It bases on the fact, that usually people buy certain items together. Every such set of items can possibly be a large itemset or even a maximal large itemset, depending on how often it will occur in data. After the according parameters stated in table 4.1 are adjusted, the generator creates transaction per transaction by first determining the size of the next transaction, and then assigning items to the transaction by picking itemsets from a predefined series of maximal potentially large itemsets, and then adding the contained items to the transaction. (These steps are explained in more detail in subsection 4.3.3.)

A generator that produces timestamped market basket data is [OLC08], which is applicable to both retail and e-commerce environments. The generation of itemsets and transactions is done analogously to [AS94], the timestamps are created on the basis of several studies about customer behavior [BSVW99, VKM02, MAR$^+$04]. These studies show that the number of sales is not distributed equally over time: in retail

denomination	meaning	default value
$\|D\|$	number of transactions in database D	10000
$\|T\|$	average size of transaction	10
$\|L\|$	number of maximal potentially large itemsets	1000
$\|I\|$	average size of maximal potentially large itemsets	4
N	number of items	1000

Table 4.1: Parameters of IBM's Almaden Research Center Data Generator. These parameters are also used by our dataset generator TARGEN. (TARGEN's further parameters are given in table 4.2.)

environments there is a peak of sales at the end of the week, and daily peaks varying from business to business. The generator models up to two daily peaks at a user given time of the day. E-commerce business has no closing times, but peaks of sale are reported during leisure time, i.e. in the evenings and on weekends. Hence the amount of sales is set to 150% on existing peaks. A declining number of sales is reported in the night and at the start and end of the day, which is considered by setting the sales volume to 50%. Thus the distribution of timestamps in the synthetical datasets is very close to real life data, but unfortunately there are no specific coherences in the created datasets which could be stated reasonably as TARs.

The paper [ORS98] mentioned in the previous section presents not only a framework for cyclic ARs, but also a generator for market basket data, which contains cyclic coherences. Again, this generator is based on [AS94], i.e. generally transactions are created one after another by assigning items from a series of maximal potentially large itemsets. The authors extended the generation process by adding a number of temporal patterns (denoting cycles) and a noise parameter to each maximal potentially large itemset. In order to model the fact, that certain rules might occur cyclically, maximal potentially large itemsets are added to a transaction with timestamp t, if t is covered by one of the assigned temporal patterns. In order to model that real world transaction will contain both cyclic and non-cyclic rules, a given maximal potentially large itemset is added to the current transaction with a certain probability, which is given by the noise parameter, even if the timestamp is not covered by temporal patterns.

The generator proposed by [LNWJ01], which creates datasets containing coherences that can be captured by calendar-based ARs, works in a similar manner: several temporal calendar-based patterns are assigned to each maximal potentially large itemset in a single common pattern itemset, and independent series of maximal potentially large itemsets, the so called per-interval itemsets, are created for every basic time interval[2].

[2]A basic time interval is an interval given by a concrete calendar pattern (without wildcards).

After that, the transactions are filled with items by choosing a ratio of itemsets out of the common pattern itemset and of the according per-interval itemsets.

The mechanism of [OLC08] for producing timestamps is very sophisticated, thus our approach makes use of this idea for distributing transactions over time. In addition to that, TARGEN bases on [AS94] in consideration of filling transactions with itemsets, and it makes use of the basic idea from [ORS98] for introducing additional temporal coherences (cyclic, calendar- and lifespan-based).

4.3.2 TARGEN's Model of the Real World

Our dataset generator TARGEN models two kinds of shopping environments, namely a retailing environment and an e-commerce environment. We assume that shops in a retail area are open to the public on workdays and closed on Sundays and general holidays, whereas e-commerce businesses do not have closing times. Peaks of sales are considered in both environments analogously to [OLC08].

The *general customer behavior* is modeled according to [AS94], which means that generally transactions are filled with items from maximal potentially large itemsets, according to a corruption level for every transaction, which determines how many items from a maximal potentially large itemset are dropped (in order to model that not every potentially large itemset is maximal). The assumption, that large itemsets usually contain common items, is modeled in the process of creating series of maximal potentially large itemsets by a correlation level, which states the amount of items, that are determined by the preceding maximal potentially large itemset.

People can be classified into a certain number of *customer groups* according to their sales, every group contains an average number of customers. People in a specified group buy similar itemsets, and they do their shopping at a certain time, both influenced by a group individuality parameter up to a certain degree.

Temporal coherences are modeled by assigning temporal patterns to a certain ratio of the maximal potentially large itemsets, the so called pattern itemsets. A fraction of these patterns is cyclic, which can be stated by cyclic ARs, and the rest of these patterns complies with calendar-based ARs. The time granularity for cyclic and calendar-based ARs is an hour, and the calendar schema used is $(\text{month}, \text{day}, \text{time})$[3]. Lifespan-based coherences are inserted by assigning a shorter lifespan to a fraction of the items occurring in the database. Table 4.1 and table 4.2 summarize the most important parameters of TARGEN.

[3]Note, that these matters can be adjusted easily.

denomination	meaning	default value
correlation level	correlation of max. pot. large itemsets	0.5
$\|G\|$	number of customer groups	10
g_n	avg. number of customers per group	15
g_i	group individuality	0.3
f_g	fraction of group itemsets in non temp. itemsets	0.5
f_{life}	fraction of items having a shorter lifespan	0.01
$time_{DB-start}$	start time of database	01.01.2012
$time_{DB-end}$	end time of database	31.12.2012
$time_{retail-open}$	opening time of retail store	6
$time_{retail_close}$	closing time of retail store	22
P_r	ratio of temp. pattern itemsets in transaction	0.4
p_{num}	number of temp. patterns associated with	
	pattern itemsets	2
$p_{frac-cyc}$	fraction of cyclic pattern itemsets (rest is	
	calendar-based)	0.25
$p_{cyc-min}$,	minimal length of cyclic patterns	10
$p_{cyc-max}$	maximal length of cyclic patterns	100
$p_{cyc-den}$	number of cycles in a pattern	0.2

Table 4.2: Remaining parameters of TARGEN (in addition to table 4.1).

4.3.3 The Algorithm for Data Generation

Outline The outline of TARGEN's algorithm for creating datasets containing temporal coherences is the following. After adjusting all parameters, TARGEN starts producing transaction per transaction by first determining size and timestamp of every transaction. On base of these timestamps and the according parameters a group ID and a customer ID is assigned to every transaction. Thereupon the transactions are filled with items by picking itemsets from three different *series of maximal potentially large itemsets*: a portion is picked from pattern itemsets, which models temporal coherences, another portion is picked from group specific itemsets, which models group specific behavior, and the rest is picked from a general series of maximal potentially large itemsets.

The three different *series of maximal potentially large itemsets* are created as follows. The size of every series is determined by the parameters $|L|$, P_r and f_g, where P_r and f_g are values between 0 and 1. The temporal pattern itemset contains $|L| \cdot P_r$ maximal potentially large itemsets, the specific group itemsets $|L| \cdot (1 - P_r) \cdot f_g$ and the general set $|L| \cdot (1 - P_r) \cdot (f_g - 1)$ itemsets, so that the cardinality of the union of these three sets equals $|L|$. The generation processes of these three series have the first four steps[4] in common:

1. Determination of next itemset size (by means of a Poisson distribution with mean $|I|$),
2. selection of items for that itemset: if it is the first itemset, choose all items randomly, otherwise choose $k\%$ of the items from the previous itemset (k is an exponentially distributed random variable with mean that equals the correlation level) and the rest randomly,
3. assignment of weights to all itemsets in the processed series (according to an exponential distribution), followed by an normalization so that the sum of all weights is 1, and
4. assignment of a corruption level to the itemsets (obtained from a normal distribution with mean 0.5 and variance 0.1).

After these four steps, in the case of *temporal pattern itemsets*, cyclic temporal patterns are assigned to a fraction (stated by p_{frac_cyc}) of the maximal potentially large itemsets, and calendar-based patterns are assigned to the rest, whereat p_{num} patterns are assigned to every single itemset. The calendar-based patterns are chosen from the weighted space of all calendar patterns, which is created by first weighting

[4]Note, that these steps are exactly the steps from [AS94] mentioned before.

every possible calendar-pattern with p_k, where p is a value between 0 and 1 and k the number of stars (wildcards) in a pattern, and then normalizing these weights. The cyclic patterns are generated according to the parameters p_{cyc_min} and p_{cyc_max} and p_{cyc_den}, which affect length and density of cycles analogously to [ORS98].

Group specific itemsets are created for every group by establishing $|G|$ different temporal patterns, some of them predefined, e.g. for people who do their shopping mostly in the lunch break, and others are chosen randomly. A series of group specific maximal potentially large itemsets is created for every specified temporal pattern. These series and the general series of maximal potentially large itemsets are generated in compliance with step 1-4 and not altered anymore.

The total number of transactions to be generated is given by the parameter $|D|$. In a first step, the size s of every transaction is determined with a Poisson distribution with mean $|T|$, and timestamps are assigned to every transaction using the mechanism adapted and adjusted from [OLC08]. In a second pass, customer group IDs are assigned to the transactions by regarding the timestamps and the former specified temporal patterns up to a certain degree, which is stated by the group individuality g_i, that again is a value between 0 and 1. The higher g_i, the stricter the specified temporal patterns have to fit to the timestamp. In this step customer IDs are assigned randomly according to the average number of customers in a group stated by the parameter g_n.

The main step of *assigning items to transactions* is done as follows. $s \cdot P_r$ items are picked from the pattern itemsets, by first determining all itemsets, which fit to the current timestamp (by means of their temporal patterns), and then by regarding the weights of the remaining itemsets. An itemset is not always added completely to the transaction, but according to its correlation level, i.e. items are dropped from the itemset one by one as long as an uniformly distributed random number between 0 and 1 is less then the corruption level. If the addition of the remaining items from an itemset would burst the calculated transaction size, the items of this itemset are added in half of the cases anyway, and elsewise they are added to the next possible transaction.

Thereafter, $s \cdot (1 - P_r) \cdot f_g$ items are picked from the appropriate group specific itemset and $s \cdot (1 - P_r) \cdot (f_g - 1)$ itemsets from the common general series of maximal potentially large itemsets (regarding weights, correlation level and quantity), so that the transaction contains altogether approximately s items.

4.3.4 Evaluation

We implemented TARGEN in Java on an Intel Core 2 Duo 3.0 GHz desktop PC running Windows Vista with 3 GB of main memory, where we conducted several experiments for evaluating purposes. All experiments are based on the standard configuration stated in tables 4.1 and 4.2. With this configuration our generator produces datasets with a size of 585 KB in 2.6 seconds, containing an header with information, transactions in every row (timestamps stored Unix-like as long integers), followed by customer ID as integer and a list of integers denoting the itemset. Tests with varying parameters show that size and running time scale up linearly with $|D|$ and $|T|$.

For checking temporal coherences in the generated data we implemented an algorithm for discovering lifespan-based TARs and both the algorithms for finding cyclic and calendar-based ARs from [ORS98] and [LNWJ01], respectively, as well as their proposed generators for creating datasets containing the accordant rules. We produced two categories of datasets with TARGEN, one containing only datasets with cyclic coherences (by setting p_{frac_cyc} to 1 and f_{life} to 0) and one with only calendar-based ones ($p_{frac\_cyc} = 0$). The comparison between these and comparable datasets created with the cyclic or calendar-based approach, respectively, shows, that TARGEN outperforms both. The implementation of [LNWJ01] has a considerably longer running time of about 10 times longer, whereas the running time of [ORS98] is just marginally longer. TARs of the corresponding kind are contained in the datasets in a similar amount, naturally according to the adjustment of the parameters. Beside its speed, TARGEN's main benefit is the possibility of modeling up to three different kinds of ARs at once in the generated data ($p_{frac\_cyc} \neq 0$, $p_{frac\_cyc} \neq 1$, $f_{life} \neq 0$), with total control via the according parameters, which proved to work as desired.

4.3.5 Conclusions on TARGEN and Future Work

This section described TARGEN, a generator for market basket datasets, which models various temporal and general features from real life, that can be controlled totally by means of the appropriate parameters. The main focus of our generator was the incorporation of temporal coherences in the generated data, which can be expressed by several kinds of TARs. The evaluation and the comparison of our approach with related ones showed that our generator works very well and produces datasets which almost behave like real life datasets, which has to be ascribed to its sophisticated underlying model. For these reasons, TARGEN provides an excellent foundation for evaluating and developing algorithms for both temporal and standard ARM.

Nevertheless, a generator for synthetical data can only be as good as its underlying model, and synthetical data can never replace real life data totally. Extending TARGEN's model with even more apposite real life facts would reduce this cleft definitely.

Other reasonable extensions could be approximate matching of timestamps with temporal patterns and predetermined TARs, which are inserted in the emerging data. In addition to the three kinds of TARs, which TARGEN can produce at the moment, there are further meaningful approaches for TARs mining, such as sequential patterns and basic TARs like $X \Rightarrow^T Y$ (as presented at the end of section 4.2), which are very interesting for market basket analysis too, and which could also be integrated in our generator.

4.4 Tree-Based Mining of Several Kinds of TARs

Market basket analysis is one important application of ARM. Real life market basket databases usually contain temporal coherences, which can only be captured by special TARM approaches (cf. sections 4.1 and 4.2). This section presents a tree-based approach for mining several kinds of TARs (i.e. cyclic, lifespan- and calendar-based TARs; of course in addition to mining standard ARs), which has already been proposed in [Sch08] and [SC10b]. Our approach bases on two novel tree structures, called *EP* and *ET tree*, which are derived from existing approaches improving standard association rule mining. They are used as representation of the database in order to make the discovery of TARs more efficient.

This section is organized as follows. The EP tree, the ET tree and related tree structures are presented in subsection 4.4.1, which integrates them in the TARM process. Details about the implementation of our tree-based approach for TARM are given in subsection 4.4.2, followed by an evaluation in subsection 4.4.3. Finally, we present conclusions and identify directions for future work concerning our tree-based approach for TARM in subsection 4.4.4.

4.4.1 EP tree and ET tree

This section presents the idea of the *EP* and the *ET tree*, which form the basis of our tree-based approach for TARM. In addition to that, it presents three further tree structures, namely the SE [Rym92], the P and the T tree [GCL00], from which the EP and ET tree are developed. The SE tree is a simple tree structure which contains all itemsets enumerated from the items; the nodes of all other tree structures mentioned are arranged in this manner. The P and the T tree have been introduced in order to speed up standard ARM; this section shows how to transfer this quality to TARM.

The second of the two new tree structures mentioned above, the ET tree, is a data structure which stores together itemsets with their (standard) support and temporal information about their occurrences. With this information, it is easy to calculate the temporal support of an itemset and to analyze it for further temporal coherences. The ET tree is built level-by-level up to a certain size on base of the EP tree, which itself is inferred from the P tree [GCL00]. The EP tree extends the P tree with temporal information. Both the ET and the EP tree make use of the set enumeration framework proposed in [Rym92], which structure simplifies the process of calculating support and gathering temporal information about itemsets. The creation algorithm of the EP tree scans the database and reorganizes it in only one pass in form of this tree, from which the ET tree is also built in one pass.

```
for $i = 1, ..., N$ do
    forall the subset $s \subseteq X_i$ do
        begin
            supportCount($s$)++;
            timestampListOf($s$).add($t(t_i)$);
        end
    end
end
```

Algorithm 1: A simple brute force algorithm for calculating support and gathering temporal information of every subset $s \subseteq I$. The itemset contained in transaction $t_i \in T$ is denoted by X_i, the timestamp of t_i by $t(t_i)$.

Basic Idea For more clearness about the basic idea how the EP tree is created and works, we first regard algorithm 1, which is a very simple algorithm that scans the transactional database T in one pass, in order to calculate the support of every itemset X (and collect the timestamps of every transaction, in which X is contained, in order to analyze TARM). The number of performed database accesses is minimal, namely N in a database containing N transaction, but the number of computational steps and the space required is exponential to the number of different items. Thus, this algorithm is infeasible for databases containing many items, e.g. 1,000 or more, which is usual in market basket analysis.

Let us assume that during the pass over the database algorithm 1 reads a transaction, which contains an itemset $\{A, B, D\}$. This single itemset would cause the incrementation of the support counts for $\{A, B, D\}$, $\{A, B\}$, $\{A, D\}$, $\{B, D\}$, $\{A\}$, $\{B\}$ and $\{D\}$, and it would add a timestamp to every timestamp list of these 7 sets. This huge effort for one single set is actually not necessary, since changes can be inferred subsequently from the information stored at $\{A, B, D\}$.

Partial and Total Support We formalize this interference by defining the expressions *Partial* and *Total Support*. From this point on, we omit the treatment of the temporal information for a while, because it would make both the understanding and the explanation of the approach unnecessarily more complicated. The temporal information can be handled analogously to the support, which we will point out below in detail. The *Partial Support* P_X of an itemset X is defined as the number of transactions, of which the itemsets are exactly equal to X. Then, the *Total Support* T_X of an itemset X can be calculated by evaluating

$$T_X = \sum_{\forall Y : Y \supseteq X} P_Y .$$

Obviously the support values calculated by algorithm 1 are total support values. Algorithm 2 makes use of the principle of calculating total from partial support, and is thus more efficient than algorithm 1.

```
for $i = 1, ..., N$ do
    begin
        $P_{X_i}$++;
        P.add($X_i$);
    end
end
forall the $X \in P$ do
    forall the $Y \in \mathbb{T}$, $Y \subseteq X$ do
        $T_Y = T_Y + P_X$;
    end
end
```

Algorithm 2: A more efficient algorithm for calculating the support of every itemset of the database T. The set P contains all occurring itemsets, $\mathbb{T}$ the total support values of every itemset in P, both initially set to zero.

Under the plausible assumption, that databases with realistic data contain a high degree of duplication, algorithm 2 again will be significantly faster.

To continue the example with the itemset $\{A, B, D\}$, which is contained in a transaction $t_i \in T$, we now regard the set of all subsets of $I = \{A, B, C, D\}$ presented as tree arranged according to the set enumeration framework [Rym92]. Here, every subset is represented by a certain node, which all are arranged according to the lexicographic ordering (cf. for instance figure 4.1 below, which illustrates the EP tree, where the nodes/subsets are arranged in exactly this manner). This tree-based representation can aid in the support calculation and the gathering of temporal information: while walking through the tree to a certain node - or a certain set, respectively - starting at the root node, several other nodes are traversed, which all represent subsets of the certain set of the destination node. This can be used in oder to accumulate an *interim support count* in the nodes traversed, which will ease the calculation of the total support later. Algorithm 3 describes this procedure of inscribing a single itemset of the database in the tree structure. Obviously, the length of the way from root node to a certain node is at most k, where k is the length of the longest occurring itemsets (or, depending on the point of view, the cardinality of the subset, which contains the maximum number of elements/items). The tree filled like this (without temporal information) is exactly the *P Tree* (which stands for *partial support tree*), which was introduced in [GCL00], and as already mentioned, it can easily be extended with temporal information to the *EP Tree* (denoting *extended partial support tree*), which will be explain after the rest of the basic idea is presented.

```
forall the nodes $X$ do
    $P(X) = 0$;
    $Q(X) = 0$;
end
for $i = 1, ..., N$ do                              /* for every database entry do */
    $X$ = root node;
    while node $X \neq$ NULL do
        if $X_i \supseteq X$ then                   /* if itemset $X_i \supseteq X$ */
            $Q(X)++$;
        if $X_i = X$ then                           /* if itemset $X_i = X$ */
            $P(X)++$;
            exit while loop
        else
            if $X_i \supseteq X$ then               /* if itemset $X_i \supseteq X$ */
                $X$ = eldestChildOf($X$);
            else
                $X$ = youngerSiblingOf($X$);
end
```

Algorithm 3: Accumulating partial and interim support counts during one pass over the database. In this algorithm, X denotes both an itemset and the node in the tree which contains the itemset.

After algorithm 3 is executed for every database entry, the P tree is filled and completed, and its nodes contain itemsets, partial support and interim support counts. The value of the interim support count $Q(X)$ in the node which contains itemset X is $Q(X) = \sum_{\forall Y:(Y \supseteq X,\ X <_l Y)} P(Y)$, where $<_l$ denotes the lexicographic order. With this, it is possible to calculate the (total) support of an itemset X by evaluating the following expression:

$$T(X) = Q(X) + \sum_{\forall Y:(Y \supset X,\ Y <_l X)} P(Y) \tag{4.1}$$

By now, the basic idea should be clear: the itemsets of the database are inscribed itemset-by-itemset in the P tree using algorithm 3. After that the total support of every itemset can be determined by adding the value of the interim support count from the according node in the P tree and some partial support values, given by equation 4.1, which makes this process very efficient.

EP Tree As already mentioned, the *EP Tree* has the same structure as the P tree, but additionally it has two lists which contain timestamps in every node, a partial and an interim list. Every time the interim support count of a node Y is increased on the way to a node which contains an itemset X, the timestamp of the current transaction which is being inscribed is added to the interim list of node Y, and every time the partial support count of X is increased, the according timestamp is added to the partial list. Thus, the complete list of timestamps for an itemset X can be determined in the same manner as the total support of itemset X is calculated, namely by accumulating the timestamps of the partial list stored in the node which contains X and the timestamps of some partial lists directed by equation 4.1. The complete timestamp list contains every timestamp which belongs to a transaction, where itemset X was part of, or in other words, this list contains all points in time, when a itemset has occurred. This list is a formidable basis for mining several kinds of TARs, and our prototype system described in subsection 4.4.2 makes use of this. An example of a concrete EP tree is given in figure 4.1 for illustration.

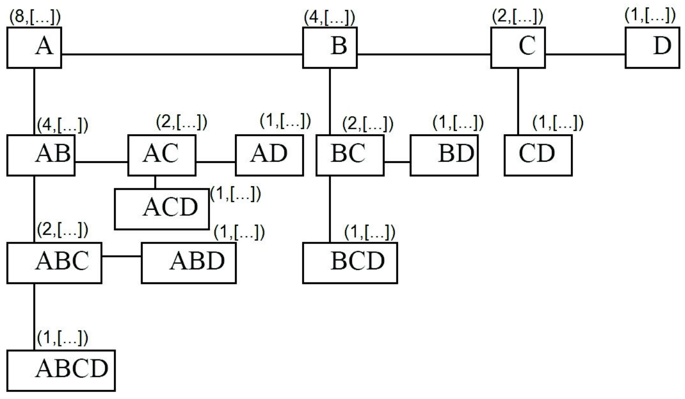


Figure 4.1: An example of an EP tree for $\mathcal{P}(\{A, B, C, D\})$, containing some interim support counts and lists of timestamps (indicated by [...]).

It is important to state that a *static* tree (with fixed nodes for every element of $\mathcal{P}(I)$) is only used in order to illustrate the whole procedure. The P and the EP tree both must be built in a *dynamic* manner, which is explained in detail in [CGL04], in order to be feasible. By doing this, the trees consist of considerably less nodes, while the basic arrangement and thus the desired functionalities are still given.

In addition to the P tree, which is used for calculating support more efficiently, [CGL04] introduces another tree structure, the *T Tree* (which stands for *target tree*), which is used for the final step of finding frequent itemsets. The T tree represents *target* item sets, for which the total support is to be computed on base of the P tree. The nodes of this tree again represent the itemsets and are designed to contain their total supports. Its structure is the dual structure of the P tree, i.e. that every subtree includes only supersets of its root node which contain an attribute that precedes all those of the root node (see figure 4.2). In order to prevent an exponential size in the

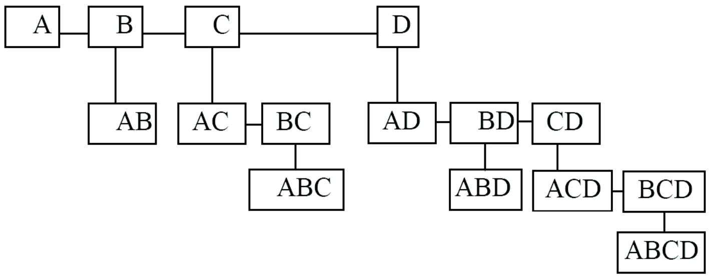


Figure 4.2: The T tree for $\mathcal{P}(\{A, B, C, D\})$.

number of items, it is built up level-by-level, starting with all 1-itemsets, for which the total support is calculated by means of the P tree. According to the Apriori principle (cf. subsection 2.3.2) the support of the supersets of a certain itemset X can only exceed the threshold `minsup`, if the support of X is greater than `minsup`. Thus, if the T tree is used to find frequent itemsets, only those nodes of the second level have to be created, for which the support of the parent node is adequate. Algorithm 4 displays this process informally; this procedure ensures that the size of the tree is only linear in the size of the number of frequent itemsets.

```
1 set k = 1;
2 build level k in the T tree according to its structure;
3 calculate the support of every node in level k by means of the P tree;
4 remove any node in level k which support is lower than minsup;
5 increase k by 1;
6 build level k in the T tree according to its structure (just nodes that have a
  parent);
7 repeat line 3-7 until no new nodes are adequately supported;
```

Algorithm 4: Informal T tree creation algorithm.

ET Tree The *ET Tree* (which denotes *extended target tree*) is an extended version of the T tree. The main difference is the fact, that all nodes contain a complete list of timestamps of the accordant itemset. It is built in almost the same manner as the T tree (cf. algorithm 4), but without itemset pruning (line 4), because otherwise it would not be possible to find TARs, which standard support is too small for being discovered. Accordingly, there are three possibilities for discovering the different kinds of TARs introduced in section 4.2:

i) decrease `minsup` to a smaller value in order to find a larger set of frequent itemsets, which can be analyzed for temporal coherence (with the original larger value of `minsup` for an itemset to be large),

ii) decrease `minsup` to a minimum value (an itemset has to occur at least once in the database) and abort the algorithm at a certain level `k-max` in the tree. The resulting set of frequent itemsets can be analyzed for temporal coherence (with the original larger value of `minsup` for an itemset to be large), or

iii) leave everything as it is and discover only TARs of which the support is high enough in the standard sense.

Obviously, the adjustment of the parameters `minsup` and `k-max` directly effects the size of the ET tree and the number and kinds of TARs, which can be discovered.

4.4.2 Implementation

In this subsection we present some details about the implementation of our approach as a prototype system, which allows mining different kinds of TARs by using the EP and ET tree, and we explicate some further interesting points.

Our whole prototype system is implemented in Java. Directly converting the idea explained in the subsection 4.4.1, it consists of the following three major components:

- a *tree maintenance component*, which is designed in order to read the database, construct the EP tree and provide the possibility to construct the ET tree with several options, among others the adjustment of `minsup` and `k-max` as discussed at the end of the preceding subsection,
- a *temporal analysis component*, which handles inputs in form of a T tree or itemsets with a complete list of their timestamps in order to discover standard, cyclic, lifespan- and calendar-based ARs, and
- a *main component*, which manages the other two component. This main component offers several heuristics for the combination of the parameters, and is capable of processing optional “hints” given by the user.

The *tree maintenance component* is based on an implementation [Joe08] of the Apriori-TFP algorithm [CGL04] for standard ARM, extended by the treatment of temporal information as described in subsection 4.4.1. Simply spoken, the input of the *temporal analysis component* is an itemset together with its complete list of timestamps. On this base the calculation of the temporal support as described in section 4.2 is straightforward. The support notions for the discovery of cyclic and calendar-based TARs are implemented by making use of sliding window techniques with several different parameters (e.g. window size, step size, number of regarded windows per step, and sliding window form).

The *main component* manages the two other components and provides them with their parameters. The parameters can be adjusted by the user, or by using heuristics in order to combine mining of the three different kinds of TARs, and the user can also optionally enter some "hints" about the range of the parameters. The parameters play an important role what kinds of ARs can be discovered and how many space and time is required for building and processing the trees. The most important parameters in this relation are `minsup` and `k-max`, as already noted at the end of subsection 4.4.1, and thus the main component tries to find a balance between effort and finding every possible AR. Finding every possible AR exceeding a given support is possible in a small database, but if the dataset grows bigger this task should be constrained to certain kinds of rules or granularities, because otherwise the runtime will be rather long and the needed space rather large.

4.4.3 Evaluation

This section presents the evaluation of our tree-based approach for mining several kinds of TARs with respect to different parameters and in comparison to the original approaches for mining standard [AIS93], cyclic [ORS98], calendar- [LNWJ01] and lifespan-based ARs [AR00].

Unfortunately, up to now there are only very few real life market basket datasets available to the public, which contain timestamps (what is a natural precondition for TARM). We conducted experiments with a small set of real life market basket data from a supermarket (which, by the way seemed to yield results quite comparable to the result on the largeer synthetical data as presented in the following), but from or point of view these results are less meaningful due to the small size of this data. In addition to that, real life datasets might contain interferences, e.g. through seasons or mode, and thus do not provide a totally controllable testing ground (especially with regard to the three kinds of TARs our approach is intended to discover), whereas the use of synthetical data created by an appropriate dataset generator enables us to remain in total control of every phenomenon, which could occur in the generated datasets. Thus, we concentrated on evaluating our tree-based TARM approach on synthetical datasets created by TARGEN [SC09], our dataset generator which has been presented in section 4.3. The created datasets contain temporal coherences, so that algorithms for TARM can find three different kinds of TARs data, namely cyclic, calendar-based and lifespan-based ARs (depending on the adjustment of the appropriate parameters), and are thus perfectly suitable for testing our approach.

Altogether, we set up five series of experiments, which were conducted on an Intel Core 2 Duo 3.0 GHz desktop PC running Windows Vista with 3 GB of main memory.

In the first four series, we used our tree-based approach for discovering TARs with varying parameters. Series 1-3 exactly follow the three possibilities stated at the end of subsection 4.4.1: in

- *series 1*, we set `minsup` to a smaller value than normal in order to find a larger set of frequent itemsets, which are analyzed for temporal coherence with the original larger value of `minsup`. In
- *series 2*, we set `minsup` to the minimum value (an itemset has to occur at least once in the database) and we pruned the T tree at a certain level `k-max` (where again the original larger value of `minsup` is used as constraint for an itemset to be large), and in
- *series 3*, we used a "normal" value for `minsup`, i.e. a value for standard ARM.

Table 4.3 states the results of all experiments on a large conventional dataset in terms of runtime and number of discovered TARs. The comparison of the sets of resulting association rules from series 1-3 showed, that our approach is able to discover every standard AR with the parameter settings from series 1, second run of series 2 and series 3. As expected, some of the standard ARs have been missed in the first run of series 2 due to the restriction of the size of the underlying frequent itemsets given by `k-max` (it turned out, that the maximum length of frequent itemsets in this configuration is 6). In series 3 only TARs were found, which could also be found by standard non-temporal ARM algorithms (due to the value for `minsup`). The results also made clear, that with decreasing `minsup` more TARs are found (1,949 TARs with `minsup`$= 0.05$ in series 3; 4,313 TARs with `minsup`$= 0.01$); but only in the second run of series 2 all contained TARs (9,956) have been found, since `k-max`$= 6$ equals the maximal size of the underlying frequent itemsets (of course with runtime increasing with `k-max`).

In another series of experiments, namely in

- *series 4*, we used the heuristics mentioned in section 4.4.2 in order to speed-up the discovery of TAR,

which results in a massively decreased runtime, but also to some TARs which were not discovered (approx. 8.7%, cf. table 4.3).

As fifth series and in order to compare the results of our tree-based approach with the original approaches, we reimplemented the algorithms proposed in [AIS93, AR00, ORS98, LNWJ01] for mining standard, lifespan-based, cyclic and calendar-based ARs in Java as well, and run them on the same dataset. We refer to this as

experiment	accordant parameters	runtime (seconds)	number of discovered TARs
series 1	`minsup`= 0.01	32.241	4,313
series 2 (run 1)	`minsup`= $\frac{1}{10{,}000}$, `k-max`= 4	42.198	7,281
series 2 (run 2)	`minsup`= $\frac{1}{10{,}000}$, `k-max`= 6	49.189	9,956
series 3	`minsup`= 0.05	19.500	1,949
series 4 (heuristics)	`minsup`= 0.05	35.231	9,157
sequential algorithm	`minsup`= 0.05	122.400	9,956

Table 4.3: Evaluation results for a conventional dataset (containing 10,000 transactions, 1,000 items, average size of large itemsets is 4). Experiments with this dataset showed that a reasonable `minsup` value for standard ARM is 0.05.

- *sequential algorithm*, which sequentially runs the four original algorithm for discovering all kinds of TARs.

Comparing the runtime of our tree-based approach in table 4.3 (series 2 run 2: 49.189 seconds), which found all possible TARs, with the runtime of the sequential algorithm (122.4 seconds), we can conclude that our approach clearly outperforms the original approaches. If we are willing to neglect some TARs (e.g. 8,7% as in series 4), which is possible in many domains, the speed-up of our approach is again greater (19,5 seconds).

4.4.4 Conclusions and Directions for Future Work

In this section, we presented a tree-based approach for mining standard and several kinds of temporal ARs (cyclic, lifespan- and calendar-based TARs), which makes use of two new tree structures, namely the EP and the ET tree. These trees provide the basis for a more efficient discovery of several kinds of different TARs as presented in section 4.2. We shortly described our implementation of a prototype system, and presented an evaluation of our approach on a synthetical dataset, which proved that our approach works very well.

There are some issues left which have to be examined in more depth in order to enhance the basic idea and thus our prototype system. First of all, at the moment our approach allows to detect the three main kinds of TARs, namely cyclic, temporal support based and calendar-based TARs. In section 4.2 we presented some more interesting kinds of TARs (e.g. sequential patterns and basic TARs of the from $X \Rightarrow^T Y$), of which the integration in our approach would be an enrichment. In addition to that, pruning techniques especially regarding the construction of the T tree could make our approach even more efficient. Another direction for future research is the adaption of other tree structures for our approach. The FP tree [HPY00] for example is used to speed up standard ARM as well as the P and the T tree, and thus an adapted version could possibly be an alternative to our proposed tree structures. In order to extend the evaluation, test on further large datasets, which contain more than the three kinds of temporal coherences modeled by our dataset generator TARGEN, or even on datasets from real life (which are not totally controllable), would be very interesting.

5 Temporal Association Rule Mining applied to Time Series Analysis

The discovery of certain patterns, which are characteristic for time series and thus have a special meaning, is an important task in the area of classical time series analysis (cf. section 3.3), and similarly, the discovery of certain pattern, namely of association rules, is an important task in the analysis of transactional data (see section 2.3). Since transactional data with temporal information can be regarded as a special case of (multidimensional) time series, and since much research has been done in the field of temporal association rule mining (cf. chapter 4), it is a natural consequence to carry these results and techniques to the analysis of time series which are not limited to transactional data, as long as these techniques are adequate.

This chapter addresses the issue of analyzing time series with temporal association rule mining techniques. Since originally association rule mining was developed for the analysis of transactional data, as it occurs for instance in market basket analysis, algorithms and time series have to be adapted in order to apply these techniques also gainfully to the analysis of "normal" time series. Continuous time series of different origins can be discretized in order to mine several temporal association rules, which reveals interesting coherences in one and between pairs of time series. Depending on the domain, the knowledge about these coherences can be used for several purposes, e.g. for the prediction of future values of time series. We explain in detail how some of the most interesting kinds of temporal association rules mining techniques - namely for discovering standard association rules (as presented in section 2.3), basic temporal association rules [DLM^{+}98], sequential patterns [AS95], cyclic [ORS98] and/or calendar-based association rules [LNWJ01] (as presented in section 4.2) - can be transfered and applied to continuous time series and present an prototype implementation. In order to show its functionality and the influence of parameters, we evaluate this approach on two real-life datasets containing river level measurements and stock data.

The rest of this chapter is organized as follows: Section 5.1 briefly presents the background of TARM applied to time series analysis, integrates previous work in this context, and discusses some further points of interest. The next section explicates the application of the temporal association rule mining approaches to time series analysis in detail. After that, we present in section 5.3 the basic flow and some implementational details of our system for mining temporal association rules in time series, followed by an evaluation on river level measurements and stock data in section 5.4. Finally, in section 5.5 we present our conclusion and identify directions for future work.

5.1 Background and Related Work

Association rule mining is the semi-automatic process of discovering interesting relationships in huge amounts of data (cf. section 2.3 for more details). Originally, it was designed for a very complex class of time series, namely for time series containing transactions, as they appear for instance in market basket analysis. The aim of temporal association rule mining is the discovery of interesting temporal particularities, which can be contained in transactional data (cf. sections 4.1 and 4.2). Since already much work has been invested in the discovery of standard and temporal association rules, it is a logical consequence to transfer some of these techniques to "normal" time series (we assume "normal" time series to contain continuous numerical values), in order to reveal interesting relationships in or between them with these methods efficiently too.

An essential first step for doing this is to transform the time series in an appropriate format to which the basic ideas of association rule mining can be applied. This transformation can be done by discretizing the continuous time series, e.g. with by the clustering-based approach presented in [DLM^{+}98] or SAX [LKLcC03] (cf. chapter 3). Formally, a continuous time series s is transformed into a discrete representation $D(s) = a_1a_2...a_j$, with typically $j \ll n$ and $a_i \in \Sigma$, where Σ is an arbitrary alphabet, by several methods. A positive side-effect of this transformation is the complexity and dimensionality reduction, which is very useful when dealing with large time series with very high granularities.

One of the first approaches which directly uses association rule mining for time series analysis is [DLM^{+}98]. The authors present the former mentioned clustering-based discretization method and the basic temporal rule format $A \Rightarrow^T B$ mentioned in section 4.2. A very similar approach for the discovery of "sequential association rules with time lags" was proposed in [HDT02]. An approach, that uses SAX as discretization method instead of the clustering-based method from [DLM+98], is [WN06], which apart from that adapts [DLM+98] for mining association rules in the basic temporal rule format. A different approach is [Höp02a], where the discretization of the time series is done by scale-space filtering [Lin94], after which state sequences mining is applied in order to discover relationships between intervals.

5.2 TARM Applied to Time Series Analysis

This section explicates the application of some temporal association rule mining approaches (basic temporal association rules, sequential patterns, cyclic and/or calendar-based association rules) to the area of time series analysis in detail.

When applying (temporal) association rule mining to a non-transactional database of discretized time series, the definitions have to be adapted, as will be shown in the following. The notion of absolute support of a symbol a from the discretization alphabet Σ, can be defined as the frequency of a in the discretized time series $D(s)$. For simplicity, we regard the discretized time series s as string $D(s) = a_1 a_2 ... a_j$ in order to define the notions more formally. Then, *absolute* and *relative support of a symbol* $a \in \Sigma$ are given by

$$sup_a(a) = |\{i \mid a_i = a \wedge 1 \leq i \leq j\}| \text{ and}$$

$$sup_r(a) = \frac{sup_a(a)}{j},$$

where j is the length of the discretization string $D(s)$. Analogously, absolute and relative support of a string $A \in \Sigma^+$ can be defined as

$$sup_a(A) = \left| \left\{ \; i \;\; | \;\; [D(s)]_i^{i+|A|-1} = A \; \right\} \right| \text{ and}$$

$$sup_r(A) = \frac{sup_a(A)}{j - |A| + 1},$$

with $1 \leq i \leq (j - |A| + 1)$, where $|A|$ denotes the length of string A and $[D(s)]_b^e$ the substring of $D(s)$, beginning at position b and ending at e (e.g. $[\text{String}]_1^3 = \text{Str}$).

In the analysis of a time series s the meaning of an *association rule* $A \Rightarrow B$ is, that if string A occurs somewhere in $D(s)$, it will most likely be followed by B. Thus the *absolute* and the *relative support of an association rule* $A \Rightarrow B$ is defined as

$$sup_a(A \Rightarrow B) = sup_a(AB) \text{ and}$$

$$sup_r(A \Rightarrow B) = \frac{sup_a(A \Rightarrow B)}{j - |AB| + 1},$$

where the denominator of the latter equals the number of possible occurrences of $A \Rightarrow B$ (AB denotes the string concatenation of A and B). With this support definition, the *confidence* of an association rule in a time series is defined exactly like in a transactional database, namely as

$$conf(A \Rightarrow B) = \frac{sup_a(A \Rightarrow B)}{sup_a(A)},$$

which is the number of occurrences of AB divided by the number of occurrences of A.

Depending on the size of the time series database, a huge number of association rules can be discovered. If the parameter `minsup` and `minconf` are set properly, this number will be reduced to a smaller number of more important association rules, through which a domain expert should browse in order to find interesting and unexpected association rules. Since the number of the remaining strong association rules might still be very large, the expert can be guided by the use of interestingness measures for association rules (cf. for instance [TSK05]). We decided to use the *J measure*, which was proposed in [SG92], because it provides a useful and sound method for ranking rules in a manner which trades-off rule frequency and confidence [DLM+98]. The J measure for an association rule $A \Rightarrow B$ can be defined as

$$J(A; B) = P(A) \left[P(B \mid A) \log \tfrac{P(B|A)}{P(B)} + (1 - P(B \mid A)) \log \tfrac{1-P(B|A)}{1-P(B)} \right],$$

where $P(A)$ and $P(B)$ denote the probability of string A (or respectively string B) occurring at a random position in $D(s)$ (i.e. the relative support of A or B) and $P(B \mid A)$ the probability of B occurring at a random position in $D(s)$ with A preceding (i.e. the confidence of $A \Rightarrow B$).

There are many different kinds of temporal association rules (i.e. association rules with special regard to the temporal component) one could think of, for instance from events that happen sequentially [AS95], cyclically [ORS98], periodically [HDY99] or in special times which can be described by calendar-based patterns [LNWJ01]. All these rules introduced in section 4.2 require their own adapted support definition in order to be applied in time series analysis, which will be given in the following.

5.2.1 Basic Temporal Association Rules

The simplest temporal association rule format, to which we refer as *simple* or *basic temporal association rule*, is “if A occurs, then B occurs within time T” [DLM+98], written as $A \Rightarrow^T B$ for two strings A and B and a duration T. (T can either be seen as duration in an arbitrary time unit, or, as in the following, as symbol distance in the string representation of the discretized time series.) The absolute support of such an association rule can be stated as

$$sup_a(A \Rightarrow^T B) = F(A, B, T),$$

where $F(A, B, T)$ is defined as

$$F(A, B, T) = \left| \left\{ i \mid [D(s)]_i^{i+|A|-1} = A \wedge B \in [D(s)]_{i+w+1}^{i+w+T-1} \right\} \right|,$$

with $1 \leq i \leq (j - |AB| + 1)$. $F(A, B, T)$ is the number of occurrences of A, that are followed by B in the distance of T symbols. (Amongst others, the parameter w is a tribute to the clustering based discretization method used in our approach, confer chapter 3. Since two subsequent symbols a_i and a_{i+1} are derived from two overlapping sliding windows, a_i and a_{i+1} are strongly correlated. Thus the optional distance parameter w determines the minimal gap, that should lie between the occurrences of A and B in the discretization string.) And finally, the relative support of $A \Rightarrow^T B$ can be defined as

$$sup_r(A \Rightarrow^T B) = \frac{sup_a(A \Rightarrow^T B)}{j - |AB| + 1 - w}.$$

Confidence is defined as usual, i.e.

$$conf(A \Rightarrow^T B) = \frac{sup_a(A \Rightarrow^T B)}{sup_a(A)},$$

and the J measure of $A \Rightarrow^T B$ is given by $J(A; B_T)$ (detailed definition see above), where $P(B_T)$ denotes the probability of at least one B occurring in a randomly chosen window of size T in $D(s)$ (i.e. the number of windows of size T containing B divided by the number of all possible windows, which is given by $j - T + 1$) and $P(B_T \mid A)$ the probability of at least on B occurring in a randomly chosen window of size T, which starts with A (i.e. $F(A, B, T)$ divided by the absolute support of A.

The format of the temporal association rule $A \Rightarrow^T B$ is quite simple, but highly expandable, as contemplated in [DLM+98]: A and B may come from one time series discretization $D(s)$, which allows *intra time serial* association rule mining, or A may come from time series s_1 and B from another time series s_2, which leads to pairwise *inter time serial* association rule mining between $D(s_1)$ and $D(s_2)$. The adaptation to the algorithms for calculating support and confidence values are straight forward. Note, that the rule format $A \Rightarrow^T B$ can be seen as generalization of the standard association rule format $A \Rightarrow B$, as introduced at the beginning of this subsection (if parameter T is set to zero).

5.2.2 Sequential Patterns

Sequential patterns, which have been proposed in [AS95], are temporal association rules derived from events that happen sequentially. An already mentioned example from market basket analysis for such a rule might be derived from the following observation made in a video rental business: People who rent the DVD "The Fellowship of the Ring" and then "The Two Towers", most likely rent the DVD with the third part of the "Lord of the Rings" trilogy, namely "The Return of the King", thereafter too. Sequential patterns, as proposed in [AS95], just state that if someone rents the first and then the second movie, he will most likely rent the third movie afterwards too, but they do not state when this will happen. Since this information is interesting too, we decided to use the extension of the basic rule format $A \Rightarrow^T B$ for sequential pattern mining: "if A_1 and A_2 and ... and A_k occur within V units of time, then B occurs within time T", written as $A_1, A_2, ..., A_k \Rightarrow^{V,T} B$ [DLM+98]. Again, A_i and B may come from one or different time series discretization, allowing intra and inter time serial sequential pattern mining.

The definitions of support and confidence for all possible intra and inter time serial sequential pattern rules between one and several time series follow the basic idea behind $A \Rightarrow^T B$, thus as an example here we only give the definition for sequential pattern rule mining for $k = 2$, with A_1 and A_2 coming from time series s_1 and B coming from time series s_2: The absolute support of such a temporal association rule $A_1, A_2 \Rightarrow^{V,T} B$ is defined as

$$sup_a(A_1, A_2 \Rightarrow^{V,T} B) = F(A_1, A_2, B, V, T),$$

which states the number of occurrences of A_1 and A_2 in time series s_1 within time V, which are followed by B in s_2 in a maximal temporal distance of T (from the beginning of A_1). Formally, $F(A_1, A_2, B, V, T)$ is given by

$$\left|\left\{ i \mid [D(s_1)]_i^{i+|A_1|-1} = A_1 \quad \wedge \quad A_2 \in [D(s_1)]_{i+|A_1|}^{i+V-1} \quad \wedge \quad B \in [D(s_2)]_{x+|A_2|}^{i+T-1} \right\}\right|,$$

where x denotes the position of the beginning of A_2 in s_1. (Note, that the given definition forbids overlapping of A_1, A_2 and B. Furthermore, we removed the parameter w, which was introduced in the definition of $F(A, B, T)$ above, in the definition of $F(A_1, A_2, B, V, T)$ due to lucidity.) The relative support and the confidence of $A_1, A_2 \Rightarrow^{V,T} B$ are given by

$$sup_r(A_1, A_2 \Rightarrow^{V,T} B) = \frac{sup_a(A_1, A_2 \Rightarrow^{V,T} B)}{|D(s_2)| - |A_1 A_2 B|} \text{ and}$$

$$conf(A_1, A_2 \Rightarrow^{V,T} B) = \frac{sup_a(A_1, A_2 \Rightarrow^{V,T} B)}{F(A_1, A_2, V)},$$

and finally, the J measure for the rule $A_1, A_2 \Rightarrow^{V,T} B$ can be stated as

$$J(A_1 \Rightarrow^V A_2; B_T),$$

where $P(B_T)$ denotes the probability of at least one B occurring in a randomly chosen window of size T in $D(s_2)$, $P(A_1 \Rightarrow^V A_2)$ the probability of association rule $A_1 \Rightarrow^V A_2$ occurring at a random position in $D(s_1)$ (i.e. the relative support of $A_1 \Rightarrow^V A_2$) and $P(B_T \mid A_1 \Rightarrow^V A_2)$ denoting the probability of at least on B occurring in a randomly chosen window of size T in $D(s_2)$, which starts with A_1 in $D(s_1)$, followed by A_2 within V time units (i.e. $F(A_1, A_2, B, V, T)$ divided by the absolute support of $A_1 \Rightarrow^V A_2$).

5.2.3 Cyclic and Calendar-Based Association Rules

Another interesting temporal association rule format was proposed in [ORS98]. A rule is called *cyclic association rule*, if it represents regular cyclic variations over time. A famous example from economics is the so called pig cycle, which describes cyclic fluctuations of supply and prices. In Europe, it has first been discovered in 1927 by Hanau [Han27] in the pig market, where every 18-24 month a big supply of pork for low prices is followed by a low supply for high prices. The support of an accordant association rule $\{\text{big supply of pork}, \text{low pork prices}\} \Rightarrow \{\text{low supply of pork}, \text{high pork prices}\}$ will be probably relatively low during the whole time, but if the time every 18-24 month (and the corresponding database segment or part of the time series, respectively) is regarded specifically, such a rule will surely have a support high enough for being discovered.

For a more formal definition and the transfer to time series analysis we assume that time is given in a fixed unit τ (for instance in month), and that the i-th time interval ($i \geq 0$) is denoted by τ^i, which corresponds to the time interval $[i \cdot \tau, (i+1) \cdot \tau]$. Let $D(s)^i$ denote the concatenation of the symbols of $D(s)$, which corresponds to τ^i, and refer to it as time segment i. Then, the absolute cyclic support $cycSup_a^i$ of an association rule $A \Rightarrow B$ in the time segment i is given by

$$cycSup_a^i(A \Rightarrow B) = \left| \left\{ \; i \;\; \middle| \;\; \left[D(s)^i\right]_i^{i+|AB|-1} = AB \; \right\} \right|,$$

i.e. the number of occurrences of $A \Rightarrow B$ in the time segment i [1], and the relative cyclic support is defined as

$$cycSup_r^i(A \Rightarrow B) = \frac{cycSup_a^i(A \Rightarrow B)}{|D(s)^i| - |AB| + 1}.$$

[1]This definition of cyclic support regards standard association rules, i.e. association rules where a string A is directly followed by B. Of course, it can simply be altered in order to discover as well basic temporal association rules as introduced in subsection 5.2.1 in the specified database segments.

The confidence measure for $D(s)^i$ is defined analogously to the confidence measures of the previous temporal association rules. A cycle c is a tuple (l, o), which consists of a length l and an offset o, $0 \leq o \leq l$, both given in time units. An association rule is said to have a cycle $c = (l, o)$, if it holds in every l-th time unit starting with time unit τ_o. An association rule which has a cycle is called *cyclic.* The pig cycle mention above for instance could be expressed by the cycles $c_1 = (24, 18)$, $c_2 = (24, 19)$, ..., $c_6 = (23, 24)$, which denote "the 18th month every 24 month", "the 19th month every 24 month" and so on, which corresponds to the time between every 18th and 24th month, where the pig cycle was observed.

An association rule format, which is quite similar to cyclic association rules, are calendar-based association rules [LNWJ01]. A *calendar-based association rule* is an association rule, that is valid in certain time intervals, which are specified by a former defined calendar schema. An example for such a schema might be (year, month, day); an example for a special time interval in this calendar schema might be the triple $(*, 11, *)$, where $*$ denotes every arbitrary integer in the domain of the accordant attribute, in this case day and year. $(*, 11, *)$ for instance represents every day in November of every year, which is probably the time with the most traffic jams (at least in Germany). As well as in the foregoing approaches, support is calculated by dividing the number of occurrences of a certain association rule by the the maximal possible number of occurrences of a rule of same length. Thus, we omit the formal definitions of calendar-based support, refer to [LNWJ01] and give one more example for clarification instead: in the calendar-based schema (month, day, hour), the beer and potato crisps rule mentioned at the beginning of section 4.1 could hold in the time represented by $(*, *, 19) \cup (*, *, 20)$, because this pattern denotes the 19th and 20th hour of every day in every month, which refers to the time from 7 to 9PM (where relatively to the time supposably more customers buy beer and crisps together than in comparison to the whole day).

Obviously, the intersection of correlations, which can be expressed by calendar-based association rules, and of correlations, which can be expressed by cyclic association rules, is very large, thus it is a matter flavor, how these rules are presented.

5.3 Implementation

We implemented our system for mining temporal association rules in time series in Java using the Eclipse IDE. Since algorithms for several data mining purposes are already implemented in Java, and since some of them are freely available, we used the SAX implementation from Pavel Senin [Pav12] for discretizing time series and

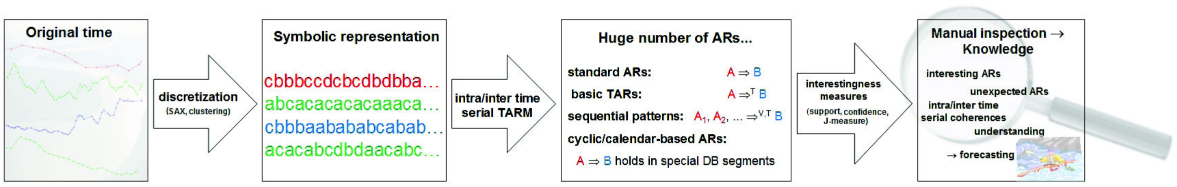

Original time
discretization
(SAX, clustering)
Symbolic representation
cbbbccdcbcdbdbba…
abcacacacacaaaca…
cbbbaababababcabab…
acacabcdbdaacabc…
intra/inter time serial TARM
Huge number of ARs...
standard ARs: $A \Rightarrow B$
basic TARs: $A \Rightarrow^{T} B$
sequential patterns: $A_1, A_2, \ldots \Rightarrow^{V,T} B$
cyclic/calendar-based ARs:
$A \Rightarrow B$ holds in special DB segments
interestingness measures
(support, confidence, J-measure)
Manual inspection → Knowledge
interesting ARs
unexpected ARs
intra/inter time serial coherences
understanding
→ forecasting

Figure 5.1: Basic flow of our prototype system for temporal association rule mining in time series.

the k-means implementation from Weka [HFH+09] for obtaining the clustering-based representations.

The basic flow of our system is the following: first, the time series have to be obtained and processed from a source. In our case, the data is stored in a MySQL database, from which it easily can be retrieved in adequate granularities. After an optional step of storing every time series derived from the database on disk or loading it from there, the time series are discretized in three ways by the two methods presented in chapter 3, which results in three discretized time series representation for each time series: the first discretization is obtained by SAX, the second by the clustering-based approach, where the basic shapes are derived from each time series on their own, and the third is derived by the clustering-based approach, where the basic shapes are derived from all time series together (which makes more sense for the interpretation of inter time serial association rules). Afterwards the discovery of the different kinds of temporal association rules (as presented in subsection 5.2) is started for each type independently and the results are displayed interactively to the user for inspection. The discovered association rules can be used for several purposes, e.g. for time series prediction (if the antecedent of a rule with a high confidence occurs, the consequent will most likely occur too; cf. section 5.4 for an example). Figure 5.1 illustrates the basic flow of our prototype system.

The apriori algorithm, which was presented in [AIS93], is the quasi-standard algorithm for efficiently mining association rules in transactional databases (cf. subsection 2.3.2). It bases on a lexicographical ordering of the items in each transaction, but since this ordering would destroy fundamental temporal relations, it cannot be applied directly for the discovery of temporal association rules in a time series database. Thus, for efficiently finding association rules between strings in discretized time series, we can only rely on two basic concepts of this algorithm, which are the monotonicity property and the candidate generation. The monotonicity property applied to time series states, that a string can only be frequent, if every substring is frequent too. Thus, we do not have to search for association rules containing every possible string in the discretizations, but only for strings, that consist of smaller frequent strings. As first step in discovering temporal association rules, we are determining every symbol, which is frequent (according to the parameter `minsup`). The concatenations of the pairs of these symbols form the candidate set C_2, from which all frequent strings of length 2 are derived, by checking if support is higher than `minsup`. All frequent strings of length $k-1$ form the set L_{k-1}, from which again the candidate set C_k is generated by adding every string of length k, which can be generated from merging two strings from L_{k-1}, which conform in $k-2$ succeeding characters. L_k again is generated from C_k by checking the `minsup` constraint. After determining the finite set $\bigcup_k L_k$ of all frequent strings, the temporal association rules can simply be generated by counting their frequencies, as explicated in subsection 5.2.

5.4 Evaluation

We tested and evaluated our prototype system on two large datasets containing river level measurements and stock data. At the time of the evaluation, the river level measurements consisted of 75,000,000 tuples from 394 river stations, which were obtained from the environmental agencies of the German federal states North Rhine-Westphalia (`http://www.lanuv.nrw.de`), Hesse (`www.hlug.de`) and Rhineland-Palatinate (`www.luwg.rlp.de`); the stock data consisted of 47,000 tuples of daily closing prices from 32 companies traded at the Nasdaq stock market from 2006-2009, obtained via Dirk Eddelbuettel's tool beancounter [Dir12] from the Yahoo finance server (`http://finance.yahoo.com`). All experiments were carried out on an Intel Core 2 Duo 3.0 GHz desktop PC running Windows Vista with 3 GB of main memory.

We conducted several experiments with varying parameters. As first experiment we investigated the influence of the the time series length on the runtime of the discovery process. Since we are only interested in the discovery of temporal association rules and not in the influence on the discretization techniques, we omit the detail related to the discretizations. The upper left plot of figure 5.2 displays time series lengths versus runtime of our prototype system for discovering intra time serial association rules. As expected, the runtime increases more than linearly with the length of the time series.

The influence of the parameter `minSup` on the runtime was measured in another series of experiment, the bottom left plot of figure displays the accordant results for inter time serial association rule mining, which show that the runtime decreases with an increasing value for `minSup`. Obviously, this is a logical consequence, since also the number of discovered association rules decreases with increasing `minSup` value, as shown in the upper right plot. The parameter `minConf` shows a similar behavior, as can be stated from the bottom right plot: the higher this value is set, the less association rules are discovered.

The comparison of the runtimes of intra and inter time serial association rule mining show that the runtime of the discovery of inter time serial association rules is

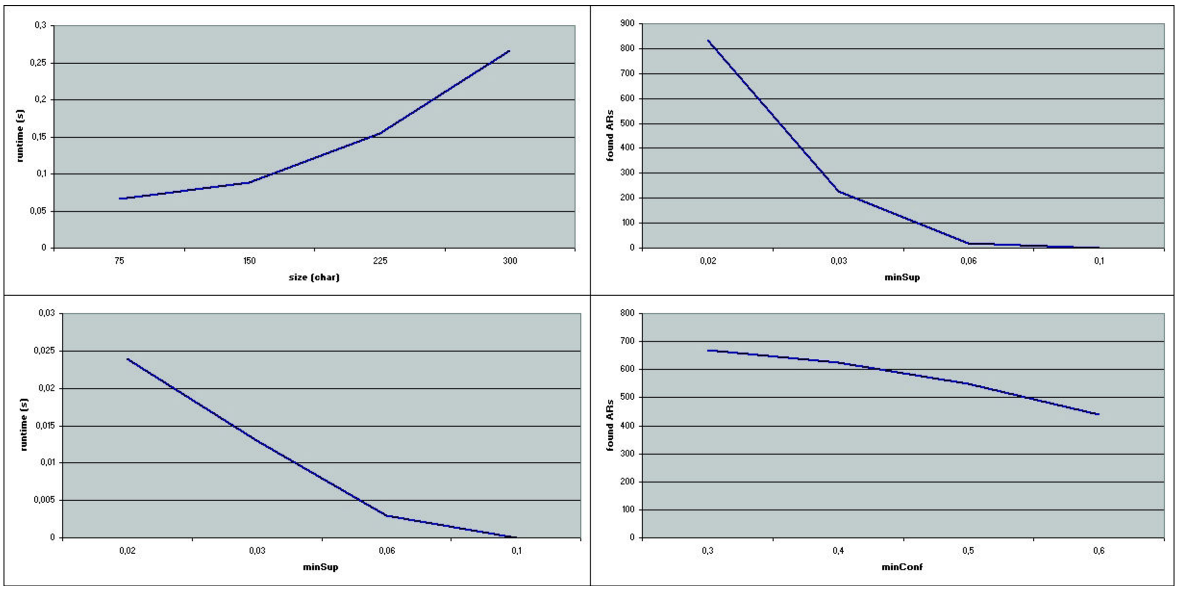


Figure 5.2: Experimental results: time series size vs. runtime for intra time serial ARM (upper left, 1.5% minSup, 50% minConf), minSup vs. found ARs and runtime for inter time serial ARM (upper right and bottom left, 50% minConf) and minConf vs. found ARs for inter time serial ARM (bottom right, 2% minSup).

approximately quadratic in the number of time series, which can be explained by the fact that the time series are compared pairwise. Setting `minsup` and `minconf` to fixed values and changing only T (the time span between the occurrence of antecedent and consequent of basic temporal association rules), it shows that the number of discovered association rules increases with T. Increasing w (the parameter determining the string distance between antecedent and consequent of a rule) yields a decrease in the number of discovered association rules.

Two examples of the outcome of the intra and inter time serial association rule mining process are given in figure 5.3. The left figure shows the plot of an intra time serial association rule obtained from the SAX discretization of the time series derived from the river level measurements of the river Rhine at Düsseldorf. In this case, the meaning of the rule $bbbbbbb \Rightarrow^T c$ could be, that after a decline of the water level over about one week at this station a rise has to be expected within the next two weeks. In contrast to the former mentioned rule, the meaning of the inter time serial association rule $bbbb \Rightarrow^T bbbbb$ obtained from the the two independently clustered and discretized time series derived from the river level stations Düsseldorf and Cologne (Köln, right figure) remains unclear, which underlines the need for domain specific knowledge for the parameter selection and the interpretation of the resulting rules in the different applications.

Figure 5.4 shows a map, which displays the rivers Rhine (“Rhein”, the big blue

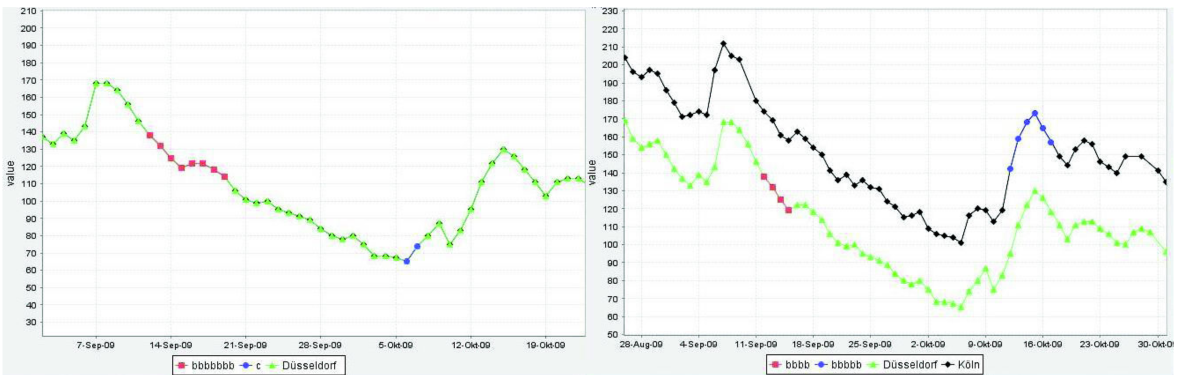


Figure 5.3: Example of an intra time serial association rule (left) and an inter time serial association rules (right) of the form $A \Rightarrow^T B$, where A is displayed red (with rectangles) and B blue (with circles).

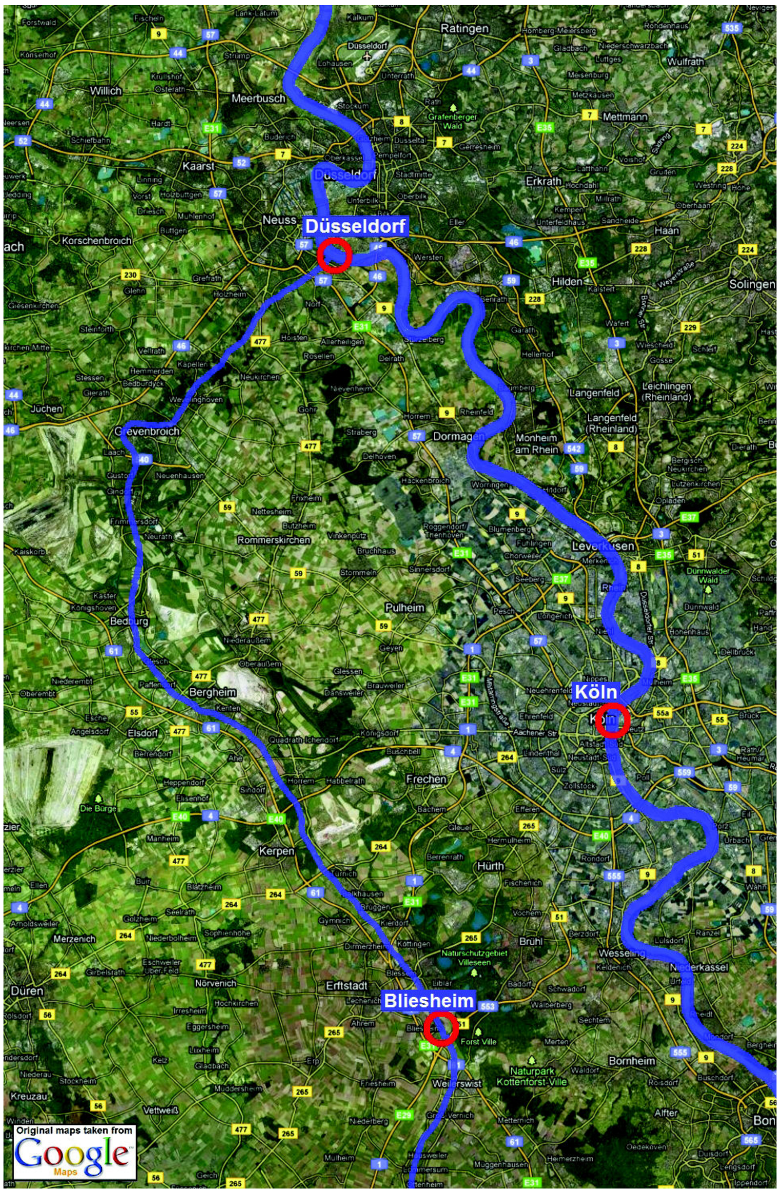


Figure 5.4: A map displaying the rivers Rhine, Erft, and the river level stations Düsseldorf, Cologne and Bliesheim.

on the right) and Erft (the smaller blue left, both flowing to the upper edge of the map/North) and the river level stations Düsseldorf, Cologne ("Köln", both located at the Rhine) and Bliesheim (at the Erft). Inter time serial temporal association rule mining on the time series derived from these stations discovered among others an association rule of the form $A_1, A_2 \Rightarrow^{V,T} B$ with a J-measure of 0.027. This association rule is one of the highest ranked rules in the result of our approach, and it reveals the coherence "if Bliesheim measures an increase of the water level, followed by an increase of the water level in Cologne, then the water level in Düsseldorf will increase a few hours later". From our point of view, this rule is indeed very interesting because it could be used for instance for predicting the river level in Düsseldorf, e.g. for floodwater warning. Figure 5.5 displays one occurrence of this association rule.

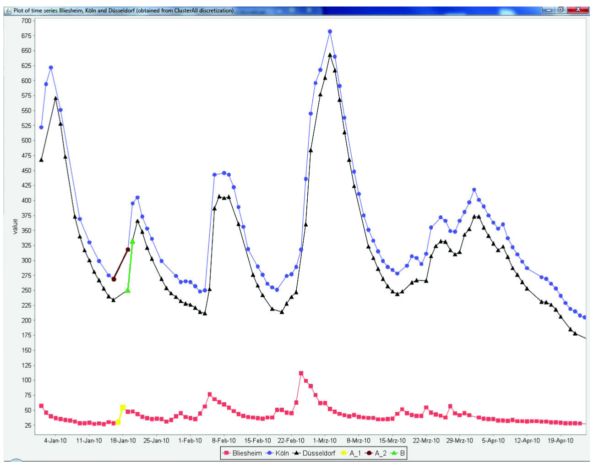


Figure 5.5: Example of an interesting inter time serial temporal association rule derived from the river level stations Düsseldorf, Cologne and Bliesheim.

5.5 Conclusion and Future Work

In the past, much effort has been invested in the research of mining standard and temporal association rules in transactional databases, but there has not been much research on association rule mining in "normal" time series. In this chapter we considered several kinds of temporal association rules (basic temporal association rules as proposed in [DLM+98], sequential patterns [AS95], cyclic [ORS98] and calendar-based association rules [LNWJ01]) and explicated consistently and in detail, how to mine these rules in continuous time series. Furthermore, we presented a prototype system, which applies these temporal association rule mining techniques to time series analysis (after discretizing the time series with SAX and two versions of the clustering-based approach from [DLM+98]), in order to reveal domain-dependent interesting coherences in one time series or between pairs of time series. We evaluated our system on two large datasets containing river levels measurements and stock data which showed the functionality and the influence of the different parameters.

Apart from integrating further kinds of TARs (e.g. interval-based approaches like [Höp02a] and [WR07]), there are some more points left for future work. First of all, further time series discretization methods have to be tested in the transformation process, for instance dynamic and deductive ones. Both discretization methods used in our approach are static in the sense that one symbol has always the same duration. An appropriate dynamic discretization might not only be interesting from the view point of data compression, but also to make basic shapes (and thus the rules) more expressive. In [KL05], clustering of time series was claimed to be not very meaningful, notwithstanding the fact, that we could not find major differences in our results using clustering- and SAX-based discretizations. Thus, further testing might reveal interesting insights. In addition to that, right now the calculation of the frequencies is done quite naively, which might be improved by the using adequate data structures and earlier pruning. An interestingness measure for association rules, which combines the two measures support and confidence, and another measure, which penalizes the overlapping of occurrences of association rules, could be very useful in addition to existing interestingness measures like the J measure. Naturally, rules with huge overlapping parts are less expressive, which is not captured by existing interestingness measures. Finally, the integration of domain knowledge (e.g. for parameter selection and rule rating) from specific application areas would surely lead to more meaningful rules.

6 Time Series Analysis and Prediction

Vast amounts of data occur nowadays nearly everywhere. A special kind of this data, which contains interesting temporal coherences, are time series. A time series is a sequence of data points, which are measured at certain typically successive points in time. Time series occur in many different areas such diverse as medicine, astronomy and geology; and the understanding of the characteristics of a certain kind of time series and the prediction on basis of this knowledge is very important for many applications (e.g. for seizure prediction [MAEL06], sleep data analysis [AGP$^+$05, SC10a], predicting sun spots [XWWL08], the El Niño phenomenon [CDK$^+$99] and many more [SS06]).

Motivated by this need for time series analysis and prediction, we developed two general approaches for the analysis and prediction of time series, namely TEMPUS [SC10c] and a motif-based approach [SC12b], which both will be presented in this chapter. These two approaches can analyze time series databases in order to find correlated time series, on which base a Hidden Markov Model can be constructed, which is used to predict one time series on basis of the other. Due to this basic idea, our systems are general in the sense that they can be applied to time series of several origins, which we are demonstrating by the example of time series containing river levels and stock data. Analyzing and predicting e.g water levels in rivers can be used for several interesting applications like boat transportation, canoeing and floodwater prediction.

While previous work in the prediction of this area has mainly been done on base of complex linear and non-linear models [PRF$^+$09, CD07], neural networks [HLN01, FDH01, BK07], fuzzy modeling [CG05] or complex physical models regarding river depth, underground particularities, drift, flow rate et cetera; our proposed approaches base on simple but interesting combinations of data mining and statistics. TEMPUS bases on derivative dynamic time warping (for finding correlations between time series) and a Hidden Markov Model (which is created on base of the correlated time series in order to predict future values); our second approach [SC12b] bases on a motif-based time series representation, which among others easily allows to find correlated time series. This motif-based time series representation and the inter-time-serial correlations discovered using this representation can be used for several purposes, e.g. again for time series prediction with the Hidden Markov Model mentioned above.

The whole chapter is organized as follows: basics about time series prediction are introduced in section 6.1, which also explains the Hidden Markov Model used for predicting time series on base of two correlated time series. Section 6.2 presents TEMPUS, our first prototype system, which detects correlations by using derivative dynamic time warping. This section contains details about the implementation of TEMPUS and an evaluation, which presents the results of our tests on a large dataset containing river level measurements (subsection 6.2.1). This evaluation showed that the combination of statistics and data mining in TEMPUS is a promising approach for analyzing and prediction time series, not only in the case of water level measurements. We conclude the section about TEMPUS with subsection 6.2.2, which also states some directions for future work.

In section 6.3 we present our second approach for the analysis and prediction of time series, which makes use of a motif-based time series representation. This motif-based representation is used to detect correlated time series pairs, on which base again the Hidden Markov Model presented in section 6.1 is created in order to predicts future values. The motif-based time series representation and the discovery of inter-time-serial correlations basing on this representation are explained in subsection 6.3.1 in detail. Implementational details of our motif-based approach are given in subsection 6.3.2, which also contains an evaluation with regard to inter-time-serial correlation discovery and time series prediction capability on two large datasets containing river level measurements and stock data. Finally, subsection 6.3.3 concludes this chapter and states future work.

6.1 Basics

A time series $s = (x_1, x_2, ...)$ is a sequence of data points x_i (in the simplest case x_i is a real number). The data points x_i can be understood as output of a function $s(i)$, where i is a number between 1 and n (n is the length of the time series s), with $s(i) = x_i$. The task of *predicting* further points of the time series s can be reformulated as finding an approximation $s'(i)$, with $s'(i) = x_i$ for $1 \leq i \leq n$, which continues the sequence past its last data point x_n. An often applied method for this is to regard the time series (or merely its discretized representation) as output of a process described by a *Hidden Markov Model* [Sch97, HN05, GV06], as we will explicate in the following.

Hidden Markov Model A *Hidden Markov Model* (HMM) is a statistical model, in which the system being modeled is assumed to be a Markov Process with unobserved state. Hidden Markov Models have been applied successfully in several different areas, e.g. in speech recognition, cryptography, protein classification and sequence alignment [RJ86, Por88, Rab89, BCP⁺93].

Formally, a HMM is defined as quintuple $H = (S, O, T, E, s)$, where $S = \{S_1, ..., S_n\}$ is the set of states and $O = \{o_1, ..., o_m\}$ the set of output symbols or observables (which can be output in a state). For our concrete purpose of predicting time series[1], we use $S = O = \Sigma$, where Σ is the the alphabet used for discretizing the time series, as will be explained below. T is the probability matrix of state transitions (i.e. the matrix which contains the probabilities for every possible transition from one state or symbol to another), E the probability matrix of emission transitions (the probabilities which symbols can be output in certain states) and s the vector containing the start probabilities for each state. Figure 6.1 illustrates a simple HMM which models the relation between two discretized time series, where the values of the time series are regarded as states and observables.

The special property of a *Hidden Markov Model* is the fact that the states are not directly visible to an observer, they are *hidden*, but the output symbols (which depend on the states) are visible. Thus, one task dealing with HMMs is to find the sequence of hidden states, which most likely generated a given output sequence. (More details about HMMs in general can be found in [RJ86, Por88, Rab89].)

In 1973, Fornay presented a dynamic programming algorithm, which exactly solves this task: the *Viterbi algorithm* [For73] for finding the sequence of hidden states, which most likely generated a given output sequence. To apply this algorithm for time series prediction, an appropriate HMM (including all parameters) has to be created, which is done as follows. As first step, the underlying continuous time series have to

[1]to be more precise, to predict one time series on base of another time series

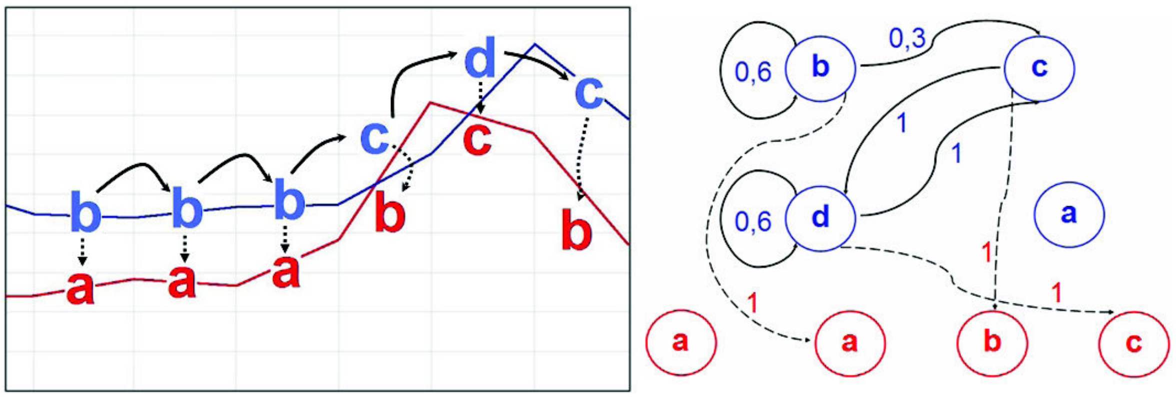


Figure 6.1: Simplified illustration of an HMM derived from the two discretized time series on the left. The upper blue time series (to which we will refer as time series a in the following) corresponds to the sequence of hidden states S (displayed as blue upper circles in the HMM on the right), the lower red time series (time series b) to the observables O (dotted red circles below). The state transitions are indicated as arrows, emissions as dotted arrows.

be discretized in order to be used as states or observables of the HMM. As already mentioned in section 3.1, a time series s can be discretized by several means into a string $D(s) = a_1 a_2 ... a_j$, with typically $j \ll n$ and $a_i \in \Sigma$, where Σ is an arbitrary alphabet which is used for the discretization.

Our HMM bases on two discretized correlated time series strings to which we refer as $D(a)$ and $D(b)$ (cf. example 4 for an illustration of two correlated time series), where the discretizations can be obtained by a simple methods using equiprobabilistic intervals, SAX or the clustering-based approach mentioned in section 3.1 (of course, further discretization methods are also possible). Then, the future development of time series a can be predicted on basis of known values from time series a and b, by considering a part of the values from time series b as visible output of a HMM, of which the states, which correspond to a part of the values from time series a, emitted these observables.

Since it is reasonable to use the same alphabet Σ for discretizing all time series, the number of observables and states equal $m = n = |\Sigma|$ as mentioned above. The transition and emission matrices are calculated on base of the discretized values of the two time series (as indicated in figure 6.1), just like the start probabilities for every state. As already mentioned, figure 6.1 displays a HMM, which models the relation between two discretized time series. In order to use such HMM for predicting time series, the relations between or the emissions from the state symbols (of time series a) and/to the output symbols (of time series b) have to be regarded as shifted, as illustrated in figure 6.2.

After all parameters have been calculated on base of the two time series in this way, the Viterbi algorithm can directly be applied for predicting future values of time series

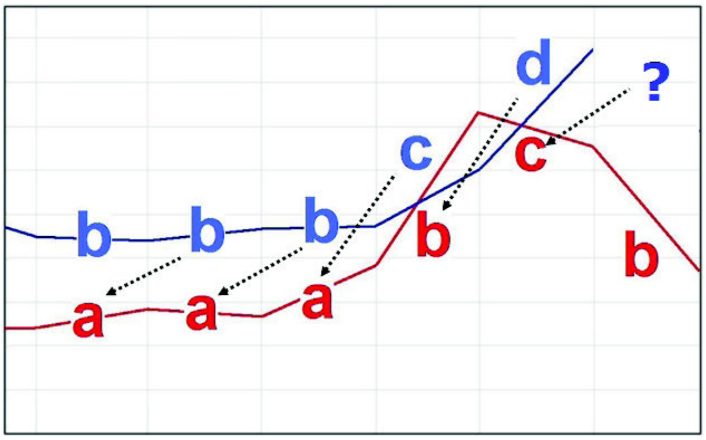


Figure 6.2: Illustration of the “shifted” relations between / emissions from S (corresponding to the hidden states and the upper blue time series, “a”) to the observables O (lower red time series, “b”). The emssions are indicated by the dotted arrows.

a, since this is equivalent to continue the sequence of hidden states (corresponding to the available symbols of time series a) which most likely generated a given output sequence (i.e. the values from time series b).

Example 4. Correlated Time Series
An example for two correlated time series in the dataset used for evaluation purposes of this chapter (sections 6.2.1 and 6.3.2) is the time series pair derived from the river stations in Düsseldorf and Kaub, two cities located at the river Rhine in Germany, which is displayed in figure 6.3. The air-line distance between the two cities is approximately 140 kilometer, and the Rhine flows from Kaub to Düsseldorf about 200 kilometer. Intuitively it is clear, that the two time series derived from the river level measurements in Düsseldorf and Kaub have to be correlated somehow, e.g. if Kaub measures a very high water level, due to the distance is will last about 24 hours until the body of water arrives in Düsseldorf, and at this time there should be a higher river level in Düsseldorf too). This correlation can be used in several ways, e.g. for predicting time series using a HMM on base of the two correlated time series, which is exactly what our two approaches presented in this chapter do.

The two approaches for analyzing and predicting of time series, which are presented in this chapter, both make use of the previously presented HMM. The basic requirement of this HMM is the correlation of the two underlying time series: the more correlated two time series are, the better is the prediction, as will be shown in subsections 6.2.1 and 6.3.2.

At this point, it is important to state that both TEMPUS and our motif-based approach do not use any domain knowledge (such as the geographical location or the physical properties of water flow in rivers) in order to discover correlated pairs of time

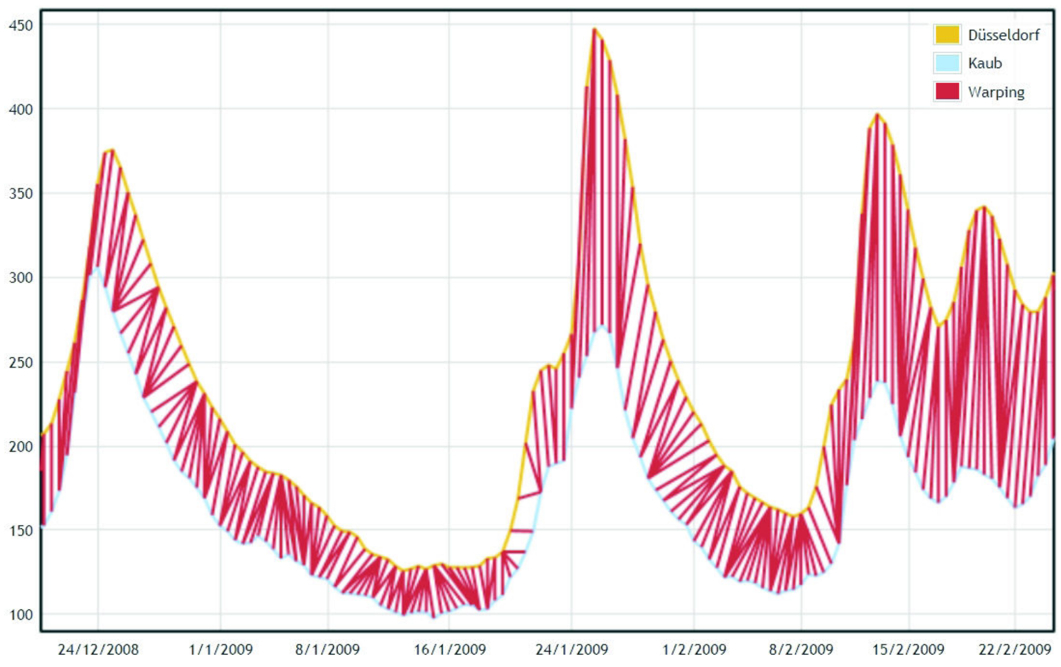


Figure 6.3: Example of two correlated time series derived from the river station in Düsseldorf (above) and Kaub. The red lines between the time series show the alignments produced by the DDTW distance, which can be used to discover correlated time series pair (e.g. as perfomed in TEMPUS; the DDTW distance between these time series is very small, which implies their correlation).

series. The correlations are merely discovered by analyzing the time series, namely by using DDTW (as employed in the next section) and hierarchically clustering a motif-based time series representation (cf. section 6.3).

6.2 TEMPUS

In this section we present TEMPUS [SC10c], a prototype system for the analysis and prediction of time series. TEMPUS is capable of analyzing and predicting time series of several origins, which is demonstrated by predicting water levels in rivers. TEMPUS analyzes time series consisting of river level measurements of numerous stations, detects correlated time series by using derivative dynamic time warping and predicts future river levels by means of the Hidden Markov Model presented in section 6.1, which is established on base of the former detected correlated time series.

We implemented TEMPUS as web application using MySQL and PHP server-side and JavaScript client-side, thus it can be accessed online[2]. TEMPUS offers several features for the analysis and prediction of time series, it displays meta-information about whole river systems and single river stations (cf. subsection 6.2.1 for the data basis), where each river station corresponds to a single time series, presents time series graphically and interactively, and displays the location of each station in a GoogleMap plugin. As core features TEMPUS calculates the Euclidean and DDTW distance between two time series, analyzes their coherence and predicts future developments by means of the HMM presented in section 6.1, which bases on two correlated time series.

Correlated time series are found by calculating the derivative dynamic time warping (DDTW) distance (cf. section 3.2) of every time series pair in the database. Pairs which have a distance smaller than a certain threshold are considered as correlated. (Originally, this was how TEMPUS found the pair Kaub/Düsseldorf from example 4, see figure 6.3). According to [KP01], the computational complexity of the DDTW distance is $O(mn)$, where m and n are the length of the time series. Optimization techniques for DTW mentioned in section 3.2 are applicable to DDTW too, but even with these techniques the calculation of the distance remains quite expensive, especially for the time series in our database (cf. subsection 6.2.1), which consisted of more than 300,000 data points for each time series (growing every day). Our database consists of river levels, which are measured every quarter hour, which is an unnecessary fine granularity with regard to meaningful river level changes. Thus our implementation uses only a part of the whole time series, which is estimated by calculating the moving average over a certain adjustable amount of data points (e.g. between 1 and 8 values per day). In addition to that it is possible to adjust the area of the time series which is used for the calculation of the distance (e.g. use only the area that complies with the last 3 month), which is useful for regarding special seasonal influences.

An additional benefit of using the moving average (instead of every single data point) as preprocessing step is elimination of missing values up to a certain degree and the decreased influence of outliers (i.e. wrong values, which are produced for instance

[2]dbcip.cs.uni-duesseldorf.de/∼tsp/tempus

by measuring errors or station breakdowns).

Using the Viterbi-Algorithm [For73] for finding hidden states of the HMM, which is established on base of the correlated time series, is equivalent to predict the development of one time series. A crucial step before doing this is the discretization of the values in the time series, because the values are the basis for the states of the HMM. As result of extensive testing, we decided to use 7 equiprobabilistic intervals (resulting in a HMM with 7 states), which provides a good balance between prediction results and performance. Each interval corresponds to a state of the HMM. Thus, no concrete values are predicted, but intervals; a typical prediction is for instance "at 11PM the river level will be between 9.5 and 9.7 meters".

TEMPUS is designed for the analysis and prediction of general time series. Although we demonstrate its functionality by the example of predicting water levels in rivers and implemented some application-specific features, it is applicable to time series of different origins. As already mentioned in the previous section, the prediction does not bases on geographical or other information, but only on the concrete time series. The only preconditions for applying TEMPUS to other time series is the appropriate formatting of the database entries (which of course could be easily adjusted) and coherences between time series. (In the next section we present a less application-specific approach, which also can be used for predicting time series. Like TEMPUS, this approach uses a HMM basing on two correlated time series, where the correlated time series pairs are discovered by using a motif-based time series representation.)

6.2.1 Evaluation

For evaluating TEMPUS, we conducted several experiments on a large dataset provided by the Federal Institute for Nature, Environment and Consumerism of North Rhine-Westphalia (Germany, www.lanuv.nrw.de). At the time of the evaluation, this dataset consisted of 50 million tuples and contained measurements of 180 station located in the rivers of North Rhine-Westphalia from January 1998 till April 2010. The experiments were carried out on an Intel Core 2 Duo 3.0 GHz desktop PC running Windows Vista with 3 GB of main memory using the Mozilla Firefox web browser.

In a first series of experiments, we tested the *result* of the *prediction* on basis of correlated time series. We calculated the DDTW distance between each pair of time series in our database in a preceding step. After that, we ordered the time series pairs according to their distances and analyzed the result of TEMPUS' prediction of the pairs with the smallest DDTW distances. The process of prediction alone took place in less than a second in all of the cases. An example for a correlated pair is the time series pair Düsseldorf/Kaub (cf. figure 6.4), which is intuitively correlated due to its

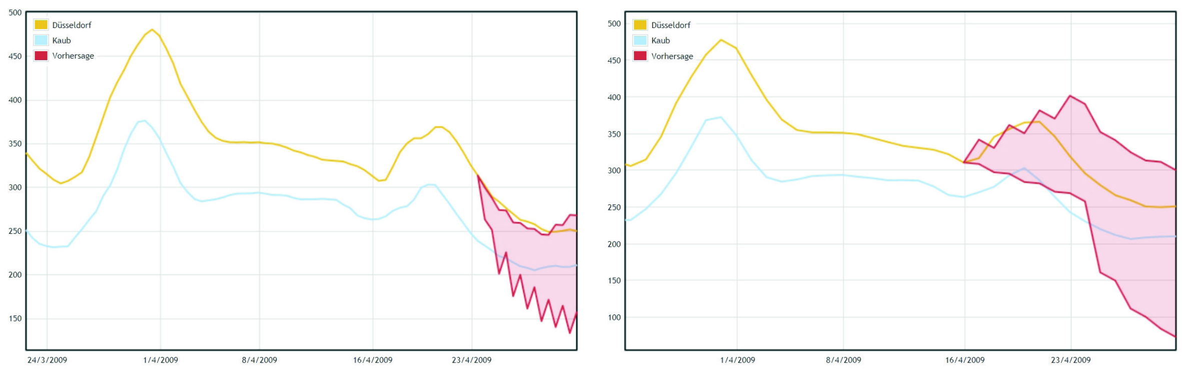


Figure 6.4: River level prediction at Düsseldorf (time series above) on base of the correlated time series derived from Kaub. The left plot displays TEMPUS' prediction result (the bounded area at the end of the upper time series) with the granularity of using 2 time series values per day, the right with 1 values.

geographical location (as already mentioned in example 4), what was stated by its small DDTW distance. The results of the correlated time series pairs show that apart from the period on which the HMM is constructed, the number of values regarded per day plays an important role for the prediction: up to a certain degree the accuracy of the prediction increased with an increasing number of values per day, but naturally the run time of the algorithm increases as well. Moreover, a too small or too large number of values per day did not lead to better prediction results, thus we implemented a middle ground of either 8, 4, 2 or 1 values per day as standard. This number of values regarded per day directly influences the period of time, which can be predicted, since we decided to implemented the prediction of a fixed number of 14 future values of a given time series. Predicting 14 future values with 8, 4, 2 or 1 values per day results in a prediction horizon between 42 hours and 14 days, which seemed to be reasonable for us, especially for the provided river level dataset (of course, this parameter could easily be adapted for other datasets). Examples of prediction results for 2 and 1 values per day (resulting in a prediction horizon of 7 and 14 days, respectively) are given in figure 6.4.

In a further series of experiments we tried to determine the relation between correlation of two time series (in terms of the DDTW distance of time series pair) and prediction accuracy. Predicting values up to two days in the future is reasonable for our application, which is affirmed by the fact that even the prediction of the Federal Institute for Nature, Environment and Consumerism of North Rhine-Westphalia[3] only reaches out two days in the future. Thus we took this point in time as reference and calculated the DDTW distance between 100 randomly chosen pair of time series

[3]The prediction of the federal institute bases on a complex physical river simulation model regarding river depth, underground particularities, drift, flow rate and several other geographic information, which thus can be regarded to be very accurate.

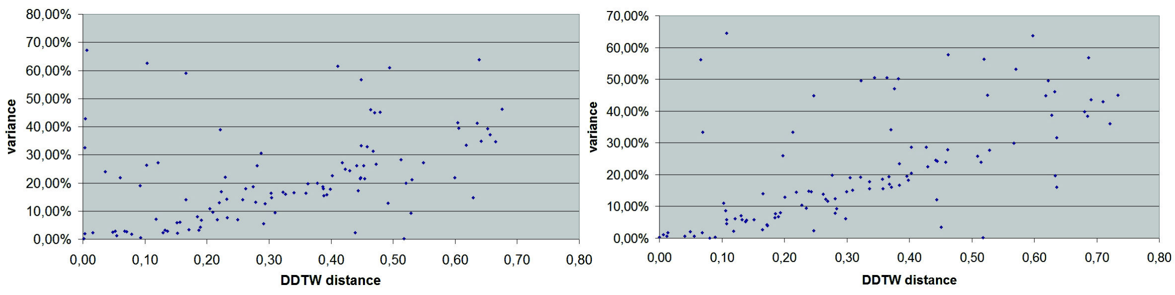


Figure 6.5: Graphical representation of the relation between time series coherence (in terms of DDTW distance) and variance from prediction result (48 hours in the future) and real values.

and the variance from the real value of the time series. As already mentioned in the previous section, TEMPUS does not calculate an exact value due to the discretization process, but an interval in which the values of the time series should be. For this evaluation we used the average of the minimal and the maximal predicted value as reference. The results are given in figure 6.5 which displays an expected trend: in general, the precision of a prediction increases with increasing correlation of time series. For correlated pairs of time series with a small DDTW distance we archive excellent prediction results, e.g. the variance of the predicted value 48 hours in the future of the former mentioned pair Kaub/Düsseldorf, which has a distance of 0.15, is 1.74%. There are even higher correlated pairs with similar good prediction results, and, of course, there are some outliers: time series, of which the distance is very small, but in spite of that the prediction does not yield good results. In at least some of these cases, this phaenomenon is caused by the fact that the DDTW distance merely regards the shape of two time series, and not their absolute differences. A very small river with very little water for instance cannot effect a large river very much, even if the shapes of their river level time series look quite similar.

6.2.2 Conclusion on TEMPUS and Future Work

This section presented TEMPUS, a prototype system for general time series analysis and prediction. TEMPUS uses a HMM, which is established on basis of two correlated time series, for analyzing and predicting the future development of time series. We demonstrated its functionality by the example of predicting water levels in rivers, which is an useful application for instance for floodwater prediction, boat transportation or canoeing. TEMPUS is applicable to many other applications based on time series of different origins, because the prediction is done only on base of the time series and without any additional domain knowledge. We also presented an evaluation of our approach, which showed that our system predicts the future development of river time series very well, if there are pairs of somehow correlated time series in the database.

Discovering correlated time series pairs is one of the first points which have to be examined in future work. In TEMPUS, the correlation is discovered by regarding the DDTW distance between two time series. However, as the evaluation showed, there are some time series pairs, for instance time series with a large absolute distance (derived from two rivers of very different sizes), which seem to be correlated due to their small DDTW distance, but where TEMPUS does not deliver a good prediction result. Thus, further means of discovering "somehow" correlated time series pairs have to be regarded and analyzed, e.g. as carried out in the next section by using a motifs-based representation, which represents certain time series characteristics.

Apart from that, there are several other interesting points left, which could be examined in more depth: at the moment the prediction is based on an adjustable time interval. In order to capture for example special seasonal characteristics (e.g. in the winter month January), further testing has to be done with several weighted time intervals as basis for the prediction (e.g. the last few month from the starting point and alleviated every January of the last few years). Although the detection of correlated time series has only to be done once, the expensive process of calculating the derivative dynamic time warping distance between all river pairs could be improved though indexing techniques.

Missing values and outliers are very problematic in the whole area of data mining, and they can be problematic for our system as well, if there are too much of them. Apart from predicting future values of time series, HMMs could also be use for predicting missing values in them, and by predicting a minimal and a maximal value that every single point in the dataset can adopt, for detecting outliers in the data (of course in combination with standard methods for solving these two problems, cf. e.g. [Hod11]).

Although we stated as advantage that TEMPUS is a general system, i.e. it is applicable to time series of several origins, the prediction of river levels could certainly be improved with domain-specific knowledge and information. An interesting extension for instance could be the consideration of rain amounts, which fall in certain areas around (and thus pour into) the rivers. This data is also available to the public on certain websites. In addition to that, geographic dependencies will surely enhance the prediction, provided automatically via GoogleMaps or semi-automatically via station meta information. Such meta information is typically provided with the river level measurements and contains for instance distances from river estuaries, coordinates of locations of stations et cetera and can be used to model river systems. HMMs sometimes tend to follow a trend in the data, for instance if values of a time series fall over a longer time, the HMM sometimes continues this trend by predicting falling future values, anyhow whether this is unrealistic. At least in the case of river levels one can state more or less a maximal and a minimal value for the range of the occurring values, which could also be considered for the prediction.

6.3 Motif-Based Time Series Analysis and Prediction

This section presents an approach for time series analysis and prediction [SC12b], which bases on a novel motif-based time series representation, inter-time-serial correlations and the Hidden Markov Model introduced in section 6.1. We explicate how time series from diverse origins can be transformed into this motif-based representation, which thus captures significant characteristics of the time series. The representation bases on semi-continuous sequential motifs [BST09] (cf. section 3.3); it is introduced in subsection 6.3.1 in detail, which also explains how it can be used for automatically detecting correlated time series in a given database. The basic idea is to cluster the motif-based time series representations hierarchically, which results in a dendogram of time series. Using this dendogram, a correlated time series can easily be found for every given time series. Figure 6.6 illustrates the whole procedure of detecting correlated time series, which will be explained in subsection 6.3.1 in more detail.

Analogously to TEMPUS, which has been presented in the previous section, our approach analyzes the time series database in order to predict time series on base of correlated time series. The main differences are the discovery of correlated time series and the fact, that the approach presented in this section is less application-specific. The prediction is done using the HMM introduced in section 6.1, which allows to predict the future development of one time series on base of known values from the one and another correlated time series. The functionality of our motif- and HMM-based time series analysis and prediction approach and the influence of the different parameters of the motif-based representation, the inter-time-serial correlation discovery and the prediction capability are evaluated on two large databases of river level measurements and stock data in subsection 6.3.2.

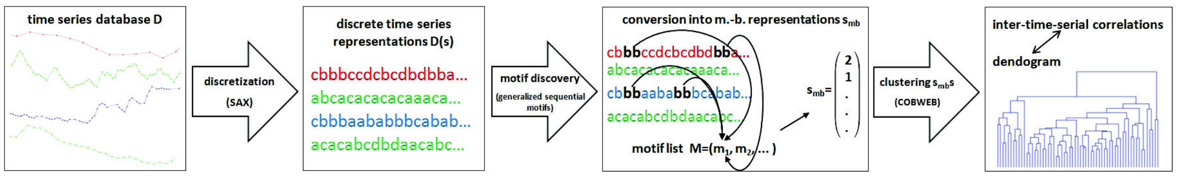

time series database D
discretization (SAX)
discrete time series representations D(s)
cbbbccdcbcdbdbba...
abcacacacacaaaca...
cbbbaababbbcabab...
acacabcdbdaacabc...
motif discovery (generalized sequential motifs)
conversion into m.-b. representations s_{mb}
cbbbccdcbcdbdbba...
abcacacacacaaaca...
cbbbaababbbcabab...
acacabcdbdaacabc...
motif list $M=(m_1, m_2, \dots)$
$s_{mb}=\begin{pmatrix}2\\1\\.\\.\\.\end{pmatrix}$
clustering s_{mb}s (COBWEB)
inter-time-serial correlations
dendogram

Figure 6.6: Illustration of inter-time-serial correlation discovery. The third rectangle shows the conversion of the first (or third) time series into its motif-based representation, assuming that the basic motif `bb` is the most significant motif in all time series discretizations, which occurs twice in this time series.

6.3.1 Discovering Inter-Time-Serial Correlations

The discovery of inter-time-serial correlations is an important part of our approach for time series analysis and prediction, since correlated time series form the basis for the prediction with the HMM presented in section 6.1. Thus, in this subsection we introduce a *m*otif-*b*ased time series representation (denoted as s_{mb}), which is inspired by the biological domain, where motifs are used for sequence classification [FA05, KSE$^+$05], and explain how it can be used to discover correlations between time series.

Since we are dealing with continuous time series, these time series have to be discretized first in order to discover motifs. From the vast number of discretization methods proposed in the literature (for an overview cf. for instance [DFT03, KK06b, LKWL07]), we chose to use SAX (Symbolic Aggregate approXimation) [LKLcC03] for discretizing, since it is successfully applied to several time series problems [Eam12] and has been shown to outperform other time series representations [DTS$^+$08]. SAX has been introduced in section 3.2 together with an appropriate distance function for symbolic time series, namely the MINDIST function, which is more meaningful as for instance the commonly used edit distance. In order to compare two time series, the MINDIST function aligns one symbol from the one time series representation linearly with one from the other, which is the analogue of the Euclidean distance for continuous values. Unfortunatelly, aligning time series linearly for comparison has certain drawbacks as we already pointed out in section 3.2, for instance when dealing with time series of different lengths, scalings, or translations. Thus, from our point of view the most suitable distance measure for comparing (parts of) discrete time series representations obtained by SAX, is the MINDIST function in combination with the non-linear alignments obtained by dynamic time warping. Whenever we compare discrete time series in the following, we assume that this combination is used as standard distance function.

The *motif-based time series representation* s_{mb} of a given time series s can intuitively be seen as d-dimensional vector, where each component contains the number of occurrences of certain previously discovered motifs. Several kinds of motifs have been proposed in the literature (cf. subsection 3.3), where from our point of view semi-continuous sequential motifs [BST09] are the best choice for capturing characteristics in time series, since they generalize continuous and non-continuous motifs. (A semi-continuous sequential motifs is a motif, in which maximal n gaps of a maximal length d are allowed. Gaps are introduced to compensate minor variations; a gap denotes an arbitrary string of a certain length, which may be contained in a semi-continuous sequential motifs.) Thus we decided to uses this kind of motifs for our motif-based time series representation.

In order to defined s_{mb} formally, let $D = (s_1, ..., s_{|D|})$ denote the database, which contains all continuous real valued time series s_i $(1 \leq i \leq |D|)$. Using SAX, every time series $s_i \in D$ is transformed into its discrete representation $D(s_i) = a_1 a_2 ... a_{j_i}$, with $a_k \in \Sigma$, where Σ is the alphabet used for discretizing all time series. Let $M = (m_1, ..., m_m)$ be the list of all semi-continuous sequential motifs discovered from all $D(s_i)$ as described in [BST09], which exceed a certain support threshold `minsup` (let m be the number of these discovered motifs), ordered by their statistical significance (i.e. approximated p-value [CA11] ascending[4]). Then the motif-based time series representation for a time series $s \in D$ is defined by

$$s_{\text{mb}} = (quantity(m_1, s), ..., quantity(m_d, s)),$$

where $quantity : M \times D \to \mathbb{N}$ is a function, which outputs the number of occurrences of a motif $m \in M$ in the discrete representation of a time series from D, and $d \in \mathbb{N}$ a constraint for the length of the motif-based representations (with $d <= m$). Depending on the parameters used for motif discovery a huge number of motifs might be discovered, but experiments showed (cf. section 6.3.2), that using all these motifs does not significantly improve the process of finding correlated time series, thus normally $d \ll m$ is chosen, which also leads to a complexity reduction. Note, that we are neglecting the significance threshold value α for discarding insignificant motifs by purpose, since for some datasets it could lead to the rejection of all discovered motifs [CA11], which would make a motif-based time series representation absurd.

After creating motif-based representations for all time series, these representations are clustered hierarchically [Joh67] in order to obtain a dendogram. The whole process of discovering inter-time-serial correlations can be stated by the following four steps:

1. Discretize all time series $s \in D$ (using SAX) in order to obtain $D(s)$,
2. extract all semi-continuous sequential motifs exceeding `minsup` from all $D(s)$, store them in list M ordered by their significance,
3. create the motif-based time series representation $s_{\text{mb}} \in \mathbb{N}^d$ for each series, and
4. cluster hierarchically all s_{mb} (e.g. by COBWEB [Fis87]).

The result of the fourth step of this process is a dendogram, which contains all time series grouped according to the similarity of their motif-based representation vectors. With this dendogram, the most correlated time series can easily be retrieved for a

[4]Note, that [CA11] regards the statistical significance for a "basic" motif (without gaps) in one time series, whereas our approach regards several time series and allows motifs with gaps. To do that, the p-values of a certain motif m (possibly containing gaps) are determined analogously to [CA11] for every time series s, which contains m in sufficient quantity (`minsup`), and the average is used as p-value for m for the whole database D.

certain time series, which allows time series prediction as explicated in the next section. The whole procedure of detecting correlated time series is illustrated in figure 6.6 above.

6.3.2 Evaluation

This subsection contains the evaluation of our approach and some implementational details. To start with the second, we implemented our approach in Java 6 using the Eclipse IDE on an Intel Core 2 Duo 3.0 GHz desktop PC running Windows 7 with 3 GB of main memory. The time series were stored in a MySQL 5 database running on a virtual machine of a XEN Dual QuadCore server (with 1.8 GHz, 2 GB main memory statically assigned to the VM). We used the SAX implementation from Pavel Senin [Pav12] for discretizing, WEKA's [HFH$^+$09] COBWEB implementation for clustering and libraries from the Colt project [Law12] for scientific computing.

In order to evaluate functionality and parameter influence, we conducted several experiments on two large databases, which contain river level measurements and stock data. At the time of the evaluation, the river level measurements consisted of approx. 81,500,000 tuples from 393 river stations over several years, which we obtained from the environmental agencies of the German federal states North Rhine-Westphalia (`http://www.lanuv.nrw.de`), Hesse (`www.hlug.de`) and Rhineland-Palatinate (`www.luwg.rlp.de`), and the stock data consisted of approx. 29,900 tuples of daily closing prices from 30 companies contained in the German stock market index DAX (as of 2011/06/30) from mid 2007-2011, obtained via Dirk Eddelbuettel's tool beancounter [Dir12] from the Yahoo finance server (`http://finance.yahoo.com`).

Inter-Time-Serial Correlation Discovery In a first series of experiments we tested what kind of correlations have been discovered by clustering our motif-based time series representations. The concrete question was whether two time series, which were output as correlated on base of our motif-based approach indeed were correlated. For answering this question we constrained us to the usage of time series with the duration of one year (in the same period of time) and picked out randomly 20 test time series from both databases (10 from each). Since it is difficult to quantify the coherence between two real-life time series in an exact value (in our case for the correlation between time series from different river levels or stocks), we examined the correlation for the 20 test time series in two experimental settings: a) we added the appropriate excerpts of the other real-life time series from both databases and 20 synthetic time series to our test databases. Each of the synthetic time series $s_i^{'}$ has been created from one of the test time series s_i by adding xca_i to every point of the time series, where x is a value randomly chosen from $[-1, 1]$, $c \geq 0$ a scaling factor and a_i the difference between the maximal and minimal value of time series s_i. Since each synthetic time series is constructed from one test time series, there should be a strong correlation between these time series pairs, as long as the scaling factor c is not chosen to large. For experiment setting b) we only added the appropriate excerpts of the remaining real-life time series to our test databases. For both settings and for each test time series we picked out the time series, which motif-based representation had the smallest distance to the test time series by using the previously obtained cluster dendogram (cf. section 6.3.1).

For these experiments, the SAX discretizations were obtained with word length parameters $j = 365$ (river level) or $j = 64$ (stock data time series), and alphabet size $|\Sigma| = 12$, which seemed appropriate to us for the domains, and we were searching for semi-continuous sequential motifs with at most 2 gap of maximal size 5 (river levels) or 2 (stock data), respectively. The minimal support threshold was set to `minsup` $= 0.05$.

The resulting test databases of experiment setting a) contained approx. 14,000,000 (river level) or 10,000 tuples (stock data), where approx. 320,000 or 650 motifs, respectively, had been discovered. For evaluating the correlation of the time series pairs, which had been discovered as correlated, we calculated the accuracy simply by dividing the number of correct pairs by the number of test time series. The left plot in figure 6.7 shows the results for the river level database with $d = 5000$ (where d is the length of the motif-based representations, see section 6.3.1). For small scaling factor ($c = 0$, $c = 0.02$, $c = 0.05$) our approach correctly identifies all correlated pairs, but of course, the larger c, the worse the accuracy: for $c = 0.01$ the first wrong time series pair is discovered and the accuracy decreases. The reason for this lies in the construction of the synthetic time series, since for the river level database with $c \geq 0.01$ the influence of the randomly added points is too big, the time series pairs which should be correlated are not high enough correlated anymore. The right plot in figure 6.7 shows the influence of d (the length of the motif-based representations) for the discovery of correlated pairs (for $c = 0.1$): if too less motifs are used for the motif-based representations, the accuracy is low, since important motifs are missing for the detection of correlated pairs. If d increases, the accuracy increases as well up to a certain degree, but then it increases only slightly, or in this case for $d \geq 5000$, not at all. The reason for this is

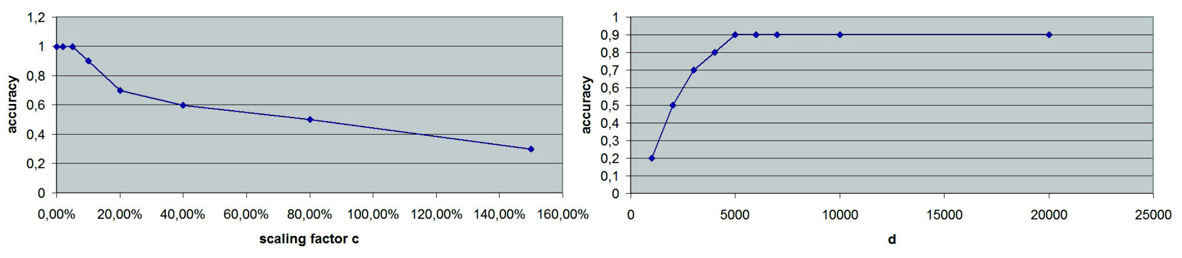


Figure 6.7: Inter-time-serial correlation discovery: results of experiment setting 1 b) (left plot $d = 5000$, right plot $c = 0.1$).

DDTW distance	DK-based correlation
0.092	very high
1.081	high
1.125	very high
1.3	medium
1.666	high

Table 6.1: Inter-time-serial correlation discovery: five real-life time series pairs discovered as correlated according to experiment setting 1 b).

the ranking of the motif according to their significance, which obviously is a successful approach.

For experiment setting b) we measured the "real" correlation of two real-life time series in the two terms of i) regarding the derivative dynamic time warping distance (cf. section 3.2 or [KP01]) between the time series, which have been discovered as correlated, and ii) judging their coherence on base of domain knowledge and time series charts. The domain knowledge-based judgment - classified into very high, high, medium, low or no correlation - is subjective up to a certain degree, but to the best of our knowledge, we used geographical facts for the river level database - e.g. if a river flows into another river, there should be a correlation between these two rivers due to the afflux, or the larger the distance between two river stations, the less their correlation - and commonly known microeconomic coherences between stock companies - e.g. that stocks from the same business area behave similarly in most of the cases - as well as the time series charts. Table 6.1 shows the first five results of this experiment ranked ascending by the derivative dynamic time warping distances.

Apart from the fourth time series pair in table 6.1, all given time series indeed are highly or even very highly correlated, which is shown by their low derivative dynamic time warping distance and our domain knowledge-based judgment. The first time series pair with the lowest derivative dynamic time warping distance for instance represents the time series derived at the two river level stations in Düsseldorf and Cologne, which are highly correlated due to their geographical location. The air-line distance between these two stations is about 30 kilometers, both stations are located at the river Rhine, which flows from Cologne to Düsseldorf about 56 kilometers. It is obvious, that the time series derived at these two stations have to be correlated somehow (e.g. if Cologne measures a very high water level, it will last about 10 hours until the body of water arrives in Düsseldorf and then a higher river level should be measured there too), which is affirmed by the time series charts (cf. figure 6.8) and their very low derivative dynamic time warping distance. On a scale from 0 to 4 (from not to very high correlated

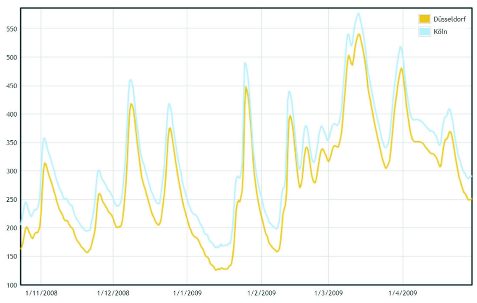


Figure 6.8: Chart of the two highly correlated time series derived at the river level stations in Düsseldorf and Cologne (Köln).

according to the former defined classification intervals), the overall correlation of the examined 20 time series pairs was 3.15, which is slightly better than high correlated (3).

Time Series Prediction After the first series of experiments, which has shown, that the time series which have been discovered as correlated by our approach indeed are correlated, we conducted a second series of experiment in order to evaluate the prediction capability using the HMM introduced in section 6.1. Therefore we randomly selected excerpts of one year of 100 time series from the river level and stock databases. For each of these test time series our system selected the time series from the rest of the databases with the highest correlation, and we deleted the last few values of the test time series in order to predict (and compare) them with the original values. The results of these experiments on the river level data is given in table 6.2, which shows the overall variance[5] of the prediction results dependent on the number of values which had been deleted and had to be predicted. A variance of up to 10% (predicting between 3 and 4 points) seems acceptable to us, especially in the context of time series derived from river level station. Since we used a word length $j = 365$ for discretizing, each point refers to one day, and prediction the future river level up to 3 or 4 day in the future with only a small variance is very good. As already mentioned in section 6.2, this is affirmed by the fact that the prediction by the environmental agencies, which are based on complex physical river models (regarding river depth, underground particularities, drift, flow rate and several other geographic information) only reach out for two days in the future before the results are not exact anymore. Thus, our approach also does a good job in predicting.

[5]Due to the discretization of the time series and the underlying HMM, our approach does not predict a concrete value, but a symbol from the discretization alphabet Σ, which represents an interval containing the predicted values. For determining the overall variance, we simply compare the mean values of the upper and lower boarders of the predicted intervals with the real values.

number of points to predict	overall variance
1	0.02
2	0.05
3	0.08
4	0.17
5	0.27

Table 6.2: Time series prediction: number of points to predict and overall variance of prediction result for 100 time series.

6.3.3 Conclusion and Future Work

In this section we presented an Hidden Markov Model based approach for time series prediction, which bases on inter-time-serial correlations. These correlations between time series are discovered by hierarchically clustering a motif-based time series representation, which consists of time series motifs ranked according to their statistical significance. We also presented an evaluation on two large datasets which shows the influence of different parameters and proves the functionality of our approach.

There are several directions for future work: firstly, motifs could not only be discovered by using the MINDIST function in combination with the non-linear alignments analog to dynamic time warping, but also by regarding more the shape and less the absolute distances between the symbols (similar to derivative dynamic time warping). Secondly, the motif-based time series representation could be altered in order to capture temporal progression in the data, e.g. by slicing time series using a sliding window mechanism with varying size. Thirdly, further prediction methods could be applied, especially for cases when the inter-time-serial correlation is not high enough for precise time series prediction via Hidden Markov Models. And finally, as already mentioned in the conclusions section 6.2.2 on TEMPUS, this Hidden Markov Model-based approach could also be used for outlier detection by predicting a maximal and minimal range for the values of the time series.

7

Applied Time Series Analysis

This chapter contains two interesting applications of time series analysis to linguistics and medicine, namely an approach for measuring text similarity between different languages using dynamic time warping, which can be used e.g. for detecting plagiarism, and an approach for automatic sleep stages scoring . Sleep stage scoring is an essential preprocessing step in the analysis of sleep; up to now it is mostly done manually by medical experts, thus an accurate automated version would be an enrichment to this domain.

Comparing texts for finding similarities is a significant challenge in several application areas, e.g. for detecting potential plagiarism among texts in the same language or even among texts in different languages (most of the approaches from linguistics are simply not applicable in the second situation). Thus, in section 7.1 we present TextWarper, a prototype system for measuring the similarity between large typed text, which applies temporal data mining methods (dynamic time warping). Artificial time series are derived from texts by counting the occurrences of relevant keywords using a sliding window mechanism, after which these time series are compared using the dynamic time warping distance. In preceding work [MSC08], the basic concept of our system has been presented, and it turned out that our idea might be especially suitable for text comparison across different languages. In the next section we present the foundations of our approach, detailed empirical results and a comparison with other established tools for text retrieval and analysis in order to show the validity of TextWarper, which produces reliable results for intra-language as well as for inter-language comparison [MSC09].

Sleep stage scoring is the medical term for classifying (usually 30 second lasting) epochs of sleep recordings into accordant predefined sleep stages. It is an essential preprocessing step in the analysis of sleep, which up to now is mostly done manually by medical experts.[1] In section 7.2 we present an approach for automatic sleep stage scoring and apnea-hypopnea detection [SKC10, SC10a, SC12a], which bases on data mining applied to time series from EEG, ECG, EOG and EMG. By several combined techniques (Fourier and wavelet transform, DDTW and waveform recognition), our approach extracts meaningful features (frequencies and special patterns like k-complexes and sleep spindles) from physiological recordings containing EEG, ECG, EOG and EMG data. Based on these pieces of information, a decision tree classifier is built, which classifies sleep epochs in their sleep stages according to the rules by Rechtschaffen and Kales and annotates occurrences of apnea-hypopnea (total or partial cessation of respiration). After that, case-based reasoning is applied in order to improve quality. We tested and evaluated this approach on several large public databases from PhysioBank, which showed very good overall accuracies of 95.2% for sleep stage scoring and 94.5% for classifying minutes as apneic or non-apneic.

[1]In some countries medical experts are even obligated by law to spend a certain amount of time for the analysis of these usually 8 hours lasting recording (e.g. in Germany), in spite of the fact that experienced personal would need less time than specified.

7.1 Measuring Text Similarity With Dynamic Time Warping

The ever-growing content of the World Wide Web together with the development of general and specialized search engines enables us to easily search for relevant documents for almost any topic. In addition, the amount of textual data stored in corporate or government databases continues growing at remarkable speed. Thus, the comparison and retrieval of texts is a problem of great importance today.

A main application of text comparison is finding texts which are similar to one the user has already labeled as relevant, i.e. the user wants to find more documents covering the same topic as a given document. This query by example approach is not mandatory since it is also valid (and common practice) to provide only a few keywords which are then used to search a set of documents. Another field of interest is the detection of plagiarism, which is an eminent problem, e.g. in journalism and academics[2]. It is vital to decide whether a given text is original work or was published before with high confidence, since unoriginality is not only a matter of bad style. In most scenarios it implies severe consequences, e.g. the failure of a class or even the revocation of an academic degree. In the course of linguistic analysis it might also be interesting to identify documents with similar characteristics, e.g. regarding the occurrences of terms within them.

This section presents TextWarper [MSC09], which is an approach for comparing texts by representing them as artificial time series in conjunction with an appropriate distance measure, namely the dynamic time warping (DTW) distance (cf. section 3.2). Artificial time series were first proposed in [RK04]; in our first work on this topic [MSC08] we showed that our idea might not only be valid for the detection of similar documents, but especially for solving the problem of cross-lingual analysis. The basic idea for obtaining the time series is to define one or more keywords and count how often these terms occur within a predefined sliding window, which is applied to the document. The derived characteristic term distributions might hint towards documents, which are similar and therefore worth a closer look.

To allow for a fair judgment of our system, we selected two established frameworks for measuring text similarity (PRAISE [LC01] and Lucene[3]) in order to compete against them in an exhaustive empirical evaluation. To make sure the amount of data is sufficient for a meaningful conclusion, we used the abundant textual data gathered in the Europarl project [Koe05] as foundation for our analysis.

The rest of this section is organized as follows. Subsection 7.1.1 reviews some related work, followed by subsection 7.1.2, which presents TextWarper, our approach for

[2]A nice collaborative website covering this topic due to recent events is `de.guttenplag.wikia.com`.
[3]Apache, `http://lucene.apache.org`

measuring text similarity. Subsection 7.1.3 briefly describes the systems for validating our prototype system (PRAISE, Lucene and the Europarl text corpus). The results of our extensive experiments are stated and discussed in subsection 7.1.4. Finally, subsection 7.1.5 concludes this section about TextWarper and states some directions for future work.

7.1.1 Related Work

In the area of information retrieval the *vector space model* [SWY75] has emerged and proven to be quite successful for identifying documents with similar content. The representation of texts as vectors of term frequencies allows to easily compare texts by computing similarity values from vectors. Typically, the terms considered are the distinct words occurring in the text after eliminating stop words and after applying stemming to reduce the terms to a base form. In the vector the number of occurrences are counted for the remaining terms. A basic similarity measure used for comparing the vectors is for instance the cosine measure [BYRN99]. One of the systems (Lucene, described in subsection 7.1.3 in more detail) which we will use later for evaluating our approach, is based on this general approach.

Other approaches uses graph representations of texts in which nodes represent the terms and edges represent associations between the terms, as e.g. in [TNI04]. These techniques are rather successful in identifying texts having the same (or similar) topic. For more specific applications like plagiarism detection their application is limited, for instance comparing term vectors is not able to take the distribution of important terms within the documents into account. Obviously, other properties beside the frequency of terms have to be considered as well.

Several other approaches try to overcome this problem for instance by considering the edit distance between text parts [ZFMP06], by computing the occurrences of letter and word combinations [LC01] and by syntactically analyzing individual sentences [LC07, WJ04]. These "low-level" (as opposed to "content-based") approaches allow for finding sentences or paragraphs which have been copied verbatim or only with slight changes. In further approaches a combination with content-based comparison was proposed to improve the retrieval performance [IS05]. One of such systems is for instance PRAISE [LC01], which we will also consider in subsection 7.1.3 in more detail. Even these hybrid methods usually fail if enough variations are introduced e.g. to cover plagiarism. In particular, they can usually not detect the similarity of translated texts due to the fact that syntax as well as other "technical" properties of the texts can be quite varying in different languages.

These problems can be resolved to some extent by introducing artificial time series as representation for texts. The idea of representing non-temporal data as time series has already successfully be implemented in other application fields, e.g. for shape recognition in images [XKWM07] and the analysis of handwritings [RM03].

7.1.2 TextWarper

The analysis of artificial time series derived from texts is the cornerstone of TextWarper. This subsection explained in detail, how these time series are defined and calculated.

As first step, documents have to be imported (TextWarper currently supports plain text, PDF or DOC), where optionally stemming and stopword reduction can be applied. These two techniques are established in the field of text comparison and often improve the quality of the results (cf. [BYRN99]). After the text import the unique terms are identified and it is measured how relevant each of them is, i.e. how often each term occurs. This relevance value will be helpful to choose the appropriate keywords in the second step.

For the calculation of the actual time series, one or more of these keywords (i.e. the words that will be counted for each window during time series computation) need to be selected. Semantically important terms with high frequency are a good choice here since they capture the characteristics of a given text best. The second parameter is the width of the window which will be moved along the text, i.e. the amount of words considered at each step. The natural maximum for this value is the word count of the whole text. The third (and last) parameter determines how far the window is moved forward in each computation step. The value for this step size is limited by the window size, since for a larger value not every word in the text would be considered. An illustration of all parameters is given in figure 7.1 where the whole time series creation process is explained with a small example. As figure 7.1 also shows, an intermediate step in calculating the time series in our implementation is creating a binary representation of the text, where the keywords selected by the user are shown as 1 and the other words as 0. This simplifies the calculation of the individual points constituting a time series as this can be achieved with basic arithmetics.

The length of the time series, and thereby the precision and computation time as well, directly depends on the choice of parameters. Given the length N of a text (here, the length is interpreted as number of words), the window size w, and the step size s for moving the window, the length l of the resulting time series can be determined as

$$l = \left\lceil \frac{N - w}{s} \right\rceil + 1.$$

In this formula $N - w$ is the number of words remaining after the first window has been calculated (i.e. the first point of the artificial time series), which is divided by the step size s of the moving window. Finally, 1 is added for correcting rounding down.

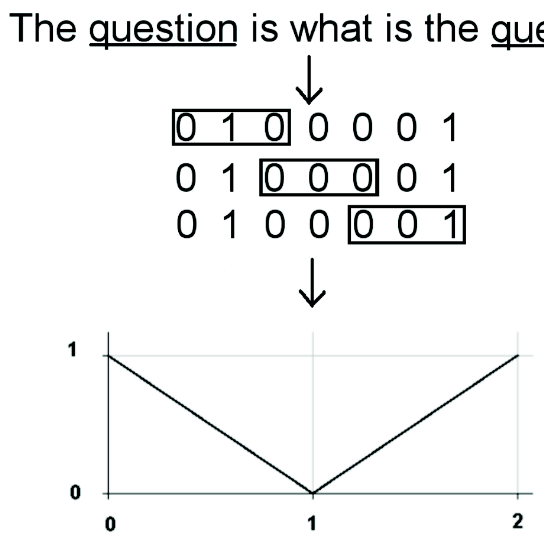


Figure 7.1: Illustration of the time series creation process. The keyword of interest is "question", hence occurrences of this term are shown as 1 in the binary representation and all other terms as 0. The window size is 3, the step size is 2, and these settings yield a time series of length 3 with the values (1, 0, 1) derived by counting the set bits in each window.

Note, that the last window will be incomplete (with less than w words), if $N - w$ is not divisible by s.

There are several reasons for carefully choosing the parameters w and s for text comparisons. On the one hand side the resulting time series should be detailed enough to provide reasonably interpretable distance values in text comparisons. On the other hand, the longer the time series is, the more computation time for comparison (i.e. for computing distances) is needed. Furthermore, time series with too fine-grained details are prone to small changes in the text producing surprisingly large distances. In subsection 7.1.4 we therefore investigate this aspect in more detail.

After the time series have been calculated the final (and crucial) step is to compare them, i.e. calculate the distances between them. In our approach, the user has several options considering which time series should be compared, ranging from the comparison of two single time series to the comparison of all time series of all texts in the database. However, we assume that the typical scenario is that a query text (or paragraph) is compared to a set of documents to check whether a similar document can be obtained.

In the actual comparison process, the distance measure of our choice is the dynamic time warping distance, which has been used in many contexts ([RM03], [SC90], [KP00]). As already mentioned in section 3.2, the basic idea is to find a non-linear matching of the points of two time series which is intuitively equivalent to stretching or compressing the time series in the x-axis. If text is translated into another language or slightly altered in order to disguise plagiarism, minor variations and shifts are to be expected. The overall word count and the exact counts and positions of the keywords may change, but the general distribution of the keywords should remain similar. In this scenario the DTW distance is appropriate since it allows comparison with a certain degree of tolerance regarding these alterations.

7.1.3 Systems for Validation

This subsection gives a short overview of the systems used as benchmark for the evaluation of TextWarper, as well as a description of the database used as foundation for our testing scenario.

The main motivation for choosing these systems was that we wanted to cover the two cases of content-based analysis (Lucene) and low-level analysis (PRAISE) with freely available software, thus commercial systems as Turnitin[4] were not considered because of their black box nature. We also tested other free systems such as Zettair[5], but the two systems that were finally chosen represent the basic approaches mentioned before and in subsection 7.1.1.

Lucene Apache *Lucene*[6] is a Java-based, open source text search engine library. It allows to efficiently index and search huge collections of textual documents based on the vector space model, and many contributions have been made which provide a great variety of functionality. One aspect of special interest for our testing scenario is the "MoreLikeThis" class, which allows to calculate a term vector from a given example document. This vector can then be used to obtain similar documents from the database, which is exactly what was needed for our testing scenario. The default distance measure used by Lucene correlates to the cosine-distance between a document and a query vector.

PRAISE *PRAISE* (Plotted Ring of Analyzed Information for Similarity Exploration) [LC01] is a tool, which allows to perform a low-level similarity analysis on a group of documents that is based on the words pair simple metric (by default). This similarity measure basically relies on the detection of pairs of words which appear in a set of documents, and based on this a similarity value for each pair of documents can be obtained. Other options include the clustering of documents and the analysis of document pair similarities with the closely related Visualization and Analysis of Similarity Tool (VAST), however, these options did not apply to our testing scenario and were therefore disregarded. Note, that it is also possible to select other similarity measures (e.g. based on characters or larger groups of words) in the implementation of

[4]`http://www.turnitin.com`
[5]`http://www.seg.rmit.edu.au/zettair`
[6]`http://lucene.apache.org`

PRAISE we used for our experiments, but we sticked to the default settings since the authors report reasonable results for the words pair metric.

Europarl *Europarl* [Koe05] is a corpus of texts from the proceedings of the European Parliament. In the version used for our experiments, it includes the manual transcripts of discussions from the years 1996 to 2006 in 11 languages with a single document for each day of discussion. This adds up to about 6,000 documents with approximately 200,000,000 words in total. The interesting aspect of this collection, which makes it especially suitable for our purposes, is that the content of a document for any given day is the same in each language. Although the original idea of Europarl was to support research in the field of statistical machine translation, the parallel texts can also be considered as cases of plagiarism between different languages. This type of man-made translations is exactly what we want to cover in our experiments.

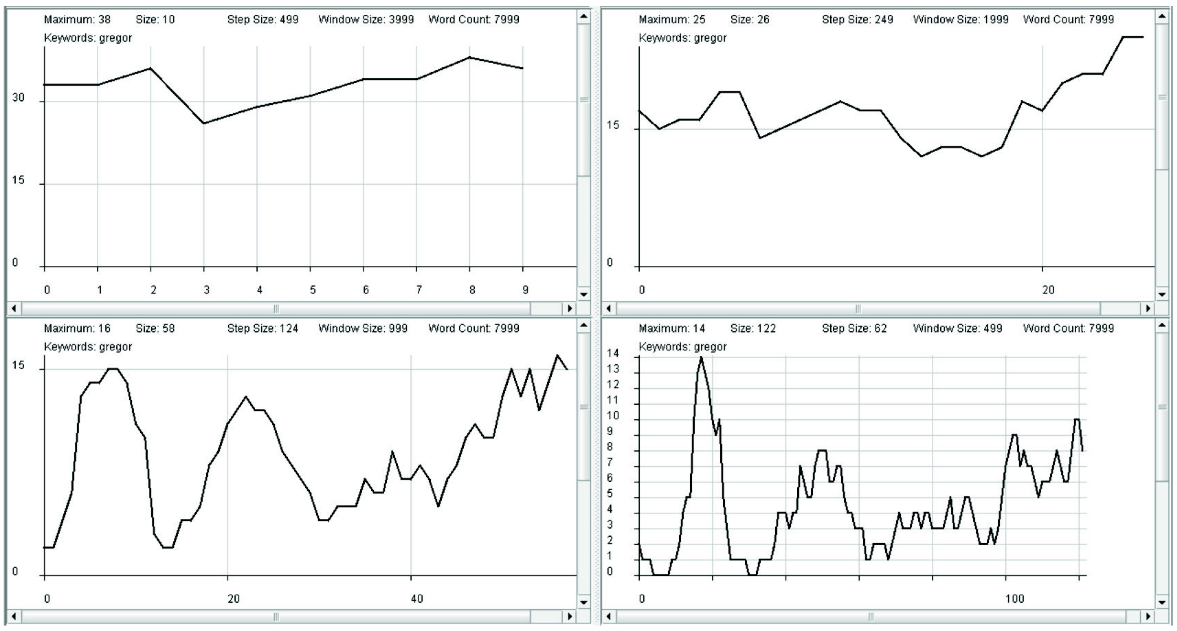
7.1.4 Evaluation

After briefly describing TextWarper and the other tools used for our experiments, this section presents our empirical results which point out the validity of our approach in practical applications.

Parameters For the calculation of time series from given texts, we first have to select appropriate parameters. In [MSC08] we already presented first results about how to choose the parameters w (size of the sliding window) and s (step size) for text comparison. In the following, we state the detailed results obtained from a number of experiments.

Obviously, the *window size* w directly influences the coarseness of the time series. In figure 7.2 we depicted the time series for different window sizes (where the step size s is chosen in relation to w). Small window sizes produce a lot of details, while large window sizes result in smoother curves. With a small w the maximum frequency of the considered keyword can only reach small values. In contrast a large w results in a curve with few changes. So, for text comparison neither a small nor a large w is reasonable. Together with the experience from further experiments with different texts, choosing a value around $\frac{N}{4}$ (with N the number of words in the text) as w turned out as a good choice.

Concerning the *step size* s, i.e. the amount of words the window is moved forward in each calculation step, we can easily see that it directly influences the length l and "grade of detail" of the time series. In turn it also determines the computation time for comparison as the DTW algorithm in its basic implementation runs in $O(l^2)$. In figure 7.3 several time series for different step sizes (here assuming a fixed window



Maximum: 38 Size: 10 Step Size: 499 Window Size: 3999 Word Count: 7999
Keywords: gregor

Maximum: 25 Size: 26 Step Size: 249 Window Size: 1999 Word Count: 7999
Keywords: gregor

Maximum: 16 Size: 58 Step Size: 124 Window Size: 999 Word Count: 7999
Keywords: gregor

Maximum: 14 Size: 122 Step Size: 62 Window Size: 499 Word Count: 7999
Keywords: gregor

Figure 7.2: Influence of the window size w on the observed curves. From top left to bottom right, w was set to $\frac{N}{2}$, $\frac{N}{4}$, $\frac{N}{8}$ and $\frac{N}{16}$ where N is the text length. For large values, the observed shape is rather coarse, and it becomes more and more detailed as w gets smaller.

size) are depicted. The time series with a smaller s show more local variations which produces larger distances when comparing time series of two texts although only small textual differences (maybe even only due to different languages) might be the reason. Setting s to $\frac{1}{4}$ of the window size w turns out to be an appropriate choice for our purposes because this leads to a time series representation which reliably reflects the characteristic distribution of keywords within a text.

Basic Testing Scenarios To test the usefulness of our idea in the detection of similar documents, we basically wanted to cover two cases: Comparison between documents in the same language (which is the focus of the existing tools for text analysis) and comparison between different languages. The latter case might be of special interest in the academic environment, where students might be tempted to simply translate research papers or other sources into their native language to turn them in as their own original work.

In order to take care of these cases, we randomly selected 100 English discussion transcripts from the Europarl corpus. Out of these transcripts, we randomly selected 20 documents as our test subjects. To emulate plagiarism within the same language, the German versions of these transcripts were translated into English via Google Translate[7]; this resulted in documents which are quite similar to the original ones, but with a

[7]`http://translate.google.com`

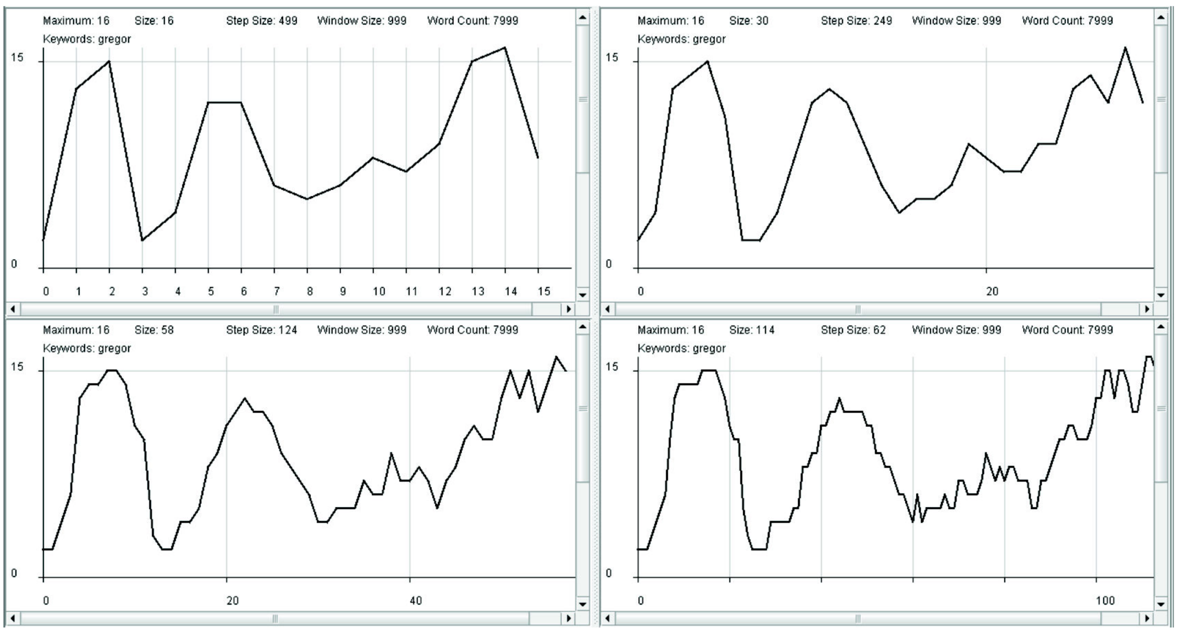


Figure 7.3: Influence of the step size s on the observed curves. From top left to bottom right, s was set to $\frac{w}{2}$, $\frac{w}{4}$, $\frac{w}{8}$ and $\frac{w}{16}$ where w is the window size. The overall shape stays the same, but local variations become more and more visible as the step size gets smaller. Note that the time series length roughly doubles when the step size is halved.

certain degree of variation which resembles manual alterations to disguise unoriginality. These documents were added to the original corpus, and these "target documents" were to be discovered by the given text comparison software. To emulate cross-lingual unoriginality, nothing more had to be done. The target documents in this case were the German versions of the selected test subjects, being a subset of the 100 German transcripts corresponding to the originally selected English transcripts. To show that our approach is not limited a specific language pair like English-German but really "language-independent", we briefly present the results of some further experiments with English, Spanish and French language later in this subsection.

For the following experiments, we selected "commission" and its German equivalent "Kommission" as keywords for time series calculation since these terms appear in every transcript with considerably high relevance. We restrict our considerations to a single keyword although TextWarper allows to consider multiple time series for a whole set of keywords, thus produces multivariate time series. At the end of this subsection we discuss the possible benefit and the partly surprising problems with such multivariate time series for text comparison.

For Lucene, we simply calculated the term vectors for each test subject using the "MoreLikeThis" class. Those were fed to the database to look for suspicious documents using the default similarity measure mentioned earlier.

For PRAISE, we performed the default pairwise analysis on both the English and the German sets of documents, each including the described "target documents".

Retrieval Results English-German The retrieval positions for the target documents in English to English retrieval and English to German retrieval are given in tables 7.1 and 7.2, respectively, as well as in figures 7.4 and 7.5. Note, that the exact values for similarity between the documents have mostly been omitted since the methods for similarity calculation are completely different for each system. However, we will discuss some observations regarding these values at the end of this subsection.

text / tool	1	2	3	4	5	6	7	8	9	10
TextWarper	1	2	1	1	1	1	6	2	1	6
Lucene	1	1	27	1	1	3	8	1	1	1
PRAISE	1	3	2	12	39	56	44	50	24	51

text / tool	11	12	13	14	15	16	17	18	19	20
TextWarper	1	1	1	3	4	7	1	1	1	3
Lucene	1	1	2	17	1	1	1	1	2	2
PRAISE	49	49	40	44	48	40	1	1	22	34

Table 7.1: Retrieval positions for English/English comparison.

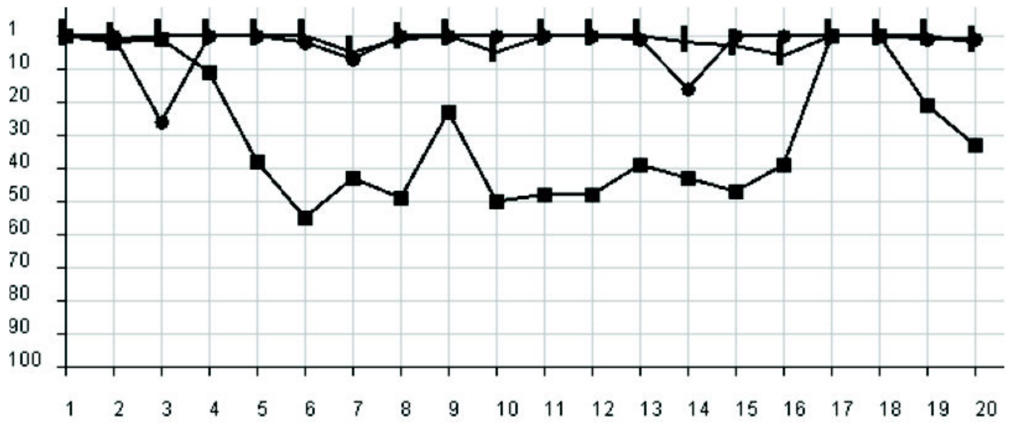


Figure 7.4: Retrieval positions for English/English comparison. The results for TextWarper, Lucene and PRAISE are marked with lines, dots and squares, respectively.

First of all it can be concluded that Lucene, and to some extent also PRAISE, are suited for the purpose of plagiarism detection within the same language. While the low-level approach of PRAISE fails at a considerable amount of examples (which may probably be battled by a variation of the standard parameters), Lucene shows excellent results, which does not come as a surprise as it is backed by years of research in the field of information retrieval. Nevertheless, the results of TextWarper indicate that our approach might be on par with the existing solutions in 12 out of 20 cases the suspicious document was returned as 1st hit, and in no case the retrieval position was worse than 7th in a database of 100 documents, which still allows for manual validation of the results.

text / tool	1	2	3	4	5	6	7	8	9	10
TextWarper	1	1	1	1	1	1	1	1	1	1
Lucene	22	1	7	1	1	1	13	1	3	1
PRAISE	3	18	30	8	18	3	14	11	12	14

text / tool	11	12	13	14	15	16	17	18	19	20
TextWarper	1	1	1	1	1	1	1	1	1	1
Lucene	1	1	1	2	1	28	2	1	1	1
PRAISE	17	3	5	21	19	6	11	11	3	3

Table 7.2: Retrieval positions for English/German comparison.

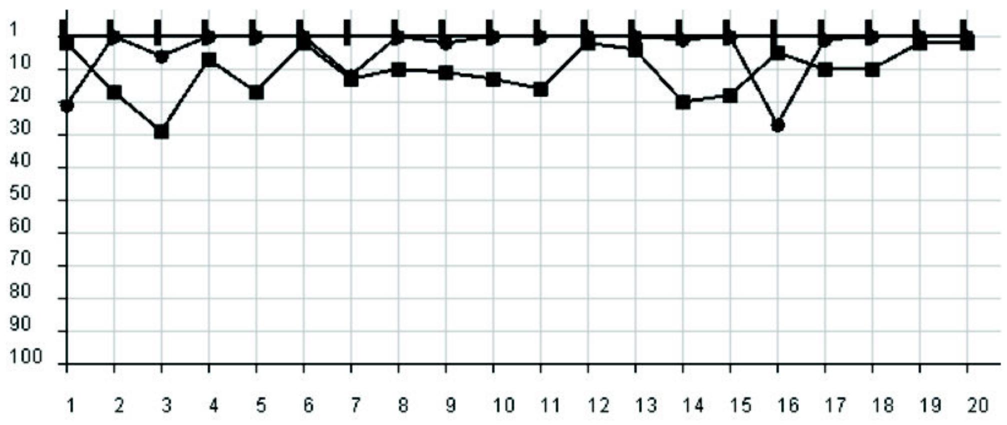


Figure 7.5: Retrieval positions for English/German comparison. The results for TextWarper, Lucene and PRAISE are marked with lines, dots and squares, respectively.

For the detection of similarities between German and English documents, Lucene and PRAISE show quite unexpected behavior. At a first glance, the results for PRAISE are disheartening since the suspicious documents are never returned as 1st hit, but in general the results are reasonable with hits mostly in the top 20. As mentioned before, better results might be achieved with different parameters. The results for Lucene were surprising in the sense that the change of language does not seem to make much of a difference; the results are still good, with only few exceptions. This might be due to the techniques of stemming and stopword reduction, which might make up for some of the differences between German and English.

The key result, however, is that in this scenario TextWarper outperformed both competitors by a large margin. In every single case the target document was returned as 1st hit, which clearly shows that our approach might be well worth for consideration for interlingual text retrieval. Moreover, we firmly believe that the results might get even better (for intra- as well as interlingual comparison) if more effort is put into the preparation of the retrieval process. For our test runs, only one keyword appearing in all documents with high relevance was chosen for each time series calculation, to achieve a single step indexation of the whole set of documents. Nevertheless, if the most relevant keyword would have been selected for each document (i.e. each retrieval run), the performance might almost certainly have been even more impressive.

On a side note, another interesting observation during our experiments was the change in absolute similarity values when the scenario was changed from English/English to English/German comparison. For TextWarper, the DTW distances between English texts and their altered versions were very close to those between the texts and their German versions. In fact, it would not be possible to tell from the values alone if they were derived from inter- or intralingual retrieval; in this light, our system can truly be called language-independent since retrieval from a mulilingual database is also feasible, cf. subsection 7.1.4. This is not true for the other two systems, where the similarity value for two English documents is usually more than an order of magnitude greater than that for an English document and its German equivalent. While this seemingly does not have too much impact on the retrieval from unilingual databases (as in our testing scenario), this might be a problem when documents from different languages are mixed in a database; in this case the target documents written in a language different from the sample document might not be discovered as the similarity for unrelated documents in the same language might still be higher.

Multi-lingual Experimental Results In order to show that TextWarper is not limited to some special language-pair like English-German we summarize in this subsection the results of our experiments with two more widespread languages, namely Spanish and French. We selected again "commission" as single English keyword, and "comisión" and "commission" in Spanish and French, respectively.

English text number	1	2	3	4	5	6	7	8	9	10
pos. in result list in										
Spanish	4	1	10	7	14	10	5	10	2	19
French	12	2	12	5	7	16	13	8	1	12

Spanish text number	1	2	3	4	5	6	7	8	9	10
pos. in result list in										
French	1	1	1	1	1	1	1	1	1	1
English	5	4	12	6	8	8	13	5	3	11

French text number	1	2	3	4	5	6	7	8	9	10
pos. in result list in										
Spanish	1	1	1	1	1	1	1	1	1	1
English	6	10	12	5	7	17	9	7	4	6

Table 7.3: Retrieval positions for English/Spanish/French comparison.

We conducted three different experiments. In the first experiment our database consisted of 100 Spanish documents and 20 appropriate French documents, and the database of our second experiment consisted of 100 French and 20 appropriate English documents. In both experiments each of the 20 documents was compared to the rest of the database, and in both case the proper document in the other languages was the first correct result.

In our third experiment we took 100 English documents and added the French and Spanish version of the 50 first documents to the database. We randomly selected 10 out of the 50 first documents of every language and compared those 30 document with the whole database. Of course, the first result was in every case the document itself. Due to this trivial case we subtracted 1 from all retrieval positions of figure 7.3. The second hit for every French document was the appropriate Spanish document and the other way round too, and than followed by the correct English one after a span of Spanish/French documents, see figure 7.3 for details. This not surprising result can be explained by the fact that both French and Spanish are Romanic languages and thus more related than Spanish and English for example. English and German are as Germanic languages related as well, and they show the same behavior as indicated by lower distance values in the comparison process described at the beginning of this subsection and confirmed by further experiments.

Performance Another interesting topic, especially when dealing with a large corpus of text, is how “usable” a retrieval approach is considering the consumption of time and memory. Considering this aspect we present some observations from our experiments, which were performed on standard desktop PC (Intel Pentium Dual Core 1.6 GHz running Windows XP with 2 GB of RAM).

First of all, indexation as well as retrieval are conveniently fast when using Lucene. Indexing hundreds of documents only took a few minutes, and retrieval results were returned almost instantly. Part of the reason for this is probably the long time of optimization Lucene has gone through, with contributions from many developers.

On the other hand, PRAISE showed a disappointing performance in the sense that it became almost unbearably slow for the amount of texts involved in our testing scenario. For even larger collections of documents, PRAISE did stop working at all as it ran out of memory. These technical issues may probably be resolved by changes in the implementation and are most likely not caused by the theoretical principles PRAISE is based on. However, in its current form PRAISE does not seem suitable for the analysis of large text corpora.

For TextWarper, it can be pointed out that for the first step of importing and indexing all texts (i.e. calculating the representative time series) a considerable amount of time is needed as it involves quite a few calculation steps. On our test system, 1 to 2 minutes per run, i.e. importing texts, calculating time series and comparing one certain time series to all others, was the average time observed. Due to the fact that up to now we have not make any effort concerning performance optimization we expect a significant improvement of our system soon. Our memory interface was already designed for dealing with huge collections of texts because only the data considered at a given moment is held in primary storage. As for the retrieval of texts, the speed is almost comparable to that of Lucene. This might not seem obvious as the DTW distance with its polynomial complexity is used for similarity calculation, however, as stated earlier reasonable retrieval results can be achieved with fairly short time series representations. Therefore, the inherent complexity of DTW only has a marginal effect on the overall performance. Nevertheless, optimizations are also possible here e.g. by using FastDTW [SC07] or LB_Keogh [Keo02].

All in all, we figured out that our system is already well suited for the retrieval of text in a large database, and with future enhancements to come the performance will hopefully allow to analyze even more data with reasonable runtime behavior.

Multivariate Time Series As mentioned in subsection 7.1.2, our system is also able to calculate multivariate time series, i.e. time series which are generated from multiple keywords. However, we refrained from using this option, and in this subsection we will give reasons for this.

text—w/s	16/2	16/4	16/8	16/16
altered text	0.75	0.59	0.56	0.44
German translation	0.79	0.61	0.55	0.42
unrelated text 1	3.29	2.94	2.86	2.7
unrelated text 2	3.12	3.13	2.85	2.83
unrelated text 3	2.99	2.8	2.72	2.54
unrelated text 4	3.53	4.35	5.11	5.49

text—w/s (MV)	16/2	16/4	16/8	16/16
altered text	1.368	1.091	0.922	0.821
German translation	0.923	0.7	0.571	0.484
unrelated text 1	2.802	2.744	2.645	2.567
unrelated text 2	2.723	2.676	2.436	2.25
unrelated text 3	2.83	2.568	2.419	2.196
unrelated text 4	3.304	3.363	3.555	3.582

Table 7.4: Distances calculated for the a sample text with varying values for window size w and step size s. For example, "16/4" means $w = \frac{N}{16}, s = \frac{w}{4}$ for a text length of N. MV denotes the results for retrieval with multivariate time series derived from the three most relevant keywords

As of now, for the comparison of multivariate time series we simply use the arithmetic mean of the distances calculated for each dimension, i.e. for each univariate (or "simple") time series. Of course, this is a rather naive approach which does not fully reflect the characteristics of the time series. As the experiments with multiple keywords show, this leads to worse results than using only a single keyword. See table 7.4 for an example.

More specifically speaking, the distances between dissimilar texts get smaller when using multiple keywords while the distances between similar texts get slightly larger. When analyzing the data in more detail, it seems as if great distances in the first dimension (i.e. the most relevant keyword) are to some extent "canceled out" by smaller distances in the other dimensions. This is explainable since the keywords with lower relevance are observed less often and thus the mean of the calculated time series is smaller; the consequence are smaller distances.

As a conclusion from the current version of TextWarper, the use of multiple keywords for comparison makes it harder to detect similarities, especially since the computational effort grows with each added time series. Therefore, the focus was on univariate time series derived from highly relevant keywords. Ideas for possible solutions include the adjustment of the parameters for window and step size to handle the less relevant keywords and the attachment of different weights to the dimensions with respect to the keyword relevance. However, more research is required to determine useful values which allow to gain informational benefit from the use of multivariate time series.

7.1.5 Conclusions and Future Work

In this section we presented TextWarper, a prototype system for comparing texts using dynamic time warping. The problem of measuring text similarity was reformulated as problem of measuring time series similarity (as originally proposed in [RK04]). We presented an thorough empirical evaluation of TextWarper, of which the results showed that our system is a serious competitor for existing text comparison software, being superior in an inter-lingual testing scenario. Nevertheless, there are several enhancements and extensions for future work.

First of all, an obvious extension is the evaluation of text comparison for other languages than English and German, especially since the technique proved to be a potential aid for language-independent comparison. The Europarl database is an excellent starting point to explore the validity of our idea for other (possibly more uncommon) language pairs.

Another interesting aspect is the automatic suggestion of appropriate keywords for time series calculation, since this crucial step completely relies on the user right now. This might be a simple task for texts in the same language (as the same keyword should most likely be used for all texts involved), but it might be less obvious when translations are considered; the incorporation of dictionaries for the different languages involved could effectively solve this problem. A more elaborate system could also make recommendations for the other parameters (window and step size) based on the text length and the keyword relevance.

As mentioned at the end of subsection 7.1.4, the usage of multivariate time series (i.e. the composition of time series from multiple keywords) is already possible in our system, but in our evaluation we could not achieve better results than using univariate time series. Thus, further research and testing is necessary in order to determine useful settings.

In addition to that, the realization of partial matchings would be a profound addition. By now, it is only possible to compare documents or paragraphs as whole, but of course it is also possible that a copied passage is seamlessly embedded into another document. This plagiarism attempt would probably go undetected when using only the

DTW distance, thus the notion of similarity needs to be adapted. Slicing a document into “artificial” paragraphs and comparing them with each other [ALSS95], and then calculating the amount of warping observed between different regions of the texts to identify similar passages could be an solution for this.

raised is Finally, it would be very interesting if and how the presented approach can be combined with established techniques such as the vector space model in order to improve the overall performance in practical applications.

7.2 Automatic Sleep Stage Scoring

Sleep is a vital part of all our lives. The result of sleep disturbances like sleep apnea can reach from light day drowsiness to really severe characteristics, e.g. to microsleep during driving, which causes a huge amount of deadly accidents every year [BTPR08]. Due to its fundamental nature, the analysis of sleep has attracted much attention during the last decades. There are several challenging problems, which have not yet been solved adequately and efficiently, two amongst others are sleep stage scoring and apnea-hypopnea detection. Sleep stage scoring is the classification of sleep in predefined stages according to its depth, which forms an important basis for the detection and thus the treatment of sleep disturbances in general. A direct way to detect the sleep disturbances apnea is the detection of nightly apnea-hypopnea events, which are characterized by total or partial cessation of the respiratory flow during sleeping.

Up to now, the detection and thus the treatment of sleep disturbances is very costly, time and effort consuming, because patients have to spend several nights in sleep labs, with dedicated systems and attending staff, where their physiological signals are recorded. These so-called polysomnogram, which usually contain electroencephalography (EEG), electrocardiography (ECG), electrooculography (EOG), electromyography (EMG) and other recordings, have to be analyzed by medical scientists after each night. As first step in this analysis, a hypnogram is created by sleep stage scoring, i.e. every 30 second lasting epoch has to be classified according to a predefined system [RK68] in its sleep stage, dependent on the depth of sleep. Up to now, reliable sleep stage scoring is done by experts by hand, i.e. every morning an expert has to analyze visually 960 epochs of an usually 8 hours lasting polysomnogram in order to create a hypnogram, and this has to be done for every polysomnogram recorded that night in the lab.

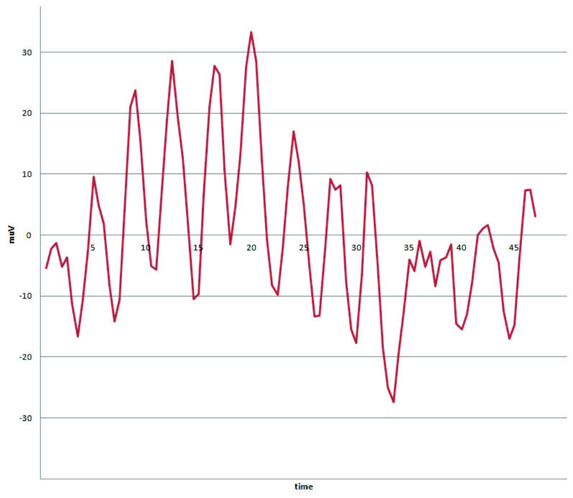
The contribution of this work is an approach for automatic sleep stage scoring and apnea-hypopnea detection [SC10a, SC12a], which applies several data mining techniques and the rules by Rechtschaffen and Kales [RK68] (which are also used by experts for manual scoring) to polysomnographic data in order to disburden the over-worked sleep expert in the classification of sleep epochs. The basic idea of our approach follows the R&K rules, i.e. after a preprocessing step, several special features, like frequencies, k-complexes and sleep spindles, which characterize certain sleep stages, are extracted from the data by means of several combined techniques (Fourier and wavelet transform for frequency analysis, derivative dynamic time warping (DDTW) [KP01] and waveform recognition [HKY02] for pattern and wave detection). Two decision tree ensembles are constructed on these pieces of information, one in order to classify every epoch independently in its sleep stage and one to annotate minutes, which contain events of apnea or hypopnea. After that, case-based reasoning is applied as postprocessing step to incorporate the context-based rules from the R&K system and to improve quality in ambiguous cases. We tested and evaluated our approach with different parameters on three large public databases from PhysioBank [GAG+00], which showed an overall accuracy of 95.2% for sleep stage scoring and 94.5% for detecting minutes of apnea-hypopnea, what is very good in comparison to related approaches (which are discussed in subsection 7.2.3 in detail).

The rest of this section is organized as follows Subsection 7.2.1 presents the required physiological background and terminology from the area of sleep research. Related work and previous approaches for automatic sleep stage scoring and apnea-hypopnea detection are presented in subsection 7.2.2. The following subsection 7.2.3 contains the basic idea, used techniques and details of our approach for automatic sleep stage scoring and apnea-hypopnea detection. An evaluation on the three databases from PhysioBank is given in subsection 7.2.4. Subsection 7.2.5 concludes this section and states future work.

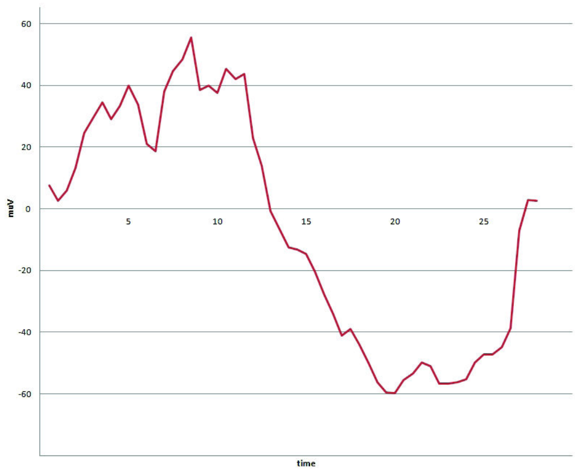
7.2.1 Physiological background

After several decades of sleep research, it is known that sleep is by far not a static condition, linear or even random, but a complex process with a recurring, ultradian periodicity [Bor86]. This periodicity consists of regular changes from rapid eye movement (REM) to non-REM sleep phases. For healthy persons, this periodicity comprises approximately four to five cycles over an average sleep duration of about eight hours. The cycles can be divided into deepening and ascending phases. Before the deepening phase, during relaxed wakefulness, the EEG (which displays electrical brain activity) shows a typical α-wave rhythm (cf. subsection 7.2.3 for the exact frequency ranges) and increased activity in EOG (which displays eye movements) as well as in EMG (muscle tension). With closed eyes a reduction of the EOG deflection is measured, and the longer the body relaxes, the more a pendulum movement of the eyes can be measured, indicating drifting into sleep. The α-basic-rhythm changes to slower waves with lower amplitudes, which characterizes the beginning of the sleep deepening phase and, according to the sleep stage classification system by Rechtschaffen and Kales[1] [RK68][8], the first sleep stage S1. After a short time staying in stage 1, the amplitudes of the EEG waves increase and first characteristics of the second stage S2 become visible: sleep spindles and k-complexes. Sleep spindles are wave patterns with medium frequency and low amplitude, which superimpose EEG in an episodic rhythm, k-complexes are characterized by a slow and a high deflection, both positively and negatively labeled (cf. figure 7.6). Muscle tension continues to decrease and eye movements become less common or disappear entirely. A little later, wave frequency decreases and amplitude

[81]In 1968, Rechtschaffen and Kales published a system for sleep stage scoring, which divides sleep in five non-REM stages (wake, S1-S4) and one REM stage. It contains rules for scoring these stages on base of special features occurring during sleep, and found world-wide acceptance since that time.



(a) Sleep spindle



(b) k-complex

Figure 7.6: Illustration of two characteristic patterns for sleep stage scoring.

increases, the so called δ-waves occur. These δ-waves are a typical feature for slow-wave sleep, to which the two following deep sleep stages S3 and S4 are counted. Stages 3 and 4 differ only in the quantity of delta waves occurrences: if the relative frequency of δ-waves in an epoch is about 20-50%, the stage is classified as S3, wheras it is classfied as S4 at a higher proportion. A person remains up to 50 minutes in stage 4. EOG does not show significant activity in S3 and S4. After reaching S4, the deepening phase is completed and an increasing number of movements can be measured. Movements during sleep usually occur in the transitions from stage 2 to 3 and 4 and also in the final stage of the deepening phase (S4). This sequence of sleep stages is typical for the deepening phase, but it is quite usual too, that the body returns in a previous stage, if it is not yet physiologically adjusted to the current stage. The skipping of stages in the deepening phase is not usual for healthy persons, but it is usual in the ascending phase, which follows after the deepening phase. After staying a while in stage 4, sleep gets lighter and "climbs" up to stage 2, where often stage 3 is skipped. Stage 2 is followed by REM sleep stage, which is the last stage of the sleep cycle. It is characterized by a flat EEG, similar to stage 1, rapid eye movement, sawtooth waves and a complete decrease of muscle tension.

This first ultradian sleep period ends after about 90-100 minutes and the next one begins. The following periods become flatter during the night, the deep sleep stages 3 and 4 are usually achieved only in the first two periods. Thus the percentage of deep sleep in the morning becomes shorter and the proportion of the light and REM sleep increases. During the fourth and fifth cycle, it may even happen that the waking state is reached for a very short time; but this is not perceived by the subject due to the short length of stay. Figure 7.7 illustrates the typical sequence of sleep stages for the

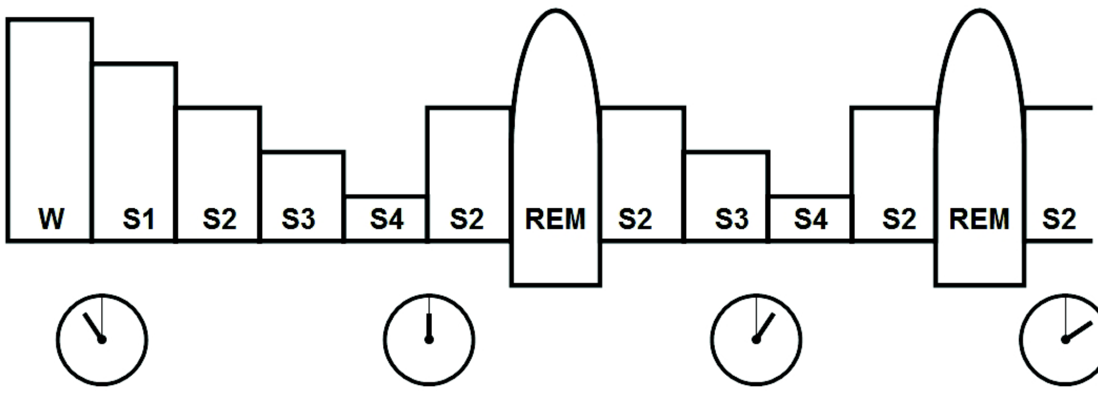


Figure 7.7: "The stairs of sleep": Illustration of the sequence of sleep stages in the first three hours of sleep (adapted from [Bor86]).

first three hours of sleep.

One way to classify epochs into sleep stages is to regard every epoch independently from the surrounding epochs, extract certain features (predominant waveforms, special patterns like k-complexes) and score the epochs according to the applicable rules by Rechtschaffen and Kales (cf. table 7.5 for an overview of the characteristics of the single sleep stages). Another way is to regard the contextual information additionally, given by the typical sequence of sleep stages as mentioned in the preceding paragraph and by the case-based rules by R&K. Case-based reasoning affects for instance the beginning and end of REM sleep, which have to be treated specially under some conditions, and a rule for treating S1 intrusion in sleep stage 2 (see [RK68, Bor86] for more details). Especially in cases, where a classifier is uncertain, e.g. because of too many movement arousals or electrode pops, case-based reasoning increases accuracy eminently.

stage	EEG (frequency)	EOG (frequency)	EMG (tension)	special characteristic
W	very high	high	high	none
S1	high	low	medium	vertex waves
S2	high	low	medium	vertex waves, sleep spindles and k-complexes
S3	low	very low	low	sleep spindles
S4	very low	very low	low	sleep spindles
REM	unsynchronized	high	very low	sawtooth waves

Table 7.5: A short overview of sleep stages and their characteristics according to Rechtschaffen and Kales.

One of our contributions is the approach for automatic sleep stage scoring presented in subsection 7.2.3, which generates hypnograms and gives indications for general diagnosis by comparing patient's hypnograms with a "normal" hypnogram and further statistical values. The other contribution is more specific to the sleep disease apnea: our approach detects occurrences of apnea-hypopnea events, which yields a direct diagnosis proposal, if a person suffers from mild obstructive sleep apnea-hypopnea syndrome (OSAHS), moderate OSAHS or severe OSAHS. (Of course, this approach should not replace a medician, but save her or him a lot of work.)

According to several studies sleep related breathing disorders have a high prevalence in the adult population (4% male, 2% female in general [YPD+93]). Obstructive sleep apnea (Greek: $\alpha\pi\nu o\iota\alpha$, from α-privative and $\pi\nu\epsilon\epsilon\iota\nu$-breathe), which is characterized by a repetitive cessation of the respiratory flow during sleep, is the most common of the different types. Due to a total or partial collapse of the upper airways at the level of the oropharynx during sleep (the so-called apnea or hypopnea, respectively), a patient cannot breath anymore or too less, which continues until the lack of oxygen and the increase of CO_2 causes a central nervous activation, called arousal. This activation reestablishes respiration, but in most of the cases the patient does not reach the level of conscious wakefulness in order to perceive this. Apnea can last up to 120 seconds in REM sleep, when chemosensity is lower. For patients with severe sleep apnea up to 600 apneas per night are recorded [PMM+00]. Apneas led to sleep fragmentation and loss of deep and REM sleep, sleep looses its restorative function on physical and mental perfomances and chronical diseases like obesity, hypertonsion and arrhytmia can occur.

Due to its nature, every apnea is accompanied by a drop in oxygen saturation and respiration itself, which can be measured in a sleep lab with an esophageal balloon or a full-face mask, what of course is very invasive for the patient. Thus, approaches have been made to detect events of apnea and hypopnea only from ECG (electrical heart activity), which can be measured more easily and even at home (cf. subsection 7.2.3). During an apnea, a drop of heart rate is observed [PMM+00], which is followed by an increase of heart rate due to the lack of oxygen, until the central nervous activation occurs. This happens cyclically with every apnea and is known as the cyclic variation of the heart rate. According to criteria by the American Academy of Sleep Medicine Task Force [oSMTF99] patients are diagnosed with obstructive sleep apnea, if they have 5 or more apnea events per hour of sleep during a full night sleep period, where each apnea event is defined as a respiratory pause lasting at least 10 seconds. (5-15 events per hour are diagnosed as OSAHS, 15-30 as moderate OSAHS and above 30 as severe OSAHS.)

7.2.2 Related work

One of the first approaches of computer-aided sleep stage scoring was proposed in 1971. The authors of [SK71] presented a "special purpose Sleep Analyzing Hybrid Computer" (SAHC), which uses three EEG and two "eye channels" (comparable to EOG) as input. The SAHC was able to process data up to 32 times faster than realtime and bases mainly on linear analog filters for the detection of sleep spindles, α- and δ-waves. The evaluation on 15 overnight polysomnograms from 15 healthy subjects, aged 15-21, showed an agreement of all scored sleep stages (S1 and REM combined) between their system and human scoring of 83.5%. In comparison to the results of the following related approaches, this accuracy is quite high, especially with regard to the date, but there are two weak points: 1.) The authors combined the stages S1 and REM into one single stage, which makes it very easy for their system, because especially these two stages are more similar than for instance S1 and S4, and it is thus more difficult to separate them. Most of the other approaches try to separate these classes, which decreases their overall accuracy in comparison to this approach. 2.) The group of subjects for the evaluation was between 15 and 21 years old and healthy. Studies show, that it is easier to classify sleep stages from young and healthy subjects, than from older ones or with sleep disorders [AGP$^+$05].

A very interesting approach and one of the first incorporating case-based reasoning is [POJP00], which consists of two components: one rule- and one case-based component, both trying to convert the rules by Rechtschaffen and Kales. The inputs are the four parameters typically used in sleep stage scoring, namely EEG, ECG, EOG and EMG. In an evaluation on a rather small dataset containing mixed polysomnograms of three healthy patients and three patients, who suffer from obstructive sleep apnea, their system reaches an average agreement rate of 87.5% without case-based reasoning and 93.1% with case-based reasoning.

In 2002, an algorithm for waveform recognition and its application to automatic sleep stage scoring was proposed in [HKY02], which has some similarities to our approach. The waveform recognition method is used in order to compare the waves of an epoch with sample waves for α-waves, β-waves etc., and finally a forest of decision tree classifiers is used to determine the sleep stage, which is done with an average accuracy of 81.5%. For stages, that occur more often (e.g. S2, S3), they report an even better accuracy. Since we are learning our decision tree ensemble on a larger dataset and since we are using more techniques properly combined (like frequency analysis and DDTW for additional pattern matching, cf. subsection 7.2.3), our approach increases accuracy by more than 10%.

A simple three step algorithm for classifying epochs in awake, REM and non-REM sleep was presented in [LLS04], which yields quite good results and an accuracy of

87.9%, and an accuracy of 90.1% in classifying just into the two stages wake and sleep (sleep = REM + non-REM). Another approach that classifies into the stages wake, deep and REM sleep, is [FGD05]. This approach does not base on the rules by Rechtschaffen and Kales, but on an Hidden Markov Model, it uses a resolution of 1 second (instead of 30 second epochs) and relies only on one single EEG signals. The authors report an accuracy of 80% and with that they claim that sleep staging with only one single EEG signal is reliable. Regarding the high disagreement rate between two different experts scoring manually (which is about 70-87% according to studies [HKY02, AGP$^+$05]), especially between experts from different sleep labs, they seem to be right.

An expert system with expert interaction in uncertain cases, which has reached commercial state meanwhile, is the Somnolyzer 24 x 7 from the Siesta Group [AGP$^+$05]. The core of this system uses a decision tree as preclassifier, which is optimized by expert knowledge, and then case-based reasoning is applied, too. Dependent on the quality of the data and in uncertain cases, they use a quality control module, i.e. experts inspect the questionable epochs manually. The reported accuracy on a large mixed database is 78.3% without and 79.6% with quality control.

In 2010, we proposed a framework for automatic sleep stage scoring [SKC10], which is the antecessor of the system presented in this section. It bases on frequency analysis, pattern recognition and an ensemble of decision trees for classification as well, but the main differences are: it considers less patterns, that are of interest for sleep staging (e.g. no sawtooth or vertex sharp waves, cf. subsection 7.2.3), performs no case-based reasoning and no apnea-hypopnea detection. The evaluation on a rather small database shows an accuracy of 92% for sleep stage scoring learned and tested on data from the same patient and 84% learned on one patient and tested on another. Some more related work about other approaches for automatic sleep stage scoring can also be found in [SKC10] and in [AGP$^+$05].

From our point of view, computer-aided apnea detection on base of oxygen saturation or respiration itself, which have to be measured in sleep labs, is due to the nature of apnea not challenging for data mining. Thus here we just focus on the detection of apnea on base of ECG signals, which can even be measured at home.

In 2000 the Computers in Cardiology (CinC) Challenge [PMM$^+$00] has been arranged in order to develop and evaluate ECG-based apnea detectors, and a large database of mixed polysomnograms (polysomnograms of completely healthy subjects, of subjects with mild OSAHS and with severe OSAHS) has been made open to the public [GAG$^+$00] for this challenge. The competitors had to solve two tasks, the first was to determine if a patient suffers from apnea, and the second to mark every minute of the polysomnograms, in which an event of apnea occurs. Several algorithms have

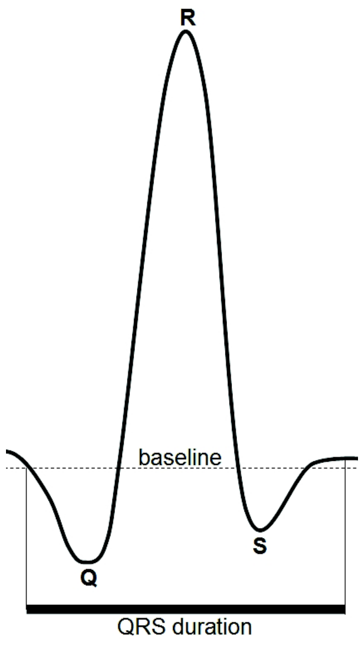


Figure 7.8: Illustration of the QRS-complex, a typical deflection of EEG, which is named after it components, the so-called Q, R and S wave.

been proposed for this contest and a systematic comparison and rating of these algorithms has been published in [PMdC+02]. The algorithms that performed best in the contest all based on the combination of frequency-domain parameters of either heart rate variability or the respiratory signal derived from ECG with R-wave morphology (the R-wave is the large positive deflection in the QRS-complex, which can be used to separate each heartbeat due to its characteristic form, see figure 7.8). The top three algorithms in minute by minute comparison were [MF00], [RCBC00] and [CHS+00] and reached an accuracy of 92.6%, 92.3% and 89.4%, respectively. The combination of these three algorithms reached an accuracy of 93.1%. Another approach that yielded a little bit worse results at the contest, but which is directly available as software package through PhysioNet [GAG+00], is [MPIG00] with an accuracy of 84.5%.

Another frequency-based approach is [ZvEWJ04], which uses Fourier and wavelet transform. The authors evaluate their approach on the CinC Challenge dataset and report an accuracy of 93.3% on the clear cases (i.e. somnograms of patients who either do not suffer from sleep apnea or who suffer from severe OSAHS) and 92% on the average for minute allocation.

More recent approaches are [KGP09] and [MCH+10]. The first one bases on wavelet analysis and a two-staged feedforward neural network and yields an accuracy of 94.7% for the independent detection of apnea and 79.8% for hypopnea. The second approach proposes the application of the empirical mode decomposition for apnea detection and compares it with the estabished wavelet analysis, where wavelet analysis yielded slightly better results in marking minutes as apneic or non-apneic with an accuracy of 90%.

7.2.3 Our approach

a) Basic idea

For sleep stage scoring, our approach directly converses the rules by Rechtschaffen and Kales in order to simulate the proceeding of an expert manually scoring polysomnograms. On the one hand special patterns, that are important for sleep stage scoring, i.e. k-complexes, sleep spindles, sawtooth waves, α- and δ-waves, vertex sharp waves, slow eye movements (SEMs) and REMs, are extracted by two different techniques (DDTW and waveform recognition, cf. subsections 7.2.3 d) and e)), and on the other hand background activity is analyzed by frequency analysis techniques (Fourier and wavelet transform, see subsections 7.2.3 b) and c)) in order to determine the predominant activity and the percentage of occurring waves. To do so, the EEG signal is band-pass filtered into the five important frequency bands for EEGs (0.5-2 Hz for δ-band, 2-7 Hz for θ-band, 7-12 Hz for α-band, 12-20 Hz for β-band and 20-40 Hz for fast β-band). In addition to that, the overall activity of EMG and EOG is determined and used as index. All these pieces of information form the features (cf. table 7.6 for an overview of all the extracted features), on which an ensemble classifier (bagging with decision trees, cf. subsection 7.2.3 f) for details) is learned.

Since case- or rule-based reasoning is know to improve accuracy in several domains (e.g. in sensor networks [CGKL11] and web services [EB10]), we are applying case-base reasoning on base of the case-based rules by Rechtschaffen and Kales [RK68] and the typical sequence of sleep stages after preclassifying each epoch independently.

For the detection of apnea-hypopnea on base of the ECG signal, some more preprocessing has do be applied (cf. subsection 7.2.3 g) for details). For every single minute our algorithm extracts frequency feature from the VLF band (very low frequency: 0.01-0.08 Hz, the most important band for apnea-hypopnea detection [MCH$^+$10]), the LF band (low frequency: 0.08-0.15 Hz) and the HF band (high frequency: 0.15-0.4 Hz), and the standard selected time domain measures [Tas96]. These features again are used to create an ensemble of decision trees which are used to mark every minute containing apnea-hypopnea. Since studies suggest, that there is no relationship between the distribution of apnea-hypopnea events and different sleep stages [WGP$^+$05], no postprocessing step is required here.

Figure 7.9 gives an overview over our system architecture for sleep stage scoring and apnea-hypopnea detection.

b) Fourier transform

The Fourier transform transforms a function of a real variable with complex values into another. In signal processing it is commonly used for transforming functions in the time

Name	Meaning
eeg-feature-alpha	occurrences of alpha waves
eeg-feature-alpha2	occurrences of alpha waves
eeg-feature-beta	occurrences of beta waves
eeg-feature-fbeta	occurrences of fast beta
eeg-feature-theta	occurrences of theta waves
eeg-feature-delta	occurrences of delta waves
eeg-feature-delta2	occurrences of delta waves
eeg-feature-spindle	occurrences of sleep spindles
eeg-feature-kcomplex	occurrences of k-complexes
eeg-feature-sawtooth	occurrences of sawtooth waves
eeg-feature-vertex	occurrences of vertex sharp waves
emg-feature	overall EMG activity
eog-feature	overall EOG activity
eog-REMs	occurences of REMs
eog-SEMs	occurences of SEMs

Table 7.6: Extracted features for sleep stage scoring.

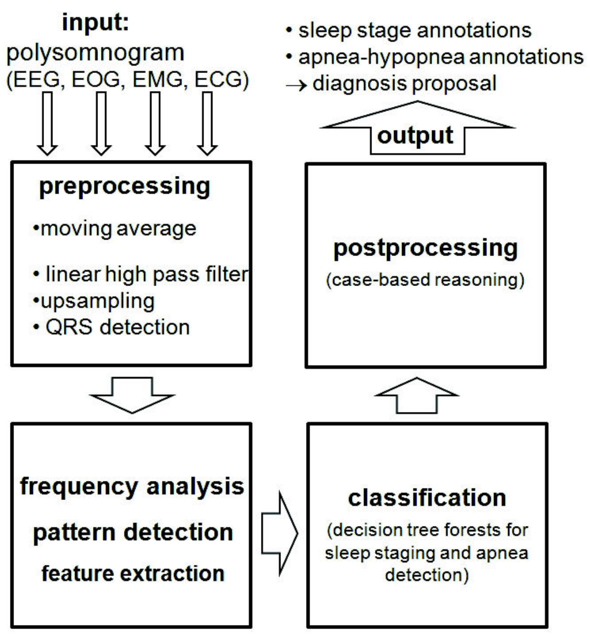


Figure 7.9: System architecture of our approach for automatic sleep stage scoring and apnea-hypopnea detection.

domain into functions that represent the original functions in the frequency domain. For sleep stage scoring the discrete Fourier transform can be applied to determine the percentage of each frequency band in the time discretized EEG signal.

There are two problems while applying the Fourier transform to EEG signals. One of them is the so-called leakage effect. The Fourier transform is based on periodic functions, which starts and ends with the same amplitude, and which duration is a multiple of the period. This cannot always be provided because of the discretization of the signal. The signal has to be regarded in sections, which is equal to a convolution in the frequency domain in the mathematical sense, i.e. the multiplication of the time signal with a rectangular function. This results in zeros on non-multiples of the period, which leads to smearing of the frequency spectrum. This smearing again leads to a broadening of the frequency base and to a decrease of the amplitudes of the available frequencies. The leakage effect can be minimized by using an appropriate window functions, such as the Hann or Hamming window. We use the windowed Fourier transform in our implementation.

The other problem is the loss of temporal progression after applying the Fourier transform, i.e. the result contains merely the added amplitudes of the corresponding frequency components. EEG waves in the human brain differ not only in frequencies, but also in their amplitudes. Low frequency waves for instance have a higher amplitude than high frequency waves [MP95], thus the result of a Fourier transform often reflects the delta frequency as the dominant band. The problem with this is, that the frequencies, which have the biggest percentage in the whole activity, are not as decisive as those, which occur most frequently in one epoch, thus the sum of amplitudes of frequencies of a Fourier transform is not an unambiguous feature to classify an epoch.

c) Wavelet transform

A method that keeps reference to the temporal domain is the wavelet transform. The wavelet transform is an advancement of the short-time Fourier transform, which - as the name indicates - is again an advancement of the Fourier transform. The short-time Fourier transform solves the problem of the ordinary Fourier transform, that does not give any information about the temporal occurrence of frequencies, by means of a sliding window. Using a window function, the short-time Fourier transform divides the whole signal into small portions, on which the ordinary Fourier transform is applied. In contrast to the short-time Fourier transform, the wavelet transform uses a window, which is called mother wavelet, that is not only shifted, but shifted and scaled. For our purposes, we use the discrete version of the wavelet transform.

More details about the discrete wavelet transform, the Fourier transform, its derivates and signal processing in general can be found in [BNDD02] and [Lyo04].

d) Derivative dynamic time warping

The sleep stage scoring rules by Rechtschaffen and Kales do not only refer to frequencies, but also to special patterns, which can be found in the EEG data, like k-complexes and sleep spindles (cf. figure 7.6 for an illustration of these two patterns and subsection 7.2.3 a) for further special patterns), which occur only in certain sleep stages and thus are essential for the classification process (cf. table 7.5). One way to identify them is to compare previously derived samples of the special patterns with sequences from the EEG data by using an appropriate distance measure. If the distance is below a certain threshold, a match is assumed.

As already mentioned in section 3.2, the Euclidean distance is not suitable for comparing curves if minor variations and shifts are observed. Unfortunately, this is exactly what is to be expected in EEG patterns: two special pattern, e.g. sleep spindles, from one patient will look very similar, but they will never be exactly the same. A distance measure which is especially designed for this scenario, is the derivative dynamic time warping (DDTW) distance, which has been proposed in [KP01] (also cf. section 3.2). The derivative dynamic time warping is an advancement of the dynamic time warping distance (DTW) [SC78], which regards the values of the derivations at the non-linearly aligned points of two time series. This leads to a very robust distance measure that takes more into account the shapes of two time series and less the differences in the y-axis. Sleep spindles and k-complexes can occur with different amplitudes, but their shapes still remain, so in our scenario the derivative dynamic time warping distance is very appropriate since it allows comparison with a certain degree of tolerance.

e) Waveform recognition

Another way to detect patterns like k-complexes, sleep spindles or special types of waves, is the recognition of waveforms. Medicians tried to detect waveforms aided by computers since many years. Unlike approaches, which mostly use mathematical algorithms for extracting frequency components, the waveform recognition method tries to detect the waveforms reliably by regarding each single wave pattern as independent object. In this domain the idea for this approach is based on the assumption that biological signals of human body cannot be described merely by sinusoids functions. The electrical brain activities are produced by many different ambiguous qualities, which are linked with each other with complex relationships. Thus a connection between certain events in human body and extracted features, such as frequency components, can be reconstructed only difficulty. The visual recognition of waveforms has been used since many years because experts are trained to detect specific patterns for the study of EEG signals, which leads us to the decision to use the investigation of wave patterns as individual objects as component for our approach.

An interesting approach for waveform recognition, which we are adapting, is proposed in [HKY02]. It is based on the zero-crossing method [FYIT58] presented in 1958, which was refined due to the lack of opportunities to identify slow waves. In [HKY02] the signal samples are analyzed for their peaks and valleys, and then they are stored as objects from a valley, a peak or another valley. The recognition of patterns is realized by using a predefined set of parameters which describe the desired patterns of a wave. Once a pattern is identified, it is stored and assigned to the corresponding wave class. For applying the rules by Rechtschaffen and Kales, we used the waveform recognition method in order to identify the amount of special patterns and waves (α- and δ-waves) in the EEG data. The final number of occurrences of the different classes within a period can be used in order to classify the current sleep stage.

f) Classification

A common problem in classification is the under- or over-representation of certain classes in the training data, which can be expected especially in the case of sleep stage scoring (since patients stay varying durations in the different sleep stages, cf. section 7.2.1). A common way to lessen this problem is to use an ensemble of several simple classifiers (e.g. like in [GT11, ZL10, KTV10]), which directly reduces overfitting (cf. for instance [Mit06]) and improves the overall performance of the classification process additionally. Four of the major ensemble classifying methods are bagging [Bre96], boosting [FS97], random forests [Bre01] and random subspace [Ho98], where bagging has been shown as slightly superior, especially in situations with noise [Die00] (what can be expected in our situation) and when using the C4.5 algorithm [Qui96] for decision trees on EEG data [Sun07]. Since decision tree induction is one of the historically first machine learning approach, which is still used widely (in practically all commercial analytical products) [RJ11], we decided to use bagging with decision trees as classification process in our approach.

Similar to the technique of cross validation, bagging divides the training data into several training sets, where for each training set instances from the original data are sampled uniformly with replacement. Thus, some instances of the training data appear repeated in a training set, and others not at all. On base of these training sets, a decision tree forest is constructed, which decides by majority vote to which class an instance has to be allocated.

It should be noted that for the numerical values of various attributes good splitting values have to be chosen. To do that, a fixed limit is set during the search for an appropriate split attribute after regarding a certain amount of training data. This reduces the running time (with minimal loss of classification quality) and is used to investigate the information gain. The limits are chosen uniformly over the remaining interval of the split attribute, i.e. the entire interval from the smallest to the largest value is divided into equal subintervals and the information gain is calculated at their borders. Below a certain threshold of training instances, all remaining instances are used to calculate the information gain.

g) Further implementational details

We implemented our approach in Java on a standard desktop PC (Intel Core 2 Duo 3.0 GHz running Windows 7 with 3 GB of main memory) using parts of the PhysioToolkit WFDB [GAG+00]. The WFDB software package provides direct access to the data stored in PhysioNet from Java without prior downloading (which of course costs a little bit more time than storing and processing it from disk) and the support of PhysioBank compatible file formats (among others: binary signal files, header and annotation files, EDF and up to a certain degree EDF+[9]).

As already mentioned at the beginning of this section, our prototype system uses EEG, ECG, EOG and EMG signals as input. For sleep stage scoring, the EEG signal is analyzed and reduced to the corresponding feature values, as described in subsection 7.2.3 a). To avoid leakage effect and associated errors for applying the Fourier and wavelet transform and in order to reduce complexity, the data is preprocessed with Hamming window and moving average. For the determination of the special patterns, which constitute essential features for the process of sleep stage scoring, predefined patterns are compared with EEG data of each epoch and then again an index is calculated for the amount of occurrences of these patterns (subsection 7.2.3 a)). Due to the high complexity of EEG data a smoothing is required by calculating the moving average. In the further process the extracted feature values (cf. table 7.6 for an overview of the extracted features for sleep staging) are used to learn an ensemble of decision trees, as described in subsection 7.2.3 f).

For the detection of apnea-hypopnea events from ECG, our approach uses as preprocessing step a linear phase high pass filter with a cutoff frequency of 0.5 Hz in order to remove baseline wander. As positive side effect of this procedure we obtain an indirect indication of the respiration cycles, because ECG shows a slow modulation of the amplitude due to an influence of the ECG from respiration movements [MMB+86]. After filtering we interpolate the signal with cubic splines [ANW67] to redigitize it at a frequency of 500 Hz, then QRS-complexes are detected and RR-intervals (RR-intervals are the intervals between two successive R-waves, see figure 7.8) are defined for further frequency analysis. (For details about the signal processing methods used confer for instance [Lyo04].)

[9]EDF (European Data Format) [KVR+92] and its enhancement EDF+ [KO03] are two standard file formats for storing and exchanging medical time series.

7.2.4 Evaluation

We evaluated our approach on three large databases: the MIT-BIH Polysomnographic Database [IM99], the St. Vincent's University Hospital / University College Dublin Sleep Apnea Database and The Sleep Heart Health Study Polysomnography Database, which are all open to the public via PhysioBank [GAG+00]. These databases consist of several overnight polysomnograms with various recordings from both healthy and sleep-disordered subjects of all ages. Since we need EEG, ECG, EOG and EMG signals and annotations of sleep stages and apnea-hypopnea events for evaluating our approach, and since these parameters are not contained in a small fraction of these databases, we could only use a subset of the entries, which consists of 31 mixed polysomnograms with a total duration of 279 hours and 22 minutes. Thus our test data consisted of exactly 33,542 epochs, which had to be classified into their sleep stages, and 16,762 minutes, which had to be marked as apneic or non-apneic.

For evaluating the sleep stage scoring functionality of our approach, we performed three experiments. At first our system had to classify sleep stages independently without case-based reasoning. Table 7.7 gives the scoring agreement between manual scoring and our approach (without case-based reasoning) as confusion table in detail. (As in all following confusion tables in this section, the correct class of an epoch, which has to be classified, is given on the left side of the table, while the percentage of classes, to which this particular epoch was classified, is depicted in the particular column.) In this course, our system reached an overall accuracy of 89.9%.

In the second experiment we applied case-based reasoning additionally, which increased the overall accuracy to 95.2%. Table 7.8 gives the results in detail. In our third experiment, we used case-based reasoning and 5-fold cross validation (instead of bagging) for the creation of the decison tree classifier, which led to an decrease of overall accuracy to 92.6%. The results are given in table 7.9 in detail.

Obviously the second experiment led to the best result, because it involved the

	W	S1	S2	S3	S4	REM
W	97.0	2.4	0.6	0.1	0.0	0.5
S1	9.1	58.1	20.2	0.8	0.2	11.6
S2	0.5	4.7	91.7	5.5	0.8	0.2
S3	0.0	0.1	20.2	62.8	18.2	0.1
S4	0.1	0.2	1.0	12.6	86.8	0.1
REM	0.7	2.3	3.0	0.1	0.0	96.6

Table 7.7: Results of exp. 1 (w/o cb. reasoning, ∅ acc. 89.9%).

	W	S1	S2	S3	S4	REM
W	97.0	2.4	0.6	0.1	0.0	0.5
S1	5.1	75.1	14.2	0.5	0.2	5.6
S2	0.5	2.5	96.7	2.1	0.6	0.2
S3	0.0	0.1	20.2	64.8	15.2	0.1
S4	0.0	0.2	0.3	7.3	92.1	0.1
REM	0.4	1.2	1.7	0.1	0.0	98.6

Table 7.8: Results of exp. 2 (with cb. reasoning, ∅ acc. 95.2%).

additional step of case-based reasoning. As assumed, case-based reasoning improves the quality of classification, especially in such cases, where the classifier is uncertain. Comparison of the results from experiment 2 and 3 show that bagging outperforms cross validation in this domain, but compared to experiment 1, cross validation with case-based reasoning is still better than bagging without case-based reasoning.

The differences between certain stages, or the opposite respectively, e.g. the similarity of the stages S1, S2 and REM, can be found in the detailed results of all three experiments: The more similar two stages are, the harder it is to separate them from each other, which leads to a decrease of accuracy. A particular stage is S2, which can be denoted as compound state and thus it is very difficult to distinguish it for instance from S1 or S3. This may be due to the special characteristics of S2, which are the occurrences of k-complexes and sleep spindles, which are very short events and thus might be overseen.

Since persons do not stay the same duration in every sleep stage, naturally there are some stages, which have more instances in the training set, e.g. stage 2. Regarding the better accuracy for classifying these stages in contrast to e.g stage 3, it appears

	W	S1	S2	S3	S4	REM
W	97.7	2.1	0.4	0.1	0.0	0.3
S1	7.2	63.2	19.3	0.4	0.2	10.2
S2	0.4	4.7	91.2	4.5	0.8	0.2
S3	0.0	0.1	20.1	65.6	16.3	0.1
S4	0.1	0.1	0.6	7.0	94.2	0.0
REM	0.7	2.3	2.5	0.1	0.0	97.9

Table 7.9: Exp. 3 (cb. reasoning + 5FCV, ∅ acc. 92.6%).

that accuracy increases with the number of instances available for training.

For evaluating the importance of the different features for sleep stage scoring (table 7.6) on the classification accuracy we conducted a further set of experiments, where we used bagging with decision trees and case-based reasoning, but neglected some features. Since sleep spindles and k-complexes play a very important role in classifying sleep stages manually, these two features were our choice for the first experiment. Neglecting the accordant features eeg-feature-spindle and eeg-feature-kcomplex in deed decreased the overall accuracy, but with only 3.9% less then expected. Sleep stage S2, which is characterized by the appearance of these two features, for instance is still detected with a relatively high precision, inspite of neglecting these characteristic features. This fact may be explained by an observation of the extraction process via waveform and DDTW pattern recognition: the complexity of the signals is extremely high and therefore it is necessary to scale and smooth the data before applying these two pattern recognition methods. Due to recording length and smoothing, it is just a matter of time until a wave pattern is found, that looks very similar to one of the derived sample patterns, although it represents no sleep spindle or k-complex. Since it is not yet clear how k-complexes and sleep spindles are caused, we cannot make statements about the exact parameters of these patterns, but have to derive them from samples, which leads to a non-perfect automated recognition.

In our experiments the feature with the highest influence on classification accuracy was emg-feature, which quantifies overal EMG activity: neglecting this features decreased accuracy by 19.2%, followed by the EOG features (eog-feature, eog-REMs and eog-SEMs) with 13.2%.

The results of the apnea-hypopnea detection functionality of our approach can be stated quickly: Using bagging with decision trees, our approach reached an accuracy of 94.5% in minute by minute comparison, which beats 5-fold cross validation with an accuracy of 93.4%.

7.2.5 Conclusion and future work

In this section we proposed an approach for automatic sleep stage scoring and apnea-hypopnea detection. Our approach converts the rules by Rechtschaffen and Kales, which are used by experts for manual sleep stage scoring, and thus extract certain patterns and frequencies, which are of special interest for sleep stage scoring, from EEG, ECG, EOG and EMG data. It classifies epochs of sleep by using an ensemble of decision trees and postprocesses these pre-results with case-based reasoning. For validating our approach, we conducted an evaluation on a public dataset, which consists of a large number of records from healthy and sleep disordered persons of every age. This evaluation showed a very good overall accuracy of 95.2% for sleep stage scoring, which is even much better than the agreement rate between two human scorers (70-87%). The detection of minutes of apnea-hypopnea occurences yielded a comparable good result with an accuracy of 94.5%.

Automatic sleep stage scoring and apnea-hypopnea detection can significantly reduce the work of experts working in sleep labs, facilitate diagnosis and treatment of sleep disturbances and thus save a vast number of money, time and even lives (e.g. regarding the accident statistics due to micro sleep during driving). For further improving our approach, we are revising feature parameters and the used features themselves, the detection of special pattern (especially the waveform recognition method) and the classification procedure. In addition to that, further evaluation has to show, how our approach works with less input data, e.g. only with EEG. Reducing the effort of sleep staging in a sleep lab is one desirable thing, but developing methods, that spare patients spending invasive nights in sleep labs, would make sleep diagnosis even more applicable.

List of Own Publications

2012

- T. Schlüter, S. Conrad: An Approach for Automatic Sleep Stage Scoring and Apnea-Hypopnea Detection (extended journal version), Frontiers of Computer Science, 2012

- T. Schlüter, S. Conrad: Hidden Markov Model-Based Time Series Prediction Using Motifs for Detecting Inter-Time-Serial Correlations, Proc. ACM Symposium on Applied Computing (SAC), Riva del Garda (Trento), Italy, March 26-30, 2012

2011

- T. Schlüter, S. Conrad: Applying Several Kinds of Temporal Association Rules to Time Series Analysis, Proc. IADIS European Conference on Data Mining (ECDM'11), Rome, Italy, July 24-26, 2011

- T. Schlüter, S. Conrad: About the Analysis of Time Series with Temporal Association Rule Mining, Proc. IEEE Symposium Series on Computational Intelligence 2011 (SSCI 2011), Paris, France, April 11-15, 2011

2010

- T. Schlüter, S. Conrad: An Approach for Automatic Sleep Stage Scoring and Apnea-Hypopnea Detection, Proc. ICDM 2010, The 10th IEEE Int. Conference on Data Mining, Sydney, Australia, December 14-17, 2010

- T. Schlüter, T. Kißels, S. Conrad: AS3: A Framework for Automatic Sleep Stage Scoring, Proc. IADIS European Conference on Data Mining 2010, Freiburg, Germany, July 28-30, 2010

- T. Schlüter, S. Conrad: TEMPUS: A Prototype System for Time Series Analysis and Prediction, Proc. IADIS European Conference on Data Mining 2010, Freiburg, Germany, July 28-30, 2010

- T. Schlüter, S. Conrad: Mining Several Kinds of Temporal Association Rules Enhanced by Tree Structures, Proc. Int. Conference on Information, Process, and Knowledge Management (eKNOW 2010), Saint Maarten, Netherlands, Antilles, February 10-15, 2010

2009

- K. Zaiß, T. Schlüter, S. Conrad: Instance-Based Ontology Matching Using Different Kinds Of Formalisms, Proc. Int. Conference on Semantic Web Engineering, World Academy of Science, Engineering and Technology, Oslo, Norway, July 29-21, 2009
- T. Schlüter, S. Conrad: TARGEN: A Market Basket Dataset Generator for Temporal Association Rule Mining, Proc. IADIS European Conference Data Mining 2009, Algarve, Portugal, June 18-20, 2009

2008

- K. Zaiß, T. Schlüter, S. Conrad: Instance-Based Ontology Matching using Regular Expressions, Proc. On the Move to Meaningful Internet Systems: OTM 2008 Workshops, ODBase 2008, LNCS 5333, Monterrey, Mexico, November 9-14, 2008
- M. Matuschek, T. Schlüter, S. Conrad: Measuring Text Similarity With Dynamic Time Warping, Proc. 2008 Int. Symposium on Database Engineering & Applications, Coimbra, Portugal, September 10-12, 2008
- T. Schlüter: Bestimmung interessanter zeitlicher Zusammenhänge für Temporal Data Mining, Proc. 20th GI-Workshop on Foundations of Databases (Grundlagen von Datenbanken), Apolda, Germany, May 13-16, 2008

References

[AA10] L. Auret and C. Aldrich. Change Point Detection in Time Series Data with Random Forests. *Control Engineering Practice*, 2010.

[ABKS99] Mihael Ankerst, Markus M. Breunig, Hans-Peter Kriegel, and Jörg Sander. Optics: Ordering points to identify the clustering structure. In Alex Delis, Christos Faloutsos, and Shahram Ghandeharizadeh, editors, *SIGMOD Conf.*, pages 49–60. ACM Press, 1999.

[AFS93] R. Agrawal, C. Faloutsos, and A. Swami. Efficient Similarity Search In Sequence Databases. In D.B. Lomet, editor, *Proc. 4th Int. Conf. on Foundations of Data Organization and Algorithms (FODO'93)*, pages 69–84. Springer Verlag, 1993.

[AGP$^+$05] Peter Anderer, Georg Gruber, Silvia Parapatics, Michael Woertz, Tatiana Miazhynskaia, Gerhard Klosch, Bernd Saletu, Josef Zeitlhofer, Manuel J Barbanoj, Heidi Danker-Hopfe, Sari-Leena Himanen, Bob Kemp, Thomas Penzel, Michael Grozinger, Dieter Kunz, Peter Rappelsberger, Alois Schlogl, and Georg Dorffner. An E-health solution for automatic sleep classification according to Rechtschaffen and Kales: validation study of the Somnolyzer 24 x 7 utilizing the Siesta database. *Neuropsychobiology*, 51(3):115–33, 2005.

[AIS93] R. Agrawal, T. Imielinski, and A. N. Swami. Mining association rules between sets of items in large databases. In *Proc. ACM SIGMOD Int. Conf. Management of Data*, pages 207–216, Washington, D.C., 1993.

[AKK$^+$06] J. Aßfalg, H. Kriegel, P. Kröger, P. Kunath, A. Pryakhin, and M. Renz. Similarity Search on Time Series Based on Threshold Queries. In Y. E. Ioannidis, M. H. Scholl, J. W. Schmidt, F. Matthes, M. Hatzopoulos, K. Böhm, A. Kemper, T. Grust, and C. Böhm, editors, *Advances in Database Technology – 2006, 10th Int. Conf. on Extending Database Technology*, LNCS 3896, pages 276–294. Springer, 2006.

[All83] James F. Allen. Maintaining knowledge about temporal intervals. *Commun. ACM*, 26:832–843, 1983.

[ALSS95] R. Agrawal, K.-I. Lin, H. S. Sawhney, and K. Shim. Fast Similarity Search in the Presence of Noise, Scaling, and Translation in Time-Series Databases. In U. Dayal, P. M. D. Gray, and S. Nishio, editors, *Proc. 21st Int. Conf. on Very Large Data Bases (VLDB)*, pages 490–501. Morgan Kaufmann, 1995.

[ANW67] J. Ahlberg, Edwin Nilson, and J. Walsh. *The theory of splines and their applications.* Academic Press, New York, 1967.

[AO01] C. M. Antunes and A. L. Oliveira. Temporal data mining: An overview. *KDD Workshop on Temporal Data Mining*, 2001.

[APWZ95] Rakesh Agrawal, Giuseppe Psaila, Edward L. Wimmers, and Mohamed Zaot. Querying shapes of histories. In *Proc. 21st Int. Conf. on Very Large Databases (VLDB)*, pages 502–514, Zurich, Switzerland, 1995.

[AR00] J. M. Ale and G. H. Rossi. An approach to discovering temporal association rules. In *Proc. ACM Symp. on Applied Computing (SAC), Como, Italy*, pages 294–300, 2000.

[AS94] Rakesh Agrawal and Ramakrishnan Srikant. Fast algorithms for mining association rules. In *Proc. 20th Int. Conf. Very Large Data Bases*, pages 487–499. Morgan Kaufmann, 1994.

[AS95] R. Agrawal and R. Srikant. Mining sequential patterns. In *ICDE*. IEEE Computer Society Press, 1995.

[AWK$^+$09] I. Assent, M. Wichterich, R. Krieger, H. Kremer, and T. Seidl. Anticipatory DTW for Efficient Similarity Search in Time Series Databases. *Proc. VLDB Endowment*, 2(1):826–837, 2009.

[BC94] D. J. Berndt and J. Clifford. Using Dynamic Time Warping to Find Patterns in Time Series. In U.M. Fayyad and R. Uthurusamy, editors, *Knowledge Discovery in Databases: Papers from the 1994 AAAI Workshop*, pages 359–370. AAAI Press, 1994.

[BCP$^+$93] Pierre Baldi, Yves Chauvin, Cathy Place, Tim Hunkapiller, and Marcella A. Mcclure. Hidden markov models in molecular biology: New algorithms and applications. In *In: Advances in Neural Information Processing Systems 5*, pages 747–754. Morgan Kaufmann, 1993.

[BK59] R. Bellman and R. Kalaba. On Adaptive Control Processes. *IRE Transactions on Automatic Control*, 4(2):1–9, 1959.

[BK07] Igor Beliaev and Robert Kozma. Time series prediction using chaotic neural networks on the cats benchmark. *Neurocomput.*, 70(13-15):2426–2439, 2007.

[BKSS90] N. Beckmann, H.-P. Kriegel, R. Schneider, and B. Seeger. The R*-tree: an Efficient and Robust Access Method for Points and Rectangles. In H. Garcia-Molina and H.V. Jagadish, editors, *Proc. ACM SIGMOD Int. Conf. on Management of Data*, pages 322–331, 1990.

[BLT$^+$10] A. Bertone, T. Lammarsch, T. Turic, W. Aigner, S. Miksch, and J. Gärtner. MuTiny: A Multi-Time Interval Pattern Discovery Approach To Preserve The Temporal Information In Between. In *IADIS European Conf. on Data Mining*, pages 101–106. IADIS Press, 2010.

[BNDD02] A. Boggess, F. J. Narcowich, D. L. Donoho, and P. L. Donoho. A first course in wavelets with fourier analysis. *Physics Today*, 55(5):63–64, 2002.

[Bod05] Ferenc Bodon. A trie-based apriori implementation for mining frequent item sequences. In *Proc. of ACM SIGKDD International Workshop on Open Source Data Mining (OSDM)*, Chicago, IL, USA, 2005.

[Bor86] A. Borbely. *Secrets of Sleep.* Basic Books, New York, 1986.

[Bre96] Leo Breiman. Bagging predictors. *Machine Learning*, 24:123–140, 1996.

[Bre01] Leo Breiman. Random forests. *Mach. Learn.*, 45:5–32, October 2001.

[BST09] Krisztian Buza and Lars Schmidt-Thieme. Motif-based Classification of Time Series with Bayesian Networks and SVMs. In *Proc. of 32nd Annual Conf. of the Gesellschaft für Klassifikation (GfKl)*, 2009.

[BSVW99] T. Brijs, G. Swinnen, K. Vanhoof, and G. Wets. Using association rules for product assortment decisions: A case study. In *Knowledge Discovery and Data Mining*, pages 254–260, 1999.

[BTPR08] L. N. Boyle, J. Tippin, A. Paul, and M. Rizzo. Driver performance in the moments surrounding a microsleep. *Transportation Research*, 11(2):126 – 136, 2008.

[BYRN99] R. Baeza-Yates and B. Ribeiro-Neto. *Modern Information Retrieval.* Pearson / Addison Wesley, 1999.

[CA11] N. Castro and P. Azevedo. Time Series Motifs Statistical Significance. In *Proc. SIAM Int. Conf. on Data Mining*, pages 687–698. SIAM, 2011.

[CC08] Jonathan D. Cryer and Kung-Sik Chan. *Time Series Analysis With Applications in R.* Springer, New York, 2008.

[CD97] Surajit Chaudhuri and Umeshwar Dayal. An overview of data warehousing and olap technology. *SIGMOD Rec.*, 26:65–74, March 1997.

[CD07] H. Chen and P. Dyke. An example of non-linear time series flood modelling using the system identification method. *Second Int. Conf. on Flood Risk Assessment*, 2007.

[CDK$^+$99] G. Cimino, G. Del Duce, L. K. Kadonaga, G. Rotundo, A. Sisani, G. Stabile, B. Tirozzi, and M. Whiticar. Time series analysis of geological data. *Chemical Geology*, 161(1-3):253 – 270, 1999.

[CF99] K.-P. Chan and A. W.-C. Fu. Efficient Time Series Matching by Wavelets. In *Proc. 15th Int. Conf. on Data Engineering (ICDE)*, pages 126–133. IEEE Computer Society, 1999.

[CG05] G. Corani and G. Guariso. Coupling fuzzy modelling and neural networks for river flood prediction. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, 2005.

[CGKL11] Suan Khai Chong, Mohamed Medhat Gaber, Shonali Krishnaswamy, and Seng Wai Loke. Energy conservation in wireless sensor networks: a rule-based approach. *Knowl. Inf. Syst.*, 28(3):579–614, 2011.

[CGL04] F. Coenen, G. Goulbourne, and P. Leng. Tree structures for mining association rules. *Data Min. Knowl. Discov.*, 8:25–51, 2004.

[CHS$^+$00] P. De Chazal, C. Heneghan, E. Sheridan, R. Reilly, and P. Nolan. Automatic classification of sleep apnea epochs using the electrocardiogram. *Comput. Cardiol.*, 2000.

[Cod70] E. F. Codd. A relational model of data for large shared data banks. *Commun. ACM*, 13:377–387, June 1970.

[CWY$^+$11] Chung-Wen Cho, Yi-Hung Wu, Show-Jane Yen, Ying Zheng, and Arbee L. Chen. On-line rule matching for event prediction. *VLDB Journal*, 20:303–334, June 2011.

[DFT03] C. S. Daw, C. E. A. Finney, and E. R. Tracy. A review of symbolic analysis of experimental data. *Review of Scientific Instruments*, 74(2):915–930, 2003.

[DGM97] G. Das, D. Gunopulos, and H. Mannila. Finding Similar Time Series. In H. J. Komorowski and J. M. Zytkow, editors, *Principles of Data Mining and Knowledge Discovery, First European Symp. (PKDD)*, pages 88–100. Springer-Verlag, 1997.

[DHS01] Richard O. Duda, Peter E. Hart, and David G. Stork. *Pattern Classification (2nd Edition).* Wiley-Interscience, 2 edition, November 2001.

[Die00] Thomas G. Dietterich. An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization. *Mach. Learn.*, 40:139–157, August 2000.

[Dir12] Dirk Eddelbuettel. Beancounter portfolio performance toolkit homepage. Online: `http://dirk.eddelbuettel.com/code/beancounter.html`, last visited January, 2012.

[dlH05] C. de la Higuera. A Bibliographical Study of Grammatical Inference. *Pattern Recognition*, 38(9):1332–1348, 2005.

[DLM$^+$98] G. Das, K. Lin, H. Mannila, G. Renganathan, and P. Smyth. Rule discovery from time series. In *Knowledge Discovery and Data Mining*, pages 16–22, 1998.

[DTS$^+$08] Hui Ding, Goce Trajcevski, Peter Scheuermann, Xiaoyue Wang, and Eamonn Keogh. Querying and mining of time series data: experimental comparison of representations and distance measures. *Proc. VLDB Endow.*, 1, 2008.

[Eam12] Eamonn Keogh. SAX homepage. Online: `http://www.cs.ucr.edu/~eamonn/SAX.htm`, last visited January, 2012.

[EB10] Michael Eckert and François Bry. Rule-based composite event queries: the language xchange^{eq} and its semantics. *Knowl. Inf. Syst.*, 25(3):551–573, 2010.

[EKSX96] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *KDD*, pages 226–231, 1996.

[FA05] Pedro Gabriel Ferreira and Paulo J. Azevedo. Protein sequence classification through relevant sequence mining and bayes classifiers. In *Portuguese Conf. on Artificial Intelligence*, pages 236–247, 2005.

[Fam70] E. F. Fama. Efficient Capital Markets: A Review of Theory and Empirical Work. *Journal of Finance*, 25(2):383–417, 1970.

[FASB06] Pedro G. Ferreira, Paulo J. Azevedo, Cândida G. Silva, and Rui M. M. Brito. Mining approximate motifs in time series. In *Proc. of the 9th Int. Conf. on Discovery Science*, pages 7–10, 2006.

[FCLN07] Tak-chung Fu, Fu-lai Chung, Robert Luk, and Chak-man Ng. Stock time series pattern matching: Template-based vs. rule-based approaches. *Eng. Appl. Artif. Intell.*, 20:347–364, 2007.

[FDH01] R.J. Frank, N. Davey, and S.P. Hunt. Time Series Prediction and Neural Networks. *Journal of Intelligent and Robotic Systems*, 31(1-3):91–103, 2001.

[FGD05] Arthur Flexer, Georg Gruber, and Georg Dorffner. A reliable probabilistic sleep stager based on a single eeg signal. *Artif. Intell. Med.*, 33(3):199–207, 2005.

[Fis87] D. H. Fisher. Knowledge acquisition via incremental conceptual clustering. *Machine Learning*, 1987.

[FJMM97] C. Faloutsos, H. Jagadish, A. Mendelzon, and T. Milo. A Signature Technique for Similarity-Based Queries. In *Proc. Int. Conf. on Compression and Complexity of Sequences*, pages 2–20. IEEE Computer Society, 1997.

[For73] G. D. Fornay. The Viterbi Algorithm. *Proc. IEEE*, 61(3):268–278, 1973.

[FPSS96] Usama Fayyad, Gregory Piatetsky-Shapiro, and Padhraic Smyth. Knowledge discovery and data mining: Towards a unifying framework. In *KDD*, pages 82–88. AAAI Press, 1996.

[Fre12] Freiburg Epilepsy Seizure Prediction EEG Database. Homepage. Online: `https://epilepsy.uni-freiburg.de/freiburg-seizure-prediction-project/eeg-database`, last visited January, 2012.

[FRM94] C. Faloutsos, M. Ranganathan, and Y. Manolopoulos. Fast Subsequence Matching in Time-Series Databases. In R. T. Snodgrass and M. Winslett, editors, *Proc. ACM SIGMOD Int. Conf. on Management of Data*, pages 419–429, 1994.

[FS97] Yoav Freund and Robert E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *J. Comput. Syst. Sci.*, 55:119–139, August 1997.

[FYIT58] B. Fujimori, T. Yokota, Y. Ishibashi, and T. Takei. Analysis of the electroencephalogram of children by histogram method. *Electroencephalography and Clinical Neurophysiology*, 10:241–252, 1958.

[GAG$^+$00] A. Goldberger, L. Amaral, L. Glass, J. Hausdorff, P. Ivanov, R. Mark, J. Mietus, G. Moody, C.-K. Peng, and H. Stanley. PhysioBank, PhysioToolkit, and PhysioNet: Components of a new research resource for complex physiologic signals. *Circulation*, 101(23):e215–e220, 2000.

[GAIM00] M. Gavrilov, D. Anguelov, P. Indyk, and R. Motwani. Mining the Stock Market: Which Measure is Best. In *Proc. 6th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining (KDD)*, pages 487–496, 2000.

[GCL00] G. Goulbourne, F. Coenen, and P. H. Leng. Algorithms for computing association rules using a partial-support tree. *Knowledge Based Systems*, 13(2-3):141–149, 2000.

[GLT01] C. Lee Giles, Steve Lawrence, and A. C. Tsoi. Noisy Time Series Prediction using a Recurrent Neural Network and Grammatical Inference. *Machine Learning*, 44(1/2):161–183, 2001.

[GNTA10] T. F. Gharib, H. Nassar, M. Taha, and A. Abraham. An efficient algorithm for incremental mining of temporal association rules. *Data & Knowledge Engineering*, 69(8):800 – 815, 2010.

[GRS98] Sudipto Guha, Rajeev Rastogi, and Kyuseok Shim. CURE: an efficient clustering algorithm for large databases. *SIGMOD Rec.*, 27(2):73–84, 1998.

[GS99] V. Guralnik and J. Srivastava. Event Detection from Time Series Data. In *Proc. 5th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining (KDD)*, pages 33–42, 1999.

[GS00] Xianping Ge and Padhraic Smyth. Deformable markov model templates for time-series pattern matching. In *Knowledge Discovery and Data Mining*, pages 81–90, 2000.

[GST01] Wolfgang Gaul and Lars Schmidt-Thieme. Mining generalized association rules for sequential and path data. In *ICDM*, pages 593–596. IEEE Computer Society, 2001.

[GT11] Valerio Grossi and Franco Turini. Stream mining: a novel architecture for ensemble-based classification. *Knowl. Inf. Syst.*, pages 1–55, 2011.

[GV06] A. Gellert and L. Vintan. Person movement prediction using hidden markov models. *Studies in Informatics and Control*, 15(1):17–30, 2006.

[GZK05] M. M. Gaber, A. Zaslavsky, and S. Krishnaswamy. Mining Data Streams: a Review. *SIGMOD Records*, 34(2):18–26, 2005.

[HA04] Victoria J. Hodge and Jim Austin. A survey of outlier detection methodologies. *Artificial Intelligence Review*, 22:2004, 2004.

[Han27] Arthur Hanau. *Die Prognose der Schweinepreise.* Vierteljahrshefte zur Konjunkturforschung. Hobbing, 1927.

[Han05] Jiawei Han. *Data Mining: Concepts and Techniques.* Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2005.

[Haw80] Douglas M. Hawkins. *Identification of outliers.* Chapman and Hall, London ; New York, 1980.

[HDT02] Sherri K. Harms, Jitender S. Deogun, and Tsegaye Tadesse. Discovering sequential association rules with constraints and time lags in multiple sequences. In *Proc. 13th Int. Symp. on Foundations of Intelligent Systems*, ISMIS '02, pages 432–441, 2002.

[HDY99] Jiawei Han, Guozhu Dong, and Yiwen Yin. Efficient mining of partial periodic patterns in time series database. In *Proc. ICDE*, pages 106–115, 1999.

[HFH$^+$09] Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H. Witten. The weka data mining software: an update. *SIGKDD Explor. Newsl.*, 11:10–18, 2009.

[HGN00] Jochen Hipp, Ulrich Güntzer, and Gholamreza Nakhaeizadeh. Algorithms for association rule mining - a general survey and comparison. *SIGKDD Explor. Newsl.*, 2:58–64, 2000.

[HKY02] Masaaki Hanaoka, Masaki Kobayashi, and Haruaki Yamazaki. Automatic sleep stage scoring based on waveform recognition method and decision-tree learning. *Systems and Computers in Japan*, 33(11):1–13, 2002.

[HLN01] T. S. Hu, K. C. Lam, and S. T. Ng. River flow time series prediction with a range-dependent neural network. *Hydrol. Sci. J*, 46:729–745, 2001.

[Häm10] Wilhelmiina Hämäläinen. Statapriori: an efficient algorithm for searching statistically significant association rules. *Knowledge and Information Systems*, 23:373–399, 2010.

[HN05] Md. Rafiul Hassan and Baikunth Nath. Stockmarket forecasting using hidden markov model: A new approach. *Int. Conf. on Intelligent Systems Design and Applications*, 0:192–196, 2005.

[Ho98] Tin Kam Ho. The random subspace method for constructing decision forests. *IEEE Trans. Pattern Anal. Mach. Intell.*, pages 832–844, 1998.

[Hod11] V. Hodge. *Outlier and Anomaly Detection: A Survey of Outlier and Anomaly Detection Methods*. LAP Lambert Academic Publishing, 2011.

[Höp01] Frank Höppner. Learning temporal rules from state sequence. In *IJCAI Workshop on Learning from Temporal and Spatial Data*, 2001.

[Höp02a] Frank Höppner. Learning dependencies in multivariate time series. In *ECAI Workshop on Knowledge Discovery from Temporal- and Spatio-Temporal Data*, 2002.

[Höp02b] Frank Höppner. Time series abstraction methods - a survey. In *Informatik bewegt: Informatik 2002 - 32. Jahrestagung der Gesellschaft für Informatik e.v. (GI)*, pages 777–786. GI, 2002.

[HPM$^+$00] J. Han, J. Pei, B. Mortazavi-Asl, Q. Chen, U. Dayal, and M.-C. Hsu. Free-Span: Frequent Pattern-projected Sequential Pattern Mining. In *Proc. 6th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining (KDD)*, pages 355–359, 2000.

[HPY00] Jiawei Han, Jian Pei, and Yiwen Yin. Mining frequent patterns without candidate generation. In *Proc. ACM SIGMOD Int. Conf. on Management of data*, pages 1–12, New York, NY, USA, 2000.

[HPYM04] J. Han, J. Pei, Y. Yin, and R. Mao. Mining Frequent Patterns without Candidate Generation: A Frequent-Pattern Tree Approach. *Data Mining and Knowledge Discovery*, 8(1):53–87, 2004.

[HW79] J. A. Hartigan and M. A. Wong. A K-means clustering algorithm. *Applied Statistics*, 28:100–108, 1979.

[IM99] Y. Ichimaru and G. Moody. Development of the polysomnographic database on cd-rom. *Psychiatry and Clinical Neurosciences*, 53:175–177(3), 1999.

[IS05] Parvati Iyer and Abhipsita Singh. Document similarity analysis for a plagiarism detection system. In *IICAI*, pages 2534–2544, 2005.

[Ita75] F. Itakura. Minimum Prediction Residual Principle Applied to Speech Recognition. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 23(1):67–72, 1975.

[JD88] Anil K. Jain and Richard C. Dubes. *Algorithms for Clustering Data.* Prentice-Hall, 1988.

[JLH09] Yu-Feng Jiang, Chun-Ping Li, and Jun-Zhou Han. Stock temporal prediction based on time series motifs. In *Int. Conf. on Machine Learning and Cybernetics*, volume 6, pages 3550 –3555, 2009.

[JMF99] A. K. Jain, M. N. Murty, and P. J. Flynn. Data clustering: a review. *ACM Comput. Surv.*, 31(3):264–323, September 1999.

[Joe08] Philipp Joeres. Tree-based Database Reorganization for Increasing the Efficiency of Association Rule Mining. Bachelor thesis. Heinrich-Heine-University, Duesseldorf, Germany, 2008.

[Joh67] S. C. Johnson. Hierarchical clustering schemes. *Psychometrika*, 2:241–254, 1967.

[KAH96] Krzysztof Koperski, Junas Adhikary, and Jiawei Han. Spatial data mining: Progress and challenges - survey paper. In *SIGMOD Workshop on Research Issues on data Mining and Knowledge Discovery (DMKD*, pages 1–10, 1996.

[KBF$^+$00] R. Kohavi, C. Brodley, B. Frasca, L. Mason, and Z. Zheng. KDD-Cup 2000 Organizers' Report: Peeling the Onion. *SIGKDD Explorations*, 2(2):86–98, 2000.

[KCHP93] Eamonn Keogh, Selina Chu, David Hart, and Michael Pazzani. Segmenting time series: A survey and novel approach. In *Data mining in Time Series Databases. Published by World Scientific*, pages 1–22, 1993.

[KCHP04] E. Keogh, S. Chu, D. Hart, and M. Pazzani. Segmenting Time Series: A Survey and Novel Approach. In M. Last, A. Kandel, and H. Bunke, editors, *Data Mining In Time Series Databases*, volume 57 of *Series in Machine Perception and Artificial Intelligence*, chapter 1, pages 1–22. World Scientific Publ., 2004.

[Keo02] E. Keogh. Exact Indexing of Dynamic Time Warping. In *Proc. 28th Int. Conf. on Very Large Data Bases (VLDB)*, pages 406–417. Morgan Kaufmann, 2002.

[KF00a] P. Kam and A. W. Fu. Discovering Temporal Patterns for Interval-based Events. In *Proc. 2nd Int. Conf. on Data Warehousing and Knowledge Discovery*, LNCS 2114, pages 317–326. Springer, 2000.

[KF00b] Po-Shan Kam and Ada Fu. Discovering temporal patterns for interval-based events. In Yahiko Kambayashi, Mukesh Mohania, and A. Tjoa, editors, *Data Warehousing and Knowledge Discovery*, volume 1874 of *Lecture Notes in Computer Science*, pages 317–326. Springer Berlin / Heidelberg, 2000.

[KF02] J. Komlos and M. Flandreau. Using ARIMA Forecasts to Explore the Efficiency of the Forward Reichsmark Market. Austria-Hungary, 1876-1914. *Münchener Wirtschaftswissenschaftliche Beiträge (VWL)*, 2002.

[KGP09] Ahsan H. Khandoker, Jayavardhana Gubbi, and Marimuthu Palaniswami. Automated scoring of obstructive sleep apnea and hypopnea events using short-term electrocardiogram recordings. *Trans. Info. Tech. Biomed.*, 13(6):1057–1067, 2009.

[KK06a] S. Kotsiantis and D. Kanellopoulos. Association rules mining: A recent overview. In *Proc. GESTS Int. Transactions on Computer Science and Engineering*, volume 32(1), pages 71–82, 2006.

[KK06b] Sotiris Kotsiantis and Dimitris Kanellopoulos. Discretization techniques: A recent survey. *GESTS Int. Transactions on Computer Science and Engineering*, 32(1):47–58, 2006.

[KL05] Eamonn Keogh and Jessica Lin. Clustering of time-series subsequences is meaningless: implications for previous and future research. *Knowl. Inf. Syst.*, 8:154–177, 2005.

[KO03] Bob Kemp and Jesus Olivan. European data format plus (edf+), an edf alike standard format for the exchange of physiological data. *Clinical Neurophysiology*, 114(9):1755 – 1761, 2003.

[Koe05] P. Koehn. Europarl: A parallel corpus for statistical machine translation. In *MT Summit*, 2005.

[KP00] E. Keogh and M. Pazzani. Scaling up dynamic time warping for data mining applications. In *Knowledge Discovery and Data Mining*, 2000.

[KP01] E. Keogh and M. Pazzani. Derivative dynamic time warping, 2001.

[KP10] Y. J. Kim and J. M. Patel. Performance Comparison of the R*-Tree and the Quadtree for kNN and Distance Join Queries. *IEEE Transactions on Knowledge and Data Engineering*, 22:1014–1027, 2010.

[KS97] E. J. Keogh and P. Smyth. A Probabilistic Approach to Fast Pattern Matching in Time Series Databases. In D. Heckerman, H. Mannila, and D. Pregibon, editors, *Proc. 3th Int. Conf. on Knowledge Discovery and Data Mining (KDD)*, pages 24–30. AAAI Press, 1997.

[KSE+05] Vered Kunik, Zach Solan, Shimon Edelman, Eytan Ruppin, and David Horn. Motif extraction and protein classification. *Computational Systems Bioinformatics Conf. Int. IEEE Computer Society*, 0:80–85, 2005.

[KTV10] Ioannis Katakis, Grigorios Tsoumakas, and Ioannis Vlahavas. Tracking recurring contexts using ensemble classifiers: an application to email filtering. *Knowl. Inf. Syst.*, 22(3):371–391, March 2010.

[KVR+92] B. Kemp, A. V&auml;rri, A. C. Rosa, K. D. Nielsen, and J. Gade. A simple format for exchange of digitized polygraphic recordings. *Electroencephalogr. Clin. Neurophysiol.*, 82(5):391–3, 1992.

[KYM07] Y. Kawahara, T. Yairi, and K. Machida. Change-Point Detection in Time-Series Data Based on Subspace Identification. In *Proc. 7th IEEE Int. Conf. on Data Mining (ICDM)*, pages 559–564. IEEE Computer Society, 2007.

[Lan12] Landesamt für Natur, Umwelt und Verbraucherschutz Nordrhein-Westfalen. Homepage. Online: `http://www.lanuv.nrw.de/aktuelles/umwdat.htm`, last visited January, 2012.

[Law12] Lawrence Berkeley National Lab. The colt project - open source libraries for high performance scientific and technical computing in java. Online: `http://acs.lbl.gov/software/colt`, last visited January, 2012.

[LC01] T. Lancaster and F. Culwin. Towards an error free plagarism detection process. *SIGCSE Bull.*, 33(3):57–60, 2001.

[LC07] C. Leung and Y. Chan. A natural language processing approach to automatic plagiarism detection. In *SIGITE '07: Proc. 8th ACM SIGITE conf. on Information technology education*, pages 213–218, New York, NY, USA, 2007.

[LHF98] H. Lu, J. Han, and L. Feng. Stock Movement Prediction And N-Dimensional Inter-Transaction Association Rules. In *Proc. ACM SIGMOD Workshop on Research Issues on Data Mining and Knowledge*, 1998.

[Lin94] T. Lindeberg. *Scale-Space Theory in Computer Vision.* Kluwer Academic Publishers, 1994.

[LKLcC03] Jessica Lin, Eamonn J. Keogh, Stefano Lonardi, and Bill Yuan chi Chiu. A symbolic representation of time series, with implications for streaming algorithms. In Mohammed Javeed Zaki and Charu C. Aggarwal, editors, *DMKD*, pages 2–11. ACM, 2003.

[LKWL07] Jessica Lin, Eamonn Keogh, Li Wei, and Stefano Lonardi. Experiencing sax: a novel symbolic representation of time series. *Data Mining and Knowledge Discovery*, 15:107–144, 2007.

[LLS04] Rhain Louis, James Lee, and Richard Stephenson. Design and validation of a computer-based sleep-scoring algorithm. *J. neuroscience methods*, 133(1-2):71–80, 2004.

[LNWJ01] Y. Li, P. Ning, X. S. Wang, and S. Jajodia. Discovering calendar-based temporal association rules. In *TIME*, pages 111–118, 2001.

[LOW01] W. Lin, M. A. Orgun, and G. J. Williams. Multilevels Hidden Markov Models for Temporal Data Mining. In *KDD Workshop on Temporal Data Mining*, 2001.

[LR98] L. Lin and T. Risch. Querying Continuous Time Sequences. In A. Gupta, O. Shmueli, and J. Widom, editors, *Proc. 24rd Int. Conf. on Very Large Data Bases (VLDB)*, pages 170–181. Morgan Kaufmann, 1998.

[LS06] S. Laxman and P. S. Sastry. A survey of temporal data mining. *Sadhana*, 31(2):173–198, 2006.

[Lyo04] R. G. Lyons. *Understanding Digital Signal Processing.* Prentice Hall PTR, Upper Saddle River, NJ, USA, 2004.

[MAEL06] Florian Mormann, Ralph G. Andrzejak, Christian E. Elger, and Klaus Lehnertz. Seizure prediction: the long and winding road. *Brain*, 2006.

[MAR+04] H. Marques, J. Almeida, L. Rocha, W. Meira, P. Guerra, and V. Almeida. A characterization of broadband user behavior and their e-business activities. *SIGMETRICS Perform. Eval. Rev.*, 32(3):3–13, 2004.

[MCH$^+$10] M. O. Mendez, J. Corthout, S. Van Huffel, M. Matteucci, T. Penzel, S. Cerutti, and A. M. Bianchi. Automatic screening of obstructive sleep apnea from the ECG based on empirical mode decomposition and wavelet analysis. *Physiological Measurement*, 31(3):273, 2010.

[MF00] Jn McNames and Am Fraser. Obstructive sleep apnea classification based on spectrogram patterns in the electrocardiogram. *Comput. Cardiol.*, 2000.

[MGG$^+$09] J. Molina, J. Garcia, A. Garcia, R. Melo, and L. Correia. Segmentation and classification of time-series: Real case studies. In *Intelligent Data Engineering and Automated Learning - IDEAL*, volume 5788 of *Lecture Notes in Computer Science*, pages 743–750. Springer Berlin / Heidelberg, 2009.

[Mie05] Taneli Mielikäinenn. *Summarization Techniques for Pattern Collections in Data Mining.* PhD thesis, Faculty of Science, University of Helsinki, 2005.

[Mit06] T. M. Mitchell. *Machine Learning.* McGraw-Hill, 2006.

[MMB$^+$86] George Moody, Roger Mark, Marjorie A. Bump, Joseph S. Weinstein, Aaron D. Berman, Joseph Mietus, , and Ary Goldberger. Clinical validation of the ECG-derived respiration (EDR) technique. *Comput. Cardiol.*, 13:507–510, 1986.

[MP95] J. Malmivuo and R. Plonsey. *Bioelectromagnetism : Principles and Applications of Bioelectric and Biomagnetic Fields.* Oxford University Press, USA, July 1995.

[MPIG00] J. Mietus, C.-K. Peng, PCh Ivanov, and A. Goldberger. Detection of obstructive sleep apnea from cardiac interbeat interval time series. *Comput. Cardiol.*, 27:753–756, 2000.

[MRBD11] Amy McGovern, Derek H. Rosendahl, Rodger A. Brown, and Kelvin Droegemeier. Identifying predictive multi-dimensional time series motifs: an application to severe weather prediction. *Data Min. Knowl. Discov.*, 22(1-2):232–258, 2011.

[MSC08] Michael Matuschek, Tim Schlüter, and Stefan Conrad. Measuring text similarity with dynamic time warping. In *Proc. Int. Symp. on Database Engineering & Applications (IDEAS)*, pages 263–267, New York, NY, USA, 2008.

[MSC09] Michael Matuschek, Tim Schlüter, and Stefan Conrad. Language-independent comparison in large text corpora. Unpublished manuscript, 2009.

[MT96a] H. Mannila and H. Toivonen. Discovering Generalized Episodes Using Minimal Occurrences. In E. Simoudis, J. Han, and U. M. Fayyad, editors, *Proc. 2nd Int. Conf. on Knowledge Discovery in Databases and Data Mining*, pages 146–151. AAAI Press, 1996.

[MT96b] Heikki Mannila and Hannu Toivonen. Discovering generalized episodes using minimal occurrences. In *KDD*, pages 146–151, 1996.

[MTIV97] Heikki Mannila, Hannu Toivonen, and A. Inkeri Verkamo. Discovery of frequent episodes in event sequences. *Data Min. Knowl. Discov.*, 1, 1997.

[MTV95] H. Mannila, H. Toivonen, and A. I. Verkamo. Discovering Frequent Episodes in Sequences (Extended Abstract). In U. M. Fayyad and R. Uthurusamy, editors, *Proc. 1st Int. Conf. on Knowledge Discovery and Data Mining (KDD'95)*, pages 210–215. AAAI Press, 1995.

[MTV97] H. Mannila, H. Toivonen, and A. I. Verkamo. Discovery of Frequent Episodes in Event Sequences. *Data Mining and Knowledge Discovery*, 1(3):259–289, 1997.

[MVM09] F. P. Miller, A. F. Vandome, and J. McBrewster, editors. *Levenshtein Distance: Information theory, Computer science, String (computer science), String metric, Damerau-Levenshtein distance, Spell checker, Hamming distance*. Alphascript Publishing, 2009.

[NES$^+$11] M. Nagi, A. ElSheikh, I. Sleiman, P. Peng, M. Rifaie, K. Kianmehr, P. Karampelas, M. Ridley, J. Rokne, and R. Alhajj. Association rules mining based approach for web usage mining. In *2011 IEEE Int. Conf. on Information Reuse and Integration (IRI)*, pages 166–171, aug. 2011.

[OLC08] Asem Omari, Regina Langer, and Stefan Conrad. Advanced data mining and applications. In *Proc. 4th Int. Conf., ADMA*, pages 400–410. Springer-Verlag, Lecture Notes in Artificial Intelligence Vol. 5139, 2008.

[ORS98] B. Ozden, S. Ramaswamy, and A. Silberschatz. Cyclic association rules. In *ICDE*, pages 412–421, 1998.

[oSMTF99] American Academy of Sleep Medicine Task Force. Sleep-related breathing disorders in adults: recommendations for syndrome definition and measurement techniques in clinical research. *Sleep*, 22(5):667–89, 1999.

[OV10] C. Orsenigo and C. Vercellis. Combining Discrete SVM and Fixed Cardinality Warping Distances for Multivariate Time Series Classification. *Pattern Recognition*, 43(11):3787–3794, 2010.

[OWT$^+$00] M. Obata, K. Wada, K. Toraichi, K. Mori, and M. Ohira. An Approximation of Data Points by Piecewise Polynomial Functions and Their Dual Orthogonal Functions. *Signal Processing*, 80(3):507–514, 2000.

[Pav12] Pavel Senin. SAX - jmotif - homepage. Online: `http://code.google.com/p/jmotif/wiki/SAX`, last visited January, 2012.

[PHMA$^+$01] Jian Pei, Jiawei Han, Behzad Mortazavi-Asl, Helen Pinto, Qiming Chen, Umeshwar Dayal, and Meichun Hsu. Prefixspan: Mining sequential patterns by prefix-projected growth. In *ICDE*, pages 215–224. IEEE Computer Society, 2001.

[PKLL02] Pranav Patel, Eamonn J. Keogh, Jessica Lin, and Stefano Lonardi. Mining motifs in massive time series databases. In *ICDM*, pages 370–377. IEEE Computer Society, 2002.

[PLCS10] Sang-Ho Park, Ju-Hong Lee, Seok-Ju Chun, and Jae-Won Song. Representation and clustering of time series by means of segmentation based on pips detection. In *Computer and Automation Engineering (ICCAE), 2010 The 2nd Int. Conf. on*, volume 3, pages 17 –21, 2010.

[PMdC$^+$02] T. Penzel, J. McNames, P. de Chazal, B. Raymond, A. Murray, and G. Moody. Systematic comparison of different algorithms for apnea detection based on electrocardiogram recordings. *Medical & Biological Engineering & Computing*, 40:402–407, 2002.

[PMM$^+$00] T. Penzel, G. Moody, R. Mark, A. Goldberger, and J. Peter. The apnea-ECG database. *Comput. Cardiol.*, 27:255–258, 2000.

[POJP00] Hae Park, Jung Oh, Do Jeong, and Kwang Park. Automated sleep stage scoring using hybrid rule- and case-based reasoning. *Comput. Biomed. Res.*, 33(5):330–349, 2000.

[Pon10] Paulraj Ponniah. *Data Warehousing Fundamentals for IT Professionals: A Comprehensive Guide for IT Professionals; 2nd ed.* John Wiley & Sons Inc, Hoboken, 2010.

[Por88] A.B. Poritz. Hidden markov models: a guided tour. In *Int. Conf. on Acoustics, Speech, and Signal Processing (ICASSP-88)*, volume 1, pages 7–13, 1988.

[PRF+09] D. J. Pedregal, R. Rivas, V. Feliu, L. Sánchez, and A. Linares. A nonlinear forecasting system for the Ebro River at Zaragoza, Spain. *Environ. Model. Softw.*, 24(4):502–509, 2009.

[Qui96] J. R. Quinlan. Bagging, boosting, and c4.5. In *AAAI/IAAI, Vol. 1*, pages 725–730, 1996.

[Rab89] L. R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proc. IEEE*, 77(2):257–286, 1989.

[RCBC00] B. Raymond, R.M. Cayton, R.A. Bates, and M.J. Chappell. Screening for obstructive sleep apnea based on the electrocardiogram. *Comput. Cardiol.*, 27:263–266, 2000.

[RHS01] J. F. Roddick, K. Hornsby, and M. Spiliopoulou. Yabtsstdmr - yet another bibliography of temporal, spatial and spatio-temporal data mining research. In *SIGKDD Temporal Data Mining Workshop*, pages 167–175, San Francisco, CA, 2001.

[RJ86] L. Rabiner and B. Juang. An introduction to hidden markov models. *ASSP Magazine, IEEE*, 3(1):4–16, 1986.

[RJ11] Piotr Rzepakowski and Szymon Jaroszewicz. Decision trees for uplift modeling with single and multiple treatments. *Knowl. Inf. Syst.*, pages 1–25, 2011.

[RK68] A. Rechtschaffen and A. Kales. *A Manual Of Standardized Terminology, Techniques And Scoring System For Sleep Stages Of Human Subjects.* Washington DC, 1968.

[RK04] C. Ratanamahatana and E. Keogh. Everything you know about dynamic time warping is wrong. In *Third Workshop on Mining Temporal and Sequential Data, in conjunction with the Tenth ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining (KDD)*, 2004.

[RM03] T. Rath and R. Manmatha. Word image matching using dynamic time warping. In *IEEE Computer Society Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2003.

[RSV07] Stephane Robin, Sophie Schbath, and Vincent Vandewalle. Statistical tests to compare motif count exceptionalities. *BMC Bioinformatics*, 8(1):84, 2007.

[Rym92] R. Rymon. Search through systematic set enumeration. Technical report, Department of Computer and Information Science, University of Pennsylvania, 1992.

[SA96] Ramakrishnan Srikant and Rakesh Agrawal. Mining sequential patterns: Generalizations and performance improvements. In *Proc. Int. Conf. on Extending Database Technology (EDBT)*, volume 1057, pages 3–17. Springer, 1996.

[Sam90] Hanan Samet. *The design and analysis of spatial data structures.* Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1990.

[SC78] H. Sakoe and S. Chiba. Dynamic programming algorithm optimization for spoken word recognition. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 26:43–49, 1978.

[SC90] H. Sakoe and S. Chiba. Dynamic programming algorithm optimization for spoken word recognition. In *Readings in speech recognition*, 1990.

[SC04] R. S. Salvador and P. Chan. FastDTW: Toward Accurate Dynamic Time Warping in Linear Time and Space. In *KDD Workshop on Mining Temporal and Sequential Data*, pages 70–80, 2004.

[SC07] S. Salvador and P. Chan. FastDTW: toward accurate dynamic time warping in linear time and space. *Intell. Data Anal.*, 11(5), 2007.

[SC09] Tim Schlüter and Stefan Conrad. TARGEN: A Market Basket Dataset Generator for Temporal Association Rule Mining. In *Proceeings of the IADIS European Conf. Data Mining*, pages 133–138, 2009.

[SC10a] Tim Schlüter and Stefan Conrad. An approach for automatic sleep stage scoring and apnea-hypopnea detection. *Proc. IEEE Int. Conf. on Data Mining*, pages 1007–1012, 2010.

[SC10b] Tim Schlüter and Stefan Conrad. Mining several kinds of temporal association rules enhanced by tree structures. *Proc. Int. Conf. on Information, Process, and Knowledge Management*, pages 86–93, 2010.

[SC10c] Tim Schlüter and Stefan Conrad. TEMPUS: A Prototype System for Time Series Analysis and Prediction. In *Proc. IADIS European Conf. on Data Mining*, pages 11–18. IADIS Press, 2010.

[SC11a] Tim Schlüter and Stefan Conrad. About the analysis of time series with temporal association rule mining. In *Proc. IEEE Symp. Series in Computational Intelligence (SSCI)*, 2011.

[SC11b] Tim Schlüter and Stefan Conrad. Applying several kinds of temporal association rules to time series analysis. In *Proc. IADIS European Conf. Data Mining*, 2011.

[SC12a] Tim Schlüter and Stefan Conrad. An approach for automatic sleep stage scoring and apnea-hypopnea detection. *Frontiers of Computer Science*, 2012.

[SC12b] Tim Schlüter and Stefan Conrad. Hidden markov model-based time series prediction using motifs for detecting inter-time-serial correlations. In *Proc. ACM Symp. on Applied Computing (SAC), Riva del Garda (Trento), Italy, March 26–30*, 2012.

[Sch97] Philip A. Schrodt. Early warning of conflict in southern lebanon using hidden markov models. In *The Understanding and Management of Global Violence*, pages 131–162. St. Martin's Press, 1997.

[Sch08] Tim Schlüter. Bestimmung interessanter zeitlicher Zusammenhänge für Temporal Data Mining. In *Proc. 20th GI-Workshop on Foundations of Databases (Grundlagen von Datenbanken)*, pages 46–50, 2008.

[SG92] Padhraic Smyth and Rodney M. Goodman. An information theoretic approach to rule induction from databases. *IEEE Trans. Knowl. Data Eng.*, 4(4):301–316, 1992.

[SK71] Jack R. Smith and Ismet Karacan. EEG sleep stage scoring by an automatic hybrid system. *Electroencephalography and Clinical Neurophysiology*, 31(3):231 – 237, 1971.

[SK08] Jin Shieh and Eamonn J. Keogh. *i*sax: indexing and mining terabyte sized time series. In Ying Li, Bing Liu, and Sunita Sarawagi, editors, *KDD*, pages 623–631. ACM, 2008.

[SKC10] Tim Schlüter, Timm Kißels, and Stefan Conrad. AS3: A Framework for Automatic Sleep Stage Scoring. In *Proc. IADIS European Conf. Data Mining*, pages 83–91, 2010.

[SM00] Y. Sakakibara and H. Muramatsu. Learning Context-Free Grammars from Partially Structured Examples. In A.L. Oliveira, editor, *Proc. 5th Int. Colloquium on Grammatical Inference: Algorithms and Applications (ICGI'00)*, LNCS 1891, pages 229–240. Springer, 2000.

[Smy99] P. Smyth. Probabilistic Model-Based Clustering of Multivariate and Sequential Data. In D. Heckerman and J. Whittaker, editors, *In Proc. 7th Int. Workshop on Artificial Intelligence and Statistics*, pages 299–304. Morgan Kaufmann, 1999.

[SS06] Robert H. Shumway and David S. Stoffer. *Time Series Analysis and Its Applications With R Examples.* Springer, 2006.

[Sun07] Shiliang Sun. Ensemble learning methods for classifying eeg signals. In Michal Haindl, Josef Kittler, and Fabio Roli, editors, *Multiple Classifier Systems*, volume 4472 of *Lecture Notes in Computer Science*, pages 113–120. Springer Berlin / Heidelberg, 2007.

[SW08] Min Song and Yi-Fang Brook Wu. *Handbook of Research on Text and Web Mining Technologies.* Information Science Reference - Imprint of: IGI Publishing, Hershey, PA, 2008.

[SWY75] G. Salton, A. Wong, and C. S. Yang. A vector space model for automatic indexing. *Commun. ACM*, 18(11):613–620, 1975.

[Tas96] Task Force European Society of Cardiology / North American Society of Pacing Electrophysiology. Heart rate variability: Standards of measurement, physiological interpretation and clinical use. *Circulation*, 93:1043–1065, 1996.

[TNI04] J. Tomita, H. Nakawatase, and M. Ishii. Calculating similarity between texts using graph-based text representation model. In *CIKM '04: Proc. thirteenth ACM int. conf. on Information and knowledge management*, pages 248–249, New York, NY, USA, 2004.

[TSK05] Pang-Ning Tan, Michael Steinbach, and Vipin Kumar. *Introduction to Data Mining.* Addison Wesley, 1 edition, 2005.

[VKM02] U. Vallamsetty, K. Kant, and P. Mohapatra. Characterization of e-commerce traffic. *Electronic Commerce Research*, 3:2003, 2002.

[VVV05] Keshri Verma, Om Prakash Vyas, and Ranjana Vyas. Temporal approach to association rule mining using t-tree and p-tree. In *MLDM*, volume 3587 of *LNCS*, pages 651–659. Springer, 2005.

[Web07] Geoffrey I. Webb. Discovering significant patterns. *Mach. Learn.*, 68:1–33, 2007.

[Wei87] C. Z. Wei. Adaptive Prediction by Least Squares Predictors in Stochastic Regression Models with Applications to Time Series. *The Annals of Statistics*, 15(4):1667–1682, 1987.

[WG94] Andreas S. Weigend and Neil A. Gerschenfeld. *Time Series Prediction: Forecasting the Future and Understanding the Past.* Addison-Wesley, 1994.

[WGP$^+$05] M. Woertz, G. Gruber, S. Parapatics, P. Anderer, T. Miazhynskaia, R. Rosipal, B. Saletu, and G. Dorffner. Automatic sleep apnea detection: analysis of apnea distribution with respect to sleep stages depending on the severity of sleep apnea. *WASM*, 2005.

[WJ04] D.. White and M. Joy. Sentence-based natural language plagiarism detection. *J. Educ. Resour. Comput.*, 4(4), 2004.

[WK06] L. Wei and E. Keogh. Semi-supervised Time Series Classification. In *Proc. 12th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining (KDD)*, pages 748–753. ACM Press, 2006.

[WN06] Kittipong Warasup and Chakarida Nukoolkit. Discovery association rules in time series data. *KMUTT RESEARCH AND DEVELOPMENT JOURNAL*, 29, 2006.

[WR07] Edi Winarko and John F. Roddick. Armada - an algorithm for discovering richer relative temporal association rules from interval-based data. *Data Knowl. Eng.*, 63(1):76–90, 2007.

[XKWM07] X. Xi, E. Keogh, L. Wei, and A. Mafra-Neto. Finding motifs in a database of shapes. In *SDM*, 2007.

[XW05] Rui Xu and Donald Wunsch. Survey of clustering algorithms. *IEEE Transactions on Neural Networks*, 16(3):645–678, 2005.

[XWWL08] T. Xu, J. Wu, Z. Wu, and Q. Li. Long-term sunspot number prediction based on EMD analysis and AR model. *Chin. J. Astron. Astrophys.*, 8:337–342, 2008.

[Yah12] Yahoo!Finance. Homepage. Online: `http://finance.yahoo.com`, last visited January, 2012.

[YJF98] B.-K. Yi, H. V. Jagadish, and C. Faloutsos. Efficient Retrieval of Similar Time Sequences Under Time Warping. In *Proc. 14th Int. Conf. on Data Engineering (ICDE)*, pages 201–208. IEEE Computer Society, 1998.

[YKM+07] Dragomir Yankov, Eamonn J. Keogh, Jose Medina, Bill Yuan chi Chiu, and Victor B. Zordan. Detecting time series motifs under uniform scaling. In *KDD*, pages 844–853. ACM, 2007.

[YPD+93] T. Young, M. Palta, J. Dempsey, J. Skatrud, S. Weber, and S. Badr. The occurrence of sleep-disordered breathing among middle-aged adults. *New Englands J. Med.*, 328:1230–1235, 1993.

[ZFMP06] Manuel Zini, Marco Fabbri, Massimo Moneglia, and Alessandro Panunzi. Plagiarism detection through multilevel text comparison. In *AXMEDIS*, pages 181–185. IEEE Computer Society, 2006.

[ZKM01] Z. Zheng, R. Kohavi, and L. Mason. Real World Performance of Association Rule Algorithms. In *Proc. 7th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining (KDD)*, pages 401–406, 2001.

[ZL10] Zhi-Hua Zhou and Ming Li. Semi-supervised learning by disagreement. *Knowl. Inf. Syst.*, 24(3):415–439, 2010.

[ZSAS02] Geraldo Zimbrao, Jano Moreira De Souza, Victor Teixeira De Almeida, and Wanderson Araújo Da Silva. An algorithm to discover calendar-based temporal association rules with item's lifespan restriction. In *SIGKDD Temporal Data Mining Workshop*, 2002.

[ZvEWJ04] C.W. Zywietz, V. von Einem, B. Widiger, and G. Joseph. ECG Analysis for Sleep Apnea Detection. *Methods Inf. Med.*, 43:56–59, 2004.

List of Figures

List of Tables