TEMPORAL DATA MINING

Pranshu Sharma , Satish Anand Arjunan

***Abstract*- One of the main unresolved problems that arise during the data mining process is treating data that contains temporal information. In this case, a complete understanding of the entire phenomenon requires that the data should be viewed as a sequence of events. Temporal sequences appear in a vast range of domains, from engineering, to medicine and finance, and the ability to model and extract information from them is crucial for the advance of the information society. This paper provides a survey on the most significant techniques developed in the past ten years to deal with temporal sequences.**

I. INTRODUCTION

With the rapid increase of stored data, the interest in the discovery of hidden information has exploded in the last decade. This discovery has mainly been focused on data classification, data clustering and relationship finding. One important problem that arises during the discovery process is treating data with temporal dependencies. The attributes related with the temporal information present in this type of datasets need to be treated differently from other kinds of attributes. However, most of the data mining techniques tend to treat temporal data as an unordered collection of events, ignoring its temporal information.

The aim of this paper is to present an overview of the techniques proposed to date that deal specifically with temporal data mining. Our objective is not only to enumerate the techniques proposed so far, but also to classify and organize them in a way that may be of help for a practitioner looking for solutions to a concrete problem. Other overviews of this area have appeared in the literature ([25], [56]), but is not widely available, while the second one uses a significantly different approach to classify the existing work and is considerably less comprehensive than the present work.

The paper is organized as follows: Section 2 defines the main goals of temporal data mining, related problems and concepts and its applications areas. Section 3 presents the principal approaches to represent temporal sequences, and Section 4 the Lecture Notes in Computer Science 2 methods developed to measure the similarity between two sequences. Section 5 addresses the problems of non-supervised sequence classification and relation finding, including frequent event detection and prediction. Section 6 presents the conclusions and points some possible directions for future research.

II. MINING TEMPORAL SEQUENCES

One possible definition of data mining is “the nontrivial extraction of implicit, previously unknown and potential useful information from data” [19]. The ultimate goal of temporal data mining is to discover hidden relations between sequences and subsequences of events. The discovery of relations between sequences of events involves mainly three steps: the representation and modeling of the data sequence in a suitable form; the definition of similarity measures between sequences; and the application of models and representations to the actual mining problems. Other authors have used a different approach to classify data mining problems and algorithms.

Roddick [56] has used three dimensions: data type, mining operations and type of timing information. Although both approaches are equally valid, we preferred to use representation, similarity and operations, since it provided a more comprehensive and novel view of the field.

Depending on the nature of the event sequence, the approaches to solve the problem may be quite different. A sequence composed by a series of nominal symbols from a particular alphabet is usually called a *temporal sequence* and a sequence of continuous, real-valued elements, is known as a *time series*.

Time series or, more generally, temporal sequences, appear naturally in a variety of different domains, from engineering to scientific research, finance and medicine.

In engineering matters, they usually arise with either sensor-based monitoring, such as telecommunications control (e.g. [14], [15]) or log-based systems monitoring (see, for instances [44], [47]). In scientific research they appear, for example, in

spatial missions (e.g. [34], [50]) or in the genetics domain (e.g. [15]).
In finance, applications on the analysis of product sales or inventory consumptions are of great importance to business planning (see for instance [5]). Another very common application in finance is the prediction of the evolution of financial data (e.g. [3], [16], [20], [11], [7], [21], [17], [55]).
In healthcare, temporal sequences are a reality for decades; with data originated by complex data acquisition systems like ECG's (see, for instances [13], [21], [24], [57]), or even with simple ones like measuring the patient temperature or treatments effectiveness ([9], [6], [12]). In the last years, with the development of medical informatics, the amount of data has increased considerably, and more than ever, the need to react in real-time to any change in the patient behavior is crucial.
In general, applications that deal with temporal sequences serve mainly to support diagnosis and to predict future behaviors ([56]).

III. REPRESENTATION OF TEMPORAL SEQUENCES

A fundamental problem that needs to be solved in the representation of the data and the pre-processing that needs to be applied before actual data mining operations take place. The representation problem is especially important when dealing with time series, since direct manipulation of continuous, high-dimensional data in an efficient way is extremely difficult. This problem can be addressed in several different ways.
One possible solution is to use the data with only minimal transformation, either keeping it in its original form or using windowing and piecewise linear approximations to obtain manageable sub-sequences. These methods are described in section 3.1. A second possibility is to use a transformation that maps the data to a more manageable space, either continuous (section 3.2) or discrete (section 3.3). A more sophisticated approach, discussed in section 3.4 is to use a generative model, where a statistical or deterministic model is obtained from the data and can be used to answer more complex questions. Approaches, that deal with arbitrary dimension transaction databases are presented in section 3.5.
One issue that is relevant for all the representation methods addressed is the ability to discover and represent the subsequences of a sequence. A method commonly used to find subsequences is to use a sliding window of size *w* and place it at every possible position of the sequence. Each such window defines a new subsequence, composed by the elements inside the window [16]
.

3.1 Time-Domain Continuous Representations

A simple approach to represent a sequence of real-valued elements (*time series*) is using the initial elements, ordered by their instant of occurrence without any preprocessing ([3], [41]).
An alternative consists in finding a piecewise linear function able to approximately describe the entire initial sequence. In this way, the initial sequence is partitioned in several segments and each segment is represented by a linear function. In order to partition the sequence there exist two main possibilities: define beforehand the number of segments or discover that number and then identify the correspondent segments ([14], [27]). The objective is to obtain a representation amenable to the detection of significant changes in the sequence.
While this is a relatively straightforward representation, not much is gained in terms of our ability to manipulate the generated representations. One possible application of this type of representations is on change-point detection, where one wants to locate the points where a significant change in behavior takes place.
Another proposal, based on the idea that the human visual system partitions smooth curves into linear segments, has been proposed. It mainly consists on segmenting a sequence by iteratively merging two similar segments. Choosing which

segments are to be merged is done based on the squared error minimization criteria [34]. An extension to this model consists in associating with each segment a weight value, which represents the segment importance relatively to the entire sequence. In

this manner, it is possible to compare sequences mainly by looking at their most important segments [35]. With this representation, a more effective similarity measure can be defined, and consequently, mining operations may be applied.

One significant advantage of these approaches is the ability to reduce the impact of noise. However, problems with amplitude differences (*scaling problems*) and the existence of gaps or other time axis distortion are not addressed easily.

3.2 Transformation Based Representations

The main idea of Transformation Based Representations is to transform the initial sequences from time to another domain, and then to use a point in this new domain to represent each original sequence.

One proposal [1] uses the *Discrete Fourier Transform* (DFT) to transform a sequence from the time domain to a point in the frequency domain. Choosing the *k* first frequencies, and then representing each sequence as a point in the *k*dimensional space, achieves this goal. The DFT has the attractive property that the amplitude of the Fourier coefficients is invariant under shifts, which allows extending the method to find similar sequences ignoring shifts [1].

Other possibilities that have been proposed represent each sequence as a point in the *n*-dimensional space, with *n* being the sequence length. This representation can be viewed as a time-domain continuous representation ([20]). However, if one maps each subsequence to a point in a new space with coordinates that are the derivatives of each original sequence point, this should be considered as a transformation based representation. In this case, the derivative at each point is determined by the difference between the point and its preceding one [20].

A more recent approach uses the *Discrete Wavelet Transform* (DWT) to translate each sequence from the time domain into the time/frequency domain [11]. The DWT is a linear transformation, which decomposes the original sequence into different frequency components, without loosing the information about the instant of the elements occurrence. The sequence is then represented by its features, expressed as the wavelet coefficients. Again, only a few coefficients are needed to approximately represent the sequence.

With these kinds of representations, time series became a more manageable object, which permit the definition of efficient similarity measures and an easier application to common data mining operations

.

3.3 Discretization Based Methods

A third approach to deal with time series is the translation of the initial sequence (with real-valued elements) to a discretized sequence, this time composed of symbols from an alphabet. One language to describe the symbols of an alphabet and the relationship between two sequences was proposed in [4]. This *Shape Definition Language* (SDL) is also able to describe shape queries to pose to a sequence database and it allows performing *'blurry'* matching. A blurry match is one where the important

thing is the overall shape, therefore ignoring the specific details of a particular sequence.

The first step in the representation process is defining the alphabet of symbols and then translating the initial sequence to a sequence of symbols. The translation is done by considering transitions from an instant to the following one, and then assigning a symbol of the described alphabet to each transition.

Constraint-Based Pattern Specification Language (CAPSUL) [56] is another language

used to specify patterns, which similarly to SDL, describes patterns by considering some abstractions of the values at time intervals. The key difference between these two languages is that CAPSUL uses a set of expressive constraints upon temporal objects, which allows describing more complex patterns, for instance, periodic patterns. Finally, to perform each of those abstractions the system accesses the knowledge bases, where the relationships between the different values are defOintehde.r approach to discretize time series [33] suggests the segmentation of a sequence by computing the change ratio from one time point to the following one, and representing all consecutive points with equal change ratios by a unique segment.
After this partition, each segment is represented by a symbol, and the sequence is represented as a string of symbols. A particular case is when change ratios are classified into one of two classes: large fluctuations (symbol 1) or small fluctuations (symbol 0). This way, each time series is converted into a binary sequence, perfectly suited to be manipulated by genetic algorithms [63].
A different approach to convert a sequence into a discrete representation is using *clustering*. First, the sequence originates a set of subsequences with length *w*, by sliding a window of width *w*. Then, the set of all subsequences is clustered, originating *k* clusters, and a different symbol is associated with each cluster. The discretized version of the initial sequence is obtained by substituting each subsequence by the symbol associated to the cluster that it belongs to [15].
Another method to convert time series into a sequence of symbols is based on the use of *self-organizing maps* ([23], [24]). This conversion consists on three steps: first, a new series composed by the differences between consecutive values of the original time series is derived; second windows with size *d* (called *delay embedding*) are inputted to the SOM, which finally outputs the winner node. Each node is associated with a symbol, which means that the resultant sequence may be viewed as a sequence of symbols.
The advantage of these methods is that the time series is partitioned in a natural way, depending on its values. However, the symbols of the alphabet are usually chosen externally, which means that they are imposed by the user, who has to know the most suitable symbols, or are established in an artificial way. The discretized time series is more amenable to manipulation and processing than the original, continuous valued, time series.

3.4 Generative Models

A significantly different approach consists in obtaining a model that can be viewed as a generator for the sequences obtained.

Several methods that use probabilistic generators have been proposed. One such proposal [21] uses semi-Markov models to model a sequence and presents an efficient algorithm that can find appearances of a particular sub-sequence in another sequence.
A different alternative is to find partial orders between symbols and build complex episodes out of serial and parallel compositions of basic events ([43], [26], [8]). A further development [47] builds a mixture model that generates sequences similar to the one that one wishes to model, with high probability. In this approach, an intuitively appealing model is derived from the data, but no particular applications in classification, prediction or similarity-based mining are presented. These models can be viewed as graph-based models, since one or more graphs that describe the relationships between basic events are used to model the observed sequences.
A class of other approaches aim at inferring grammars from the given time sequence. Extensive research in grammatical inference methods has led to many interesting results ([48], [31], [51],) but has not yet found their way into a significant

number of real applications. Usually, grammatical inference methods are based on
discrete search algorithms that are able to infer very complex grammars. Neural netbased
approaches have also been tried (see, for example, part VI of reference [49])
but have been somewhat less successful when applied to complex problems ([32],
[22]). However, some recent results have shown that neural-network based inference
of grammars can be used in realistic applications [23].
The inferred grammars can belong to a variety of classes, ranging from deterministic
regular grammars ([39], [38], [52]) to stochastic ([30], [61], [10]) and context
free grammars ([58], [59]). It is clear that the grammar models obtained by induction
from examples can be used for prediction and classification, but, so far, these methods
have not been extensively applied to actual problems in data mining.

3.5 Transactional databases with timing information

A type of time sequences that does not easily match any of the classes described
above are transactional datasets that incorporate timing information. For example,
one might have a list of items bought by customers and would like to consider the
timing information contained in the database. This type of high-dimensional discrete
data cannot be modelled effectively by any of the previously described approaches.
Although the modelling and representation of this type of data is an important
problem in temporal data mining, to our knowledge, there exist only a few proposals
for the representation of data that has this kind of characteristics. In one of the proposed
representations [5] each particular set of items is considered a new symbol of
the alphabet and a new sequence is created from the original one, by using this new
set of symbols. These symbols are then processed using methods that aim of finding
common occurrences of sub-sequences in an efficient way. This representation is
effective if the objective is to find common events and significant correlations in
time, but fails in applications like prediction and classification. Other authors have

used a similar representation [28], although they have proposed new methods to
manipulate it.

IV. SIMILARITY MEASURES FOR SEQUENCES

After representing each sequence in a suitable form, it is necessary to define a similarity
measure between sequences, in order to determine if they match. An event is
then identified when there are a significant number of similar sequences in the database.

An important issue in measuring similarity between two sequences is the ability to
deal with outlying points, noise in the data, amplitude differences (*scaling problems*)
and the existence of gaps and other time axis distortion problems. As such, the similarity
measure needs to be sufficiently flexible to be applied to sequences with different
lengths, noise levels, and time scales.
There are many proposals for similaritude measures. The model used to represent
sequences clearly has a great impact on the similarity measure adopted, and, therefore,
this section follows an organization that closely matches that of section 3.

4.1 Distances in time-domain continuous representations

When no pre-processing is applied to the original sequence, a simple approach to
measure the similarity between two sequences consists on comparing the *ith* element
of each sequence. This simple method, however, is insufficient for many applications.

The similarity measure most used with time-domain continuous representations is
the Euclidean distance, by viewing each sub-sequence with *n* points as a point in Rn.
In order to deal with *noise*, *scaling* and *translation problems*, a simple improvement

consists in determining the pairs of sequences portions that agree in both sequences after some linear transformation is applied. A concrete proposal [3] achieves this by finding one window of some fixed size from each sequence, normalizing the values in them to the [-1, 1] interval, and then comparing them to determine if they match.
After identifying these pairs of windows, several non-overlapping pairs can be joined, in order to find the longest match length. Normalizing the windows solves the scaling problem, and searching for pairs of windows that match solves the translation problem.
A more complex approach consists in determining if there is a linear function *f*, such that a long subsequence of one sequence can be approximately mapped into a long subsequence of the other [14]. Since, in this approach, the subsequences don't necessarily consist of consecutive original elements but only have to appear in the same order, this similarity measure allows for the existence of gaps and outlying points in the sequence.

In order to surpass the high sensibility of the Euclidean distance to small distortions in the time axis, the technique of *Dynamic Time Warping* (DWT) was proposed to deal with temporal sequences. This technique consists on aligning two sequences so that a predetermined distance measure is minimized. The goal is to create a new sequence, the *warping path*, which elements are (*i*, *j*) points with *i* and *j* the indexes of original sequences elements that match [7]. Some algorithms that improve the performance of dynamic time warping techniques were proposed in ([36], [65]).
A different similarity measure consists on describing the distance between two sequences using a probabilistic model [34]. The general idea is that a sequence can be 'deformed', according to a prior probability distribution, to generate the other sequence. This is easily accomplished by composing a *global shape* sequence with *local features*, where each of one is allowed some degree of deformation. In this way, some elasticity is allowed in the global shape sequence, permitting stretching in time and distortions in amplitude. The degree of deformation and elasticity are governed by the prior probability distributions [34].

4.2 Distances in Transformation Based Methods

When Transformation Based Representations are used, a simple approach is to compare the points, in the new domain, which represent each sequence. For example, the similarity between two sequences may be reduced to the similarity between two points in the frequency domain, again measured using the Euclidean distance ([1], [11]). In order to allow for the existence of outlying points in the sequence, the discovery of sequences similar to a given query sequence is reduced to the selection of points in an -neighborhood of the *query point*. However, this method may allow for false hits, and after choosing the neighbor points in the new domain, the distance in the time domain is computed to validate the similarity between the chosen sequences.
Notice that this method implies that the sequences have the same length, and in order to be able to deal with different lengths, the method was improved to find relevant subsequences in the original data [16]. Another important issue is the fact that to compare different time series, the *k* Fourier coefficients chosen to represent the sequences must be the same for all of them, which could not be the optimal one for each sequence.

4.3 Similarities measures in discrete spaces

When a sequence is represented as a sequence of discrete symbols of an alphabet, the similarity between two sequences is, most of the times, achieved by comparing each element of one sequence with the correspondent one in the other sequence, like in the first approach described in 4.1.
However, since the discrete symbols are not necessarily ordered some provisions must be taken to allow for blurry matching.

One possible approach is to use the Shape Definition Language, described in 3.3, where a *blurry* match is automatically possible. This is achieved since the language

has defined a set of operators, such as *any*, *noless* or *nomore*, that allows describing overall shapes, solving the existence gaps problem in an elegant way [4].
Another possibility is to use well-known algorithms for string editing to define a distance over the space of strings of discrete symbols ([33], [45]). This distance reflects the amount of work needed to transform a sequence to another, and is able to deal with different sequences length and gaps existence.

4.4 Similarity measures for generative models

When a generative model is obtained from the datasets, the similarity measure between sequences is obtained in an obvious way by how close the data fits one of the available models. For deterministic models (graph-based models [43], [26], and deterministic grammars [32], [22], [58], [59]), verifying if a sequence matches a given model will provide a *yes* or *no* answer, and therefore the distance measures are necessarily discrete, and, in many cases, will take only one of two possible values.
For stochastic generative models like Markov chains ([21]), stochastic grammars ([30], [61], [10]) and mixture models ([47]), it is possible to obtain a number that indicates the probability that a given sequence was generated by a given model.
In both cases, this similaritude measure can be used effectively in classification problems, without the need to resort to complex heuristic similarity measures between sequences.

V. MINING OPERATIONS

5.1 Discovery of Association Rules

One of the most common approaches to mining frequent patterns is the *apriori* method [2] and when a transactional database represented as a set of sequences of transactions performed by one entity is used (as described in section 3.5), the manipulation of temporal sequences requires that some adaptations be made to the *apriori* algorithm. The most important modification is on the notion of *support*: support is now the fraction of entities, which had consumed the *itemsets* in any of their possible transactions [5], i.e. an entity could only contribute one time to increment the support of each *itemset*, beside it could had consumed that *itemset* several times.
After identifying the large *itemsets*, the *itemsets* with support greater than the minimum support allowed, they are translated to an integer, and each sequence is transformed in a new sequence, whose elements are the large *itemsets* of the previous-one. The next step is to find the large sequences. For achieve this, the algorithm acts iteratively as *apriori*: first it generates the candidate sequences and then it chooses the large sequences from the candidate ones, until there are no candidates.

One of the most costly operations in *apriori*-based approaches is the candidate generation. A proposal to frequent pattern mining states that it is possible to find frequent patterns avoiding the candidate generation-test [28]. Extending this to deal with sequential data is presented in [29].
The discovery of relevant *association rules* is one of the most important methods used to perform data mining on transactional databases. An effective algorithm to discover association rules is the *apriori* [2]. Adapting this method to deal with temporal information leads to some different approaches.
Common sub-sequences can be used to derive association rules with predictive value, as is done, for instance, in the analysis of discretized, multi-dimensional time series [24].
A possible approach consists on extending the notion of a typical rule X Y (which states if X occurs then Y occurs) to be a rule with a new meaning: X
T Y

(which states: if X occurs then Y will occur within time T) [15]. Stating a rule in this new form, allows for controlling the impact of the occurrence of an event to the other event occurrence, within a specific time interval.
Another method consists on considering cyclic rules [53]. A *cyclic rule* is one that occurs at regular time intervals, *i.e*. transactions that support specific rules occur periodically, for example at every first Monday of a month. In order to discover these rules, it is necessary to search for them in a restrict portion of time, since they may occur repeatedly at specific time instants but on a little portion of the global time considered. A method to discover such rules is applying an algorithm similar to the *apriori*, and after having the set of traditional rules, detects the cycles behind the rules. A more efficient approach to discover cyclic rules consists on inverting the process: first discover the cyclic large *itemsets* and then generate the rules. A natural extension to this method consists in allowing the existence of different time units, such as days, weeks or months, and is achieved by defining calendar algebra to define and manipulate groups of time intervals. Rules discovered are designated *calendric association rules* [54].
A different approach to the discovery of relations in multivariate time sequences is based on the definition of N-dimensional transaction databases. Transactions in these databases are obtained by discretizing, if necessary, continuous attributes [42]. This type of databases can then be mined to obtain association rules. However, new definitions for association rules, support and confidence are necessary. The great difference is the notion of address, which locates each event in a multi-dimensional space and allows for expressing the confidence and support level in a new way.

5.2 Classification

Classification is one of the most typical operations in supervised learning, but hasn't deserved much attention in temporal data mining. In fact, a comprehensive search of applications based on classification has returned relatively few instances of actual uses of temporal information.

One classification method to deal with temporal sequences is based on the *merge* operator [35]. This operator receives two sequences and returns "a sequence whose shape is a compromise between the two original sequences"[35]. The basic idea is to iteratively merge a typical example of a class with each positive example, building a more general model for the class. Using negative examples is also possible, but then is necessary to emphasize the shape differences. This is accomplished by using an influence factor to control the merge operator function: a positive influence factor implies the generalization of the input sequences, and a negative one conduces to exaggerate the differences between the positive and negative shapes.
Since traditional classification algorithms are difficult to apply to sequential examples, mostly because there is a vast number of potentially useful features for describing each example, an interesting improvement consists on applying a preprocessing mechanism to extract relevant features. One approach [40] to implement this idea consists on discovering frequent subsequences, and then using them, as the relevant features to classify sequences with traditional methods, like Naive Bayes or Winnow.
Classification is relatively straightforward if generative models are employed to model the temporal data. Deterministic and probabilistic models can be applied in a straightforward way to perform classification [39] since they give a clear answer to the question of whether a sequence matches a given model. However, actual applications in classification have been lacking in the literature.

5.3 Non-supervised Clustering

The fundamental problem in clustering temporal sequences databases consists on the discovery of a number of clusters, say *K*, able to represent the different sequences.

There are two central problems in clustering: choose the number of clusters and initialize their parameters. Another important problem appears, when dealing with temporal sequences: the existence of a meaningful similarity measure between sequences.

Considering that *K* is known, if a sequence is viewed as being generated according to some probabilistic model, for example by a Markov model, clustering may be viewed as modeling the data sequences as a finite group of *K* sequences in the form of a finite mixture model. Through the EM algorithm their parameters could be estimated and each *K* group would correspond to a cluster [61]. Learning the value of *K*, if it is unknown, may be accomplished by a Monte-Carlo cross validation approach as suggested in [60].

A different approach proposes to use a hierarchical clustering method to cluster temporal sequences databases [37]. The algorithm used is the COBWEB [18], and it works on two steps: first grouping the elements of the sequences, and then grouping the sequences themselves. Considering a simple time series, the first step is accomplished without difficulties, but to group the sequences is necessary to define a generalization mechanism for sequences. Such mechanism has to be able to choose the most specific description for what is common to different sequences.

5.4 Applications in Prediction

Prediction is one of the most important problems in data mining. However, in many cases, prediction problems may be formulated as classification, association rule finding or clustering problems. Generative models can also be used effectively to predict the evolution of time series.

Nonetheless, prediction problems have some specific characteristics that differentiate them from other problems. A vast literature exists on computer-assisted prediction of time series, in a variety of domains ([17], [64]). However, we have failed to find in the data mining literature significant applications that involve prediction of time series and that do not fall into any of the previously described categories.

Granted, several authors have presented work that aims specifically at obtaining algorithms that can be used to predict the evolution of time series ([46], [42]), but the applications described, if any, are used more to illustrate the algorithms and do not represent an actual problem that needs to be solved. One exception is the work by Giles *et all* [23], which combines discretization with grammar inference and applies the resulting automata to explicitly predict the evolution of financial time series. In the specific domain of prediction, care must be taken with the domain where prediction is to be applied. Well-known and generally accepted results on the inherent unpredictability of many financial time series [17] imply that significant gains in prediction accuracy are not possible, no matter how sophisticated the techniques used.

In other cases, however, it will be interesting to see if advances in data mining techniques will lead to significant improvements in prediction accuracy, when compared with existing parametric and non-parametric models.

VI. CONCLUSIONS

We have presented a comprehensive overview of techniques for the mining of temporal sequences. This domain has relationships with many other areas of knowledge, and an exhaustive survey that includes all relevant contributions is well beyond the scope of this work.

Nonetheless, we have surveyed and classified a significant fraction of the proposed approaches to this important problem, taking into consideration the representations they utilize, the similarity measures they propose and the applications they have been applied to.

In our opinion, this survey has shown that a significant number of algorithms exist

for the problem, but that the research has not be driven so much by actual problems
but by an interest in proposing new approaches. It has also shown that the researchers
have ignored, at least for some time, some very significant contributions
from related fields, as, for example, Markov-process based modeling and grammatical
inference techniques. We believe that the field will be significantly enriched if
knowledge from these other sources is incorporated into data mining algorithms and
applications.

REFERENCES

1. Agrawal, R., Faloutsos, C., Swami, A.: Efficient similarity search in sequence databases.
FODO - Foundations of Data Organization and Algorithms Proceedings
(1993) 69-84
2. Agrawal, R., Srikant, R.: Fast Algoritms for Mining Association Rules. VLDB
(1994) 487-499
3. Agrawal, R., Lin, K., Sawhney, H., Shim, K.: Fast Similarity Search in the Presence
of Noise, Scaling, and Translation in Time-Series Databases. VLDB (1995) 490-501
4. Agrawal, R., Giuseppe, P., Edward, W.L., Zait, M.: Querying Shapes of Histories.
VLDB (1995) 502-514
5. Agrawal, R., Srikant, R.: Mining sequential patterns. ICDE (1995) 3-14
6. Antunes, C.: Knowledge Acquisition System to Support Low Vision Consultation.
MSc Thesis at Instituto Superior Técnico, Universidade Técnica de Lisboa. (2001)
7. Berndt, D., Clifford, J.: Finding Patterns in Time Series: a Dynamic Programming
Approach. In: Fayyad, U., Shapiro, G., Smyth, P., Uthurusamy, R. (eds.): Advances
in Knowledge Discovery and Data Mining. AAAI Press. (1996) 229-248