# A Data Mining Methodology and Its Application to Semi-Automatic Knowledge Acquisition


Mika Klemettinen
Dept of Computer Science
University of Helsinki
P.O.Box 26, FIN-00014
Helsinki, Finland
mklemett@cs.Helsinki.FI

Heikki Mannila
Dept of Computer Science
University of Helsinki
P.O.Box 26, FIN-00014
Helsinki, Finland
mannila@cs.Helsinki.FI

Hannu Toivonen
Dept of Computer Science
University of Helsinki
P.O.Box 26, FIN-00014
Helsinki, Finland
htoivone@cs.Helsinki.FI



### Abstract

*We introduce a methodology for knowledge discovery in databases (KDD) where one first discovers large collections of patterns at once, and then performs interactively retrieves subsets of the collection of patterns. The proposed methodology suits such KDD formalisms as association and episode rules, where large collections of potentially interesting rules can be found efficiently.*

*We present methods that support interactive exploration of large collections of rules. With these methods the user can flexibly specify the focus of interest, and also iteratively refine it.*

*We have implemented our methodology in the* TASA *system which discovers patterns in telecommunication alarm databases. In this paper, we give concrete examples of how to use frequent patterns in the construction of alarm correlation expert systems.*


## 1 Introduction

Data mining and knowledge discovery (KDD) is a promising approach for semi-automatic knowledge acquisition from large masses of existing data. In KDD one typically wants to find a small collection of interesting (e.g., strong, useful, surprising, or valuable) patterns, rules etc., that occur or hold in the data. Consider as an example the task of fault management in telecommunication networks. The networks produce large amounts of alarm information which must be analyzed and interpreted before faults can be located. So-called alarm correlation systems (see, e.g., [6, 8]) are expert systems for analysis of alarm data. They are quite popular, but acquiring all the knowledge necessary for constructing an alarm correlation system for a network and its elements is difficult. The complexity and diversity of network elements and the large variation in the patterns of alarm occurrences pose serious problems for network management experts trying to build a correlation model. Moreover, characteristics of the alarm flow change often, as new equipment and software releases are added to the network. On the other hand, large amounts of alarms are available, so the area is potentially good for KDD application.

Most knowledge discovery systems prune and rank the set of discovered patterns during the search for the patterns. We adopt a different approach. We use methods that locate all rules from a given class that satisfy certain simple frequency constraints. Discovered rules are not pruned automatically, but the user has explicit control over the operations on a rule set. This way the user can always be sure that the view on the data is in a sense complete: the system has not put aside any rules automatically.

Our approach is based on two simple facts:

- **Iteration is essential in KDD**. It can be hard, or even impossible to specify beforehand what is interesting or relevant.
- **Pattern generation from very large data sets is time consuming.** Efficient algorithms exist, but the pattern extraction phase still is a barrier for smooth interaction.

We propose a KDD process where emphasis is on two central phases:

1. In the pattern discovery phase, find *all* potentially interesting patterns according to some rather loose criteria.
2. Provide flexible methods in the presentation phase for iteratively and interactively creating different views to the discovered patterns.

Preprocessing and transformation steps are typically needed before patterns can be discovered (Fig-

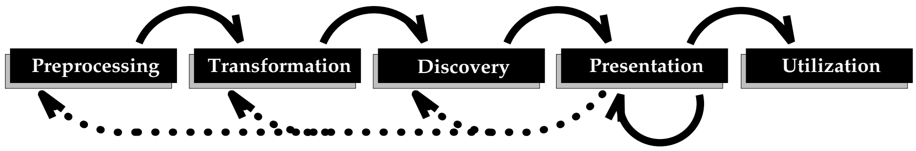


Figure 1: The KDD process model.

ure 1). A goal of our model is to avoid repeating the time consuming phases of preprocessing, transformation, and discovery.

We have implemented our methodology in the TASA system [7] which discovers patterns in telecommunication alarm databases and provides tools for interactive identification of the relevant patterns.

The rest of this paper is organized as follows. In Section 2 we briefly discuss two motivating examples and give the basic properties of the knowledge discovery methods we have employed. The methodology is described in Section 3. Then in Section 4 we concentrate on the pattern presentation phase of our KDD process. In Section 5 we summarize our experiences in applying the methodology in building alarm correlation systems. Finally, Section 6 is a short conclusion.

**Related work** KDD process and methodology have been discussed in [3, 4, 5], where the importance of interaction and iteration is also underlined. The iteration covers, however, either the whole process or at least the pattern discovery and presentation phases.

To the best of our knowledge, none of the existing KDD systems supports the methodology proposed here. For instance, in Explora [9], 49er [16] and Kefir [11] the user can interactively change the focus, but that requires a new pattern discovery. Additionally, the approach of discovering all patterns can be contrasted with numerous methods, e.g., in machine learning, which aim directly at more focused discovery and produce one or at most few patterns that match the given problem specification. These methods usually require that the searched or learned subject is described quite carefully in advance.

# 2 Examples of Application Domains

In this section we outline first a very simple application domain (student enrollment data) and then discuss in more detail a significantly harder case, the analysis of telecommunications alarm data.

## 2.1 Student Enrollment Data

The first, simple case study concerns *association rules* [1] in the student enrollment database of courses in computer science at the University of Helsinki. In the course enrollement database we may find an association rule

```
IF    Introduction to Unix
THEN  Programming in C
WITH  conf(0.84)   freq(0.34).
```

The rule states that 84% of the students that have taken "Introduction to Unix" also have taken "Programming in C", and that 34% of all the students actually have taken both courses. More formally, an association rule is an expression $X \Rightarrow Y$, where $X$ and $Y$ are sets of predicates. Given a set of students, the *confidence* of such rule is the conditional probability with which predicates in $Y$ are satisfied by a tuple in the database given that predicates in $X$ are satisfied. The rule is called *frequent*, if its *frequency* exeeds a user-given threshold; i.e., if all the predicates in $X \cup Y$ occur together at least a user-specified minimum number of times.

The goal in analyzing student enrollment data is to obtain accurate and useful information about the interrelationships between enrollments to various courses. Such relationships can be valuable, e.g., for expert systems aimed at planning curriculums and allocating resources.

## 2.2 Telecommunication Alarm Data

The other case study, which deals with a more interesting knowledge acquisition effort, handles rules derived from episodes [14] in telecommunication network alarm sequences. A telecommunication alarm database contains event data from a number of interconnected components, such as switches. Alarms are produced by the components to report abnormal situations. There are typically thousands of alarms a day, and they are of thousands of different types.

Telecommunication networks are growing fast in size and complexity, and at the same time their management is becoming more difficult. The task of identifying and correcting faults in telecommunication networks is a critical task of network management. Thus, the techniques of alarm correlation — automatic filtering of redundant alarms, identification of faults, and suggestions for corrective actions — are valuable.

We adopt the following view to alarm correlation, similar to the one taken, e.g., in [8]. Abstractly, the input to a correlation system is an ordered *alarm sequence*. An alarm consists of *time*, *sender*, and *alarm message* fields. The time is recorded by the sender, typically at a granularity of one second. The sender of the alarm can be identified at the level of, e.g., a network element. The alarm message contains all the available information about the problem.

A *correlation pattern* describes a situation that can be recognized in an alarm sequence within a time window of a given length. Typically, a correlation pattern is an expression on the set of active alarms of, e.g., the last five minutes. Correlation patterns are constructed as logical combinations of alarm predicates; alarm predicates are derived from the fields of alarm messages by constructing boolean-valued assertions. If in a given window there is a set of alarms that matches the correlation pattern, then the set is said to be an *occurrence* of the pattern. Associated with each correlation pattern is a *correlation action*, which is to be executed when there is an occurrence of the corresponding pattern in a window.

Constructing an alarm correlation system for a network and its elements is, however, difficult. Both networks and network elements evolve quickly over time, so a correlation system is never complete. It also takes time for the experts to learn new correlations and to modify existing ones.

KDD methods can be used in constructing a correlation system. The central idea is to discover recurrent patterns of alarms that are potentially useful in the specification of correlation patterns. A discovered pattern can be, for instance, a set of alarms that occurs frequently within a time window of a given width, or a pattern can state that a certain alarm tends to be followed by another alarm from the same sender.

*Episode rules* and *episodes* [12, 14] are a modification of the concept of *association rules* and *frequent sets*, applied to sequential data, e.g., telecommunication alarms. An episode rule is a rule of the form[1]

| | | | | |
|---|---|---|---|---|
| `IF` | *link alarm* | | | |
| | *link failure* | | | |
| `THEN` | *high fault rate* | | | |
| `WITH` | `[20]` | `[40]` | `conf(0.23)` | `freq(246/1056)` |

which tells us that in 23 per cent of the cases, where *link alarm* and *link failure* occurred within 20 seconds, also *high fault rate* occurred within 40 seconds. Moreover, in the data the left-hand side occurred 1 056 times and in 246 cases it was followed by the right-hand side, within the given time windows. An episode is *parallel*, if there is no requirement for the order of the events within the time window, and *serial*, if the events are required to occur in a certain order.

In general, we are not interested in rules with a negligible confidence, e.g., less than 20 per cent; it is typical to select only those rules with a confidence exceeding a given *confidence threshold*. For finding all potentially interesting episode rules in a given event sequence, the necessary parameters are the frequency threshold, a set of window sizes which may be used in the episode rules, and the confidence threshold. The method that we have used to discover frequent episodes and episode rules in our data is described in [12].

In summary, our scenario for building alarm correlation systems is the following (see Figure 2).

- First, a large database of alarms is analyzed offline, and connections between sets of alarms are discovered automatically.
- Then, for the construction of an alarm correlation system, the network management specialists have at hand a collection of alarm patterns which they can use in specifying the alarm correlation system.
- In the final step, the correlation rules are applied in real-time fault identification.

## 3 The Methodology

KDD is an iterative process and requires interaction with the user. A general KDD process (adapted from [5]) consists of several phases and iteration between them (Figure 1). Our methodology gives special roles to the pattern discovery and pattern presentation phases of the KDD process. In our approach, large collections of patterns are discovered at once. Then, iterative information retrieval methods are used to give flexible views to the discovered patterns. The emphasis in our methodology is in the pattern presentation part, where interesting information is searched for iteratively, without repeating the time consuming pattern discovery phase.

In the first two phases of the whole process the raw data is collected and prepared into a suitable form for the pattern discovery. A general overview of the data can be produced at this phase. Attributes identified as irrelevant are removed, and potentially useful new attributes can be derived. The preprocessing in our methodology does not, however, mean that one explicitly defines interestingness as in some other methodologies.

All relevant background knowledge should be taken advantage of. For instance, in the case of telecommunication alarm data, attributes can be derived to contain, e.g., information about the network topology.

[1] The actual predicates occurring in the example rules have been changed.

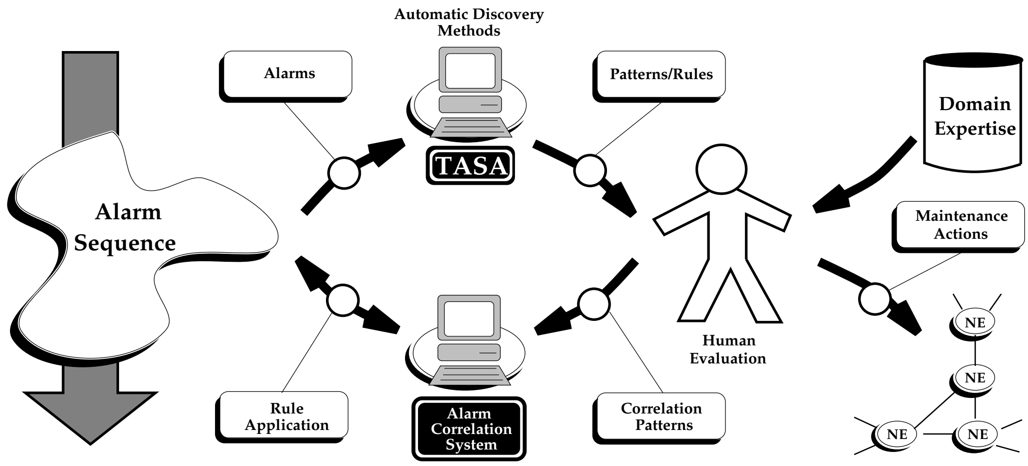


Figure 2: An alarm correlation model using discovered alarm patterns.

In the pattern discovery phase all potentially interesting patterns are generated from the data set. The (rather loose) criteria for interestingness can be, e.g., frequency, prevention of taking rarely occurring phenomena into account, or strength of the patterns.

The user specifies the following parameters for the discovery of the rules using Tasa.

- The alarm predicates are given by the user. For episode rules, the type of the alarm and the sender of the alarm are the most typical predicates. For association rules in turn, much larger predicate classes are useful.
- In the case of episodes, the user specifies whether serial or parallel episodes are searched for.
- For episode rules, the user also supplies a set of time bounds with which the rules are constructed. Our fault management experts have typically preferred time windows ranging from 5 seconds to 10 minutes.
- For both episode and association rules, a *frequency threshold* is given by the user. The methods output all episode and association rules specified by the parameters above, whose frequency exceeds or is equal to the user-specified threshold.
- For both episode and association rules, the user also specifies a *confidence threshold*. The algorithms then output all episode and association rules whose confidence is at least equal to the threshold.

The discovery method produces each episode rule and association rule satisfying the above conditions. The conditions are typically quite loose, so large amounts of rules are discovered. For each rule, Tasa outputs the confidence and the frequency, and an estimate of the statistical significance of the rule.

**Example 1** *The course enrollment database consists of registration information of 1 066 students who had registered for at least two courses. There are 2 010 association rules that match at least 11 students (i.e., the frequency threshold is* 0.01*). There are 6 715 rules that match at least 6 students.*

*In contrast, in the alarm sequence containing 43 478 alarms from a period of 10 days, there are 5 498 serial episode rules that hold with a frequency threshold of 10 (absolute value) with a maximum time bound of 60 seconds. There are 1 497 association rules between attribute values of individual alarms, that match at least 435 alarms (i.e., frequency threshold is* 0.1*).* □

The goal of the methodology is that returning to the pattern discovery phase is needed as seldom as possible. It is realistic to expect, however, that the algorithms need to be run several times. Parameter settings may need adjustment, or errors in the data cleaning phase may be revealed so that returning to the first phase of the process is necessary.

The basic idea — iteration in the pattern presentation phase — can be applied to formalisms that have certain properties:

- The desired focus is not known definitely in advance.
- There is an algorithm that can produce all patterns from a class of potentially interesting patterns.
- The time requirement for discovering all potentially interesting patterns is not considerably

longer than if the discovery was focused to a small subset of the potentially interesting patterns.

There are several KDD formalisms that fit in this setting. Association rule [1, 2] and episode rule [12, 14] algorithms can efficiently discover thousands of rules from relatively simple databases. A formal treatment of the setting of discovering all interesting sentences and an analysis of a general algorithm can be found in [13].

The presentation of discovered knowledge is a main part of our methodology. In this phase the relevant patterns should be located from large collections of potentially interesting patterns. The problem of locating a small set of truly relevant information is a generic problem in data mining: while formal statistical criteria for strength and significance of the discovered items abound, it is much harder to know which parts of the discovered knowledge really are useful for the user.

Finally, the discovered knowledge has to be understood and applied, e.g., in an expert system. The role of the KDD system is to provide appropriate tools for displaying and browsing the knowledge, methods for expressing the desired focus, and to support iterative specification of the focus.

## 4 Presentation of Rules

As a result of the pattern discovery phase of our KDD process model, all patterns matching the specified criteria are produced. All the patterns are, however, not supposed to be interesting to a particular user or in a particular situation. Discovered rules can fail to be interesting for several reasons.

- **A rule can correspond to prior knowledge or expectations.** For example, it can be known from the implementation that if an element sends an alarm $A$, it will also send an explanatory alarm $B$.
- **A rule can refer to uninteresting attributes or attribute combinations.** E.g., a rule containing low priority alarms may be non-interesting, and the user would like to filter out all such rules.
- **Rules can be redundant.** For example, rule (b) below has no additional predictive power regarding the course "Unix Platform" over rule (a), and could be pruned as redundant.

```
(a)  IF    Data Communications
           Programming in C
     THEN  Unix Platform
     WITH  conf(0.14)   freq(0.03)
(b)  IF    Data Communications
           Programming in C
           Introduction to Unix
     THEN  Unix Platform
     WITH  conf(0.14)   freq(0.03)
```

The aim of the presentation phase of our methodology is to support efficient, interactive, and focused views to the rules. In this section, we present methods for exploring large sets of association and episode rules. The ideas of this section can be best applied on large, unstructured sets of rules and other similar, simple pattern formalisms.

Three types of operations are useful in processing a large collection of rules:

- **Pruning**: reduction of the number of rules;
- **Sorting**: ranking of rules according, e.g., to statistical significance; and
- **Structuring**: organization of the rules, e.g., to clusters or hierarchies.

Obvious pruning and sorting criteria for association and episode rules are rule confidence, frequency, and statistical significance. Rules can be pruned by setting thresholds on these properties. Examples of the effect of threshold-value based pruning with alarm data are presented in Figure 3.

*Templates* [10], pattern expressions that describe the form of rules that are to be selected or rejected, are one way of focusing the view to a large space of rules. A template is an expression

$$A_1, \ldots, A_k \Rightarrow A_{k+1}, \ldots, A_l,$$

where each $A_i$ is either an attribute name, a class name, or an expression $C+$ or $C*$, where $C$ is a class name.[2] Here $C+$ and $C*$ correspond to one or more and zero or more instances of the class $C$, respectively. A rule

$$B_1, \ldots, B_m \Rightarrow B_{m+1}, \ldots, B_n$$

matches a template if the rule can be considered to be an instance of the pattern. The order of the components in the templates has no significance if the attribute order in the rules is not significant (association rules and parallel episode rules). With templates, the user can explicitly specify both what is interesting and what is not. To be interesting, a rule has to match a *selective template.* If a rule, however, matches an *unselective template,* it is considered uninteresting. To be presented to the user, a rule must be considered interesting

[2] $A_i$ can also be a regular expression.

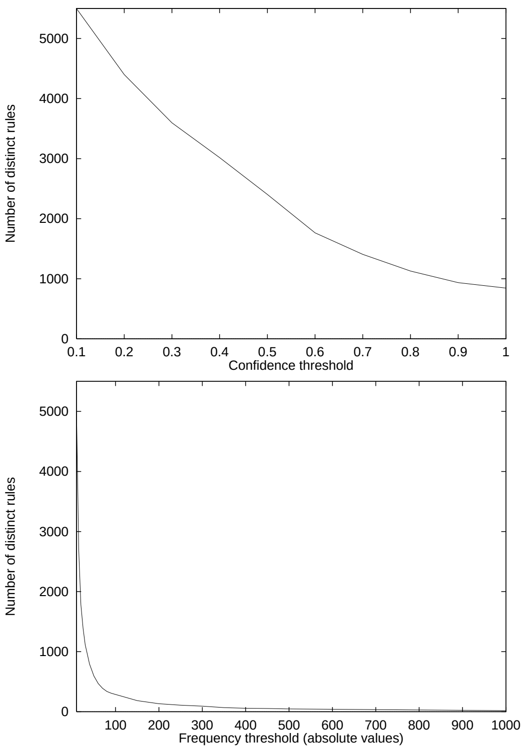


Figure 3: Effect of rule confidence and frequency thresholds to the number of rules in alarm data.

— i.e. match one of the selective templates — and it must not be uninteresting — i.e. not match with any of the unselective templates.

**Example 2** *In our example domain of computer science course enrollment, the courses are divided into three classes: basic, undergraduate, and graduate courses. In addition, all courses belong to the parent class 'any course'. Thus, we have such generalizations as in Figure 4.*

*Using the selective template (a) below the user obtains a number of rules with (at least) a graduate course on the IF part and the course "Design and Analysis of Algorithms" on the THEN part. Examples of such rules are (b), (c) and (d).*

*Of the 2 010 rules in the example database there are 15 rules that match the above specifications and have confidence of 0.2 or more and frequency of 0.01 or more. This sort of rules give an idea of what the students do in addition to taking "Design and Analysis of Algorithms", and the discovered rules can be utilized when planning the course. Note that here the IF–THEN structure implies no temporal order.*

| | | |
|---|---|---|
| (a) | `IF` | *Graduate Course* |
| | | *Any Course∗* |
| | `THEN` | *Design and Analysis of Algorithms* |
| (b) | `IF` | *Compilers* |
| | | *Introduction to Computers* |
| | `THEN` | *Design and Analysis of Algorithms* |
| | `WITH` | `conf(0.16)` `freq(0.01)` |
| (c) | `IF` | *Neural Networks* |
| | | *Data Communications* |
| | `THEN` | *Design and Analysis of Algorithms* |
| | `WITH` | `conf(0.33)` `freq(0.01)` |
| (d) | `IF` | *User Interfaces* |
| | | *Information Systems* |
| | `THEN` | *Design and Analysis of Algorithms* |
| | `WITH` | `conf(0.02)` `freq(0.10)` |

*The user may consider some of the resulting rules uninteresting, such as (b) including basic courses. By adding an unselective template with "Basic Course" and "Any Course∗" on the IF part and "Any Course∗" on the THEN part the user can filter out all rules with a basic course on the left-hand side.* □

This technique is surprisingly powerful. For example, background knowledge about the normal curricula of computer science studies can be taken into account by using templates that unselect all rules that correspond to the normal course of studies.

**Example 3** *As an example of how the system can be used in case of alarm data, consider the following scenario.*[3] *Assume the network manager has used* TASA *to discover episode rules for the past 10 days. The manager knows that during that period, a device* AA–16 *caused serious problems and loss of money. First he selects all rules that have* AA–16 *in the THEN part, thus traceing alarms that potentially have some causal relationship with the problem in consideration.*

*The number of such rules is, however, large: 1 848. The network manager decides to restrict the THEN part to only contain one alarm. The number of selected rules is still large, 901, so he adds an unselective template to remove all rules with devices from the same network area (indicated by the first two characters in the device identificator, here* AA*), which he knows by experience not to have any effect to each other.*

*The number of rules is still large, 240 rules, so he further tightens the scope and adds a further selective template to get rules with at most two alarms on the IF part and confidence of at least 75%. The resulting rule set contains 19 rules from the 5 498 original ones. By browsing the resulting rules he finally finds alarms which indicate that the problems with device* AA–16 *had been caused by high fault rate in a similar nearby device.* □

[3] The device name and the situation are fictional, but the numbers reflect actual results and the effect of our methods.

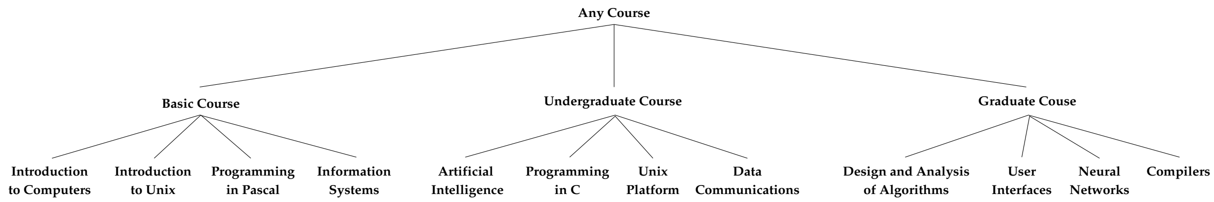


Figure 4: Course hierarchy.

It should be noted that sometimes considerable amounts of rules remain, even when the user has found the desired focus with the described methods. Automatic pruning, sorting, and structuring methods should at this point be available for invocation by the user, especially for removal of redundancy; see, e.g., [15] for *rule covers*, statistical methods, and clustering.

## 5 Application Experience

TASA has been in prototype use in four telecommunication companies since the beginning of 1995, and experiences are encouraging. A telecommunication network manager can use episode rules as correlation patterns, after possibly editing them and assigning them appropriate correlation actions. Discovered patterns can be used for filtering uninformative alarms, for combining alarms to construct hypothesis of faults, or for prediction of faults.

Episode and association rules are also useful for providing different views to the analyzed alarms, as in Example 3. Although the discovery algorithms are not directly applicable for on-line analysis, TASA has turned out to be useful also in network surveillance. The analysis can be rerun or augmented, e.g., every day or every hour. Recent patterns may point to yet unnoticed problems in the network.

The fault management experts in the telecommunication companies have found the approach and TASA useful in, e.g.,

- finding long-term, rather frequently occurring dependencies,
- creating an overview of a short-term alarm sequence, and
- evaluating the alarm data base consistency and correctness.

On the other hand, many of the rules discovered by TASA are deemed trivial by the network managers. Luckily, much of the trivial knowledge can be expressed and removed with templates.

**Example 4** *Consider an alarm correlation system which operates in real time and is also able to handle delayed alarms and slightly inaccurate time stamps. In the correlation patterns, delays are handled with a special* wait *function. Episode and association rules can be applied in this system in a rather straightforward way. The episode rule*

IF *link alarm* < *link failure*
THEN *high fault rate*
WITH [5] [60] conf(0.70) freq(700/1000)

*discovered by* TASA *can be coded in the system as follows:*

**if** "alarm type = link alarm" **then**
start time;
wait until "alarm type = link failure" or "time = 5 sec";
**if** "alarm type = link failure" **then**
display warning "high fault rate with 70% probability in 60 sec"
**else** forward the original alarm;

*That is, if a link alarm occurs and a link failure follows within 5 seconds, the rule right-hand side information is sent and the original alarms are suppressed. If a link failure does not follow within 5 seconds, the original link alarm is forwarded.*

*The correlation procedure can be enhanced using association rules. For example, assume that we have detected that if the alarm type is* link alarm*, the time of the day is office hours, and alarm severity is 1, then with probability of 95% the alarming element type is BS. After expert evaluation of the rule we know that such alarms from network elements of type BS can be ignored. However, if an element of any other type sends that alarm, it is an interesting one. The following correlation function removes such alarms:*

**if** "alarm type = link alarm" and
"office hours" and "severity = 1" and
"element type = BS" **then**
exit
**else** forward the original alarm;

□

Unexpected but useful dependencies have been found, e.g., between network elements which are not immediately connected in the network topology. Beginning from the first tests, discovered rules have been applied in alarm correlation systems.

## 6 Conclusion

We have described a knowledge discovery methodology that can be used to automate knowledge acquisition. The methodology has the following two distinctive characteristics: (1) a large collection of potentially relevant rules is discovered at once, and (2) different views can be formed on the discovered rules iteratively and interactively. The motivation is that after the discovery of all potentially interesting rules, iteration is very efficient.

As an example application area we descibed telecommunication networks alarm data. The flow of alarms should be correlated automatically to a more intelligible form, in order to facilitate identification and correction of faults. Unfortunately, the construction of an alarm correlation system requires a lot of both expertise and time, and is a process that never is complete.

Our experience in this field with both association and episode rules supports the claim that the methodology is useful in practice. For formalisms stronger than association and episode rules, strong focus in the pattern discovery phase may be essential for keeping the computation tractable. For well-defined problems this is probably fine, but for exploring regularities in large data sets the interaction may be cumbersome.

## References


[1] R. Agrawal, T. Imielinski, and A. Swami. Mining association rules between sets of items in large databases. In *Proceedings of ACM SIGMOD Conference on Management of Data (SIGMOD'93)*, pp. 207 – 216, May 1993.

[2] R. Agrawal, H. Mannila, R. Srikant, H. Toivonen, and I. Verkamo. Fast discovery of association rules. In *Advances in Knowledge Discovery and Data Mining*, pp. 307 – 328. AAAI Press, Menlo Park, CA, 1996.

[3] R. Brachman and T. Anand. The process of knowledge discovery in databases: A first sketch. In *Knowledge Discovery in Databases, Papers from the 1994 AAAI Workshop (KDD'94)*, July 1994.

[4] U. Fayyad, G. Piatetsky-Shapiro, and P. Smyth. The KDD process for extracting useful knowledge from volumes of data. *Communications of the ACM*, 39(11):27 – 34, November 1996.

[5] U. Fayyad, G. Piatetsky-Shapiro, and P. Smyth. From data mining to knowledge discovery: An overview. In *Advances in Knowledge Discovery and Data Mining*, pp. 1 –34. AAAI Press, Menlo Park, CA, 1996.

[6] R. Goodman, B. Ambrose, H. Latin, and C. Ulmer. Noaa – an expert system managing the telephone network. In *Integrated Network Management IV*, pp. 316 – 327. Chapman & Hall, London, 1995.

[7] K. Hätönen, M. Klemettinen, H. Mannila, P. Ronkainen, and H. Toivonen. Knowledge discovery from telecommunication network alarm databases. In *12th Int'l Conference on Data Engineering (ICDE'96)*, pp. 115 – 122, New Orleans, Louisiana, February 1996.

[8] G. Jakobson and M. Weissman. Alarm correlation. In *IEEE Network*, 7(6):52 – 59, November 1993.

[9] W. Kloesgen. Efficient discovery of interesting statements in databases. *Journal of Intelligent Information Systems*, 4(1):53 – 69, 1995.

[10] M. Klemettinen, H. Mannila, P. Ronkainen, H. Toivonen, and I. Verkamo. Finding interesting rules from large sets of discovered association rules. In *Proceedings of the Third Int'l Conference on Information and Knowledge Management (CIKM'94)*, pp. 401 – 407, Gaithersburg, MD, November 1994. ACM.

[11] C. Matheus, G. Piatetsky-Shapiro, and D. McNeill. Selecting and reporting what is interesting. In *Advances in Knowledge Discovery and Data Mining*, pp. 495 – 515. AAAI Press, Menlo Park, CA, 1996.

[12] H. Mannila and H. Toivonen. Discovering generalized episodes using minimal occurrences. In *Proceedings of the Second Int'l Conference on Knowledge Discovery and Data Mining (KDD'96)*, pp. 146 – 151, Portland, Oregon, August 1996. AAAI Press.

[13] H. Mannila and H. Toivonen. On an algorithm for finding all interesting sentences. In *Cybernetics and Systems, Volume II, The Thirteenth European Meeting on Cybernetics and Systems Research*, pages 973 – 978, Vienna, Austria, April 1996. Austrian Society for Cybernetic Studies.

[14] H. Mannila, H. Toivonen, and I. Verkamo. Discovering frequent episodes in sequences. In *Proceedings of the First Int'l Conference on Knowledge Discovery and Data Mining (KDD'95)*, pp. 210 – 215, Montreal, Canada, August 1995. AAAI Press.

[15] H. Toivonen, M. Klemettinen, P. Ronkainen, K. Hätönen, and H. Mannila. Pruning and grouping of discovered association rules. In *Workshop Notes of the ECML-95 Workshop on Statistics, Machine Learning, and Knowledge Discovery in Databases*, pp. 47 – 52, Heraklion, Greece, April 1995. MLnet.

[16] R. Zembowicz and J. Zytkow. From contingency tables to various forms of knowledge in databases. In *Advances in Knowledge Discovery and Data Mining*, pp. 329 – 349. AAAI Press, Menlo Park, CA, 1996.