Mining Expressive Temporal Associations From Complex Data[*]

Keith A. Pray
Department of Computer Science
Worcester Polytechnic Institute
100 Institute Road
Worcester, MA 01609 USA
kap@wpi.edu

Carolina Ruiz
Department of Computer Science
Worcester Polytechnic Institute
100 Institute Road
Worcester, MA 01609 USA
ruiz@wpi.edu

ABSTRACT

We introduce an algorithm for mining expressive temporal relationships from complex data. Our algorithm, AprioriSetsAndSequences (ASAS), extends the Apriori algorithm to data sets in which a single data instance may consist of a combination of attribute values that are nominal sequences, time series, sets, and traditional relational values. Datasets of this type occur naturally in many domains including health care, financial analysis, complex system diagnostics, and domains in which multi-sensors are used. AprioriSetsAndSequences identifies predefined events of interest in the sequential data attributes. It then mines for association rules that make explicit all frequent temporal relationships among the occurrences of those events and relationships of those events and other data attributes. Our algorithm inherently handles different levels of time granularity in the same data set. We have implemented AprioriSetsAndSequences within the Weka environment and have applied it to computer performance, stock market, and clinical sleep disorder data. We present here experimental results using financial data. We show that AprioriSetsAndSequences produces rules that express significant temporal relationships that describe patterns of behavior observed in the data set.

Categories and Subject Descriptors

H.2.8 [**Database Applications**]: Data mining; I.5.2 [**Design Methodology**]: Pattern analysis

General Terms

Algorithms, Performance

Keywords

temporal association rules, mining complex data

1. INTRODUCTION

This paper expands the general use of association rules [4] to complex temporal relationships essential to many domains. Our association rule mining approach discovers patterns in data sets whose instances consist of any combination of standard, set-valued, and sequential attributes. These data sets occur naturally in several scientific, engineering, and business domains, and are generally richer than the transactional, the relational, and the sequence data sets to which association rule mining has been applied. To date, our mining approach has been applied to computer system performance, stock market analysis [18], and clinical sleep disorder data [19]. Complex system diagnostics, network intrusion detection, and medical monitoring are some related domains to which this work can also be applied.

A main motivation for mining these temporal rules from complex data comes from our previous experience working on the performance analysis of a hardware and software backup/restore product. The time it takes to complete a backup can be of great importance. One of the most labor consuming tasks is to predict this time. A rule based expert system was built as a way to disseminate the performance knowledge used to predict backup time. This expert system quickly grew to include over six hundred rules. The need for automating the discovery of new rules was apparent.

To create a rule to handle a new feature or new technology, an estimation is made about how performance will be affected. Tremendous amounts of information are collected to verify that the estimation is accurate. For a small scenario including a backup server, a single client, the Ethernet network that connects them, and a single SCSI connected tape drive, an average of 2 MB of data can be collected every ten minutes that a test is run. This includes CPU and memory usage for all processes running on both server and client, I/O metrics for storage sub-systems and the tape drive and the connections between them and the client and server machines as well as network traffic. With a short test run of an hour, analyzing this amount of data manually is very impractical. The reasoning behind the initial estimation serves to focus analysis efforts but is usually inadequate. It is often the case that the estimation is not sufficiently accurate and more analysis must be done to identify the factors behind the observed performance. This process can take weeks or months and often involves running new tests and collecting even more data to be analyzed. With our approach to mine association rules from these data, the resulting rules

---

[*]

WOOD STOCK '97 El Paso, Texas USA

can be used to focus analysis efforts and possibly explain the observed behavior.

Figure 1 depicts a small sample of the type of complex data set that our mining approach applies to. In this computer performance data set, each instance (row) corresponds to a single test in which a process was run until it completed a task. The attributes describe the conditions under which the test was run and state information collected during the test including "standard" attributes such as processor speed and physical memory size; set-valued attributes such as the options or flags that were specified to the process and which algorithms the process implemented; and sequential attributes such as the CPU utilization percentage, memory usage, and the total number of processes running over time. Other such numeric and nominal sequential attributes not shown in Figure 1 include CPU usage of all other processes, memory usage of all other processes, I/O activity on main storage devices, memory paging, and process state (running, exit normally, exit without output). All time sequence attributes in a single data instance share the same time line.

Events of interest in the sequential attributes in this data set include among others *increases* and *decreases* in the utilization of a resource (CPU, memory), going above/below a certain usage threshold, and whether or not a process terminates normally.

A sample interesting temporal pattern in this domain is: "Java processes running a machine learning algorithm that are not given the -P option, that exhibit an increase in its memory usage, and that during this increase its memory paging also increases tend to, with likelihood 82%, exit prematurely". This temporal pattern is captured by our association rule:

flag=-P-missing & process(java)-mem-usage-increase [t_0,t_2]
& process(java)-page-increase [t_1,t_2] $\Rightarrow$
process(java)-exit-without-output [t_3,t_4], conf=82%.

Here, t_0, t_1, t_2, t_3, and t_4 are *relative* time indices that are used to express the relative order in which these events are observed.

The algorithm presented here to mine these temporal relationships, *AprioriSetsAndSequences*, takes as input a complex temporal data set as described, a set of predefined event types, and the standard Apriori minimum support and minimum confidence parameters. An event type is a template of a subsequence of interest in a time sequence attribute. Our mining algorithm starts by identifying occurrences of these event types in the corresponding sequence attributes as described in Section 2. Then, it generates all the frequent relationships among the temporal occurrences of these events and among these and the other non-temporal attributes. Section 3 describes the algorithm in detail. Since there are 13 ways of sorting just two events in time (*before, after, overlaps,* etc.) [7], and the number of possible orderings of a set of events grows exponentially with the number of events in the set, central issues in the design and implementation of our mining algorithm are the strategy used to prune unnecessary orderings from consideration, and the data structures used to effectively handle potentially frequent orderings of events. We describe these in Sections 3 and 4. These sections also describe our extensions of the basic notions of support and confidence needed to handle multiple occurrences of an event or a pattern in a complex data instance. Section 5 presents an evaluation of our mining algorithm in the stock market domain. Section 7 summarizes the contributions of this paper and discusses future work investigating alternative measures of item set interestingness and alternative search techniques for item set generation in the context of complex data.

2. IDENTIFYING EVENTS IN SEQUENCES

.

Events of interests are available in multiple domains. Examples of those are *head & shoulders reversal* and *ascending triangle* in stock market analysis [21], and *increase in CPU utilization* in the performance domain. An event can be described by a Boolean condition or by a template of the event "shape". Such a template can be for example a 2–dimensional curve.

Given a collection of domain–specific events of interest, we identify occurrences of those events in the sequential attributes. There are many methods available for matching patterns against a sequence. These include similarity search using Discrete Fourier Transform [3], bounding boxes for clustering over a Fourier Transform feature space [15], and dynamic programming time warping [9].

Work has also been done to discover common patterns in transactional sequences [28]. These are all valid methods for specifying and identifying events for AprioriSetsAndSequences.

Once the occurrences of an event of interest have been identified in the values of a sequential attribute, they are stored in a new event set attribute. We define an *event set attribute* as an attribute whose values are sets of an event type occurrences. As an example, consider an event attribute for the CPU time sequence attribute. The CPU time sequence attribute is the percentage of CPU usage in the overall computer system. This event attribute specifies when the CPU usage increases. Assume that in a particular data set instance I, increases in CPU usage occur from time 2 to 8, 13 to 19, and 35 to 47. Then, the I' value for the new attribute CPU-Increase is { [2,8], [13,19], [35,47] }.

AprioriSetsAndSequences mines temporal associations directly from events. This keeps intact the temporal information represented in the data set while eliminating much of the work involved in scanning the actual sequences during mining. The events are akin to indexes into the sequences.

3. BASIC ASAS

Input. ASAS takes as input a data set consisting of instances. Each instance has a set of attributes. Attributes can be of type nominal, set, and event set. An event set is simply a set whose elements are events. An event consists of three pieces of information. First is the time sequence attribute in which the event occurred. Second is the name of the event. Third is the period of time during which the event occurred. The time sequence attribute name and the event name together define the event item type.

ASAS takes parameters that are similar to those accepted by regular Apriori. In addition to minimum support and minimum confidence is the maximum number of event items of the same type allowed in a rule.

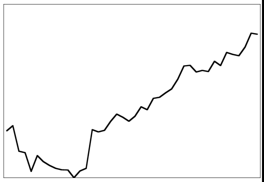
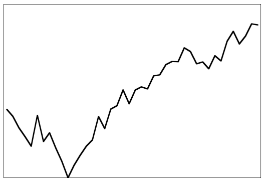
ID	flags	CPU %	CPU (MHz)	memory %	memory (kB)	algorithms
1	{-R, -H}		600		523760	{neural net, back-propagation}
2	{-H}	40, 52, 67, 80, ...	600	10, 26, 46, 69, 86, ...	523760	{C4.5}
3	{-U, -R}	10, 39, 87, 96, ...	300	19, 50, 82, 80, 70 ...	260592	{naive bayes}
...	...	...	...	...	...	...

Figure 1: A sample of a data set containing complex instances. Here, sequential values are represented in a graphical manner only for the first row to save space.

Internal Representation of Items. The attribute value format of the data set is common among data mining tools. In order to mine association rules from this data it is translated into an item representation. Each possible attribute value pair that exists in the data set is defined as an item and given an integer number called the item number. In ASAS there are two kinds of items. Regular items are formed from attribute value pairs where the value is not a sequence or an event set. The regular items are uniquely identified by their integer number. Event items are formed from attributes whose values are event sets. While event items are given an item number it is still necessary to interpret the item's value during the association rule mining process.

Handling Set Attributes. We use the extension of the Apriori algorithm to mine association rules from set valued data described in [27]. This extension was added to the Weka system.

Handling Event Attributes. Event items are interpreted rather than being treated as a literal. That is the value of an event item has meaning during the mining process. For instance, we are not interested in the occurrence of a CPU-Increase event at absolute time [13, 18], but rather on the fact that this event occurred in a relative temporal position with respect to other events. For this purpose, ASAS uses a relative time line. The relative time line is denoted by $t_0, t_1, t_2, \ldots$. There are no units implied. Each point on the relative time line corresponds to the begin times or end times of one or more events in the item set.

When an event is added to an item set the item set's relative time line must be updated. Let's add the event Memory Sustain with real times [2,35] to the item set shown below. Both the real time and relative times for the events in the item set are shown.

real times: {Disk Increase [5,25], CPU Increase [10,40]}
relative times:
{Disk Increase [t_0,t_2], CPU Increase [t_1,t_3]}
+ Memory Sustain [2,35] =
relative times:
{Disk Increase [t_1,t_3], CPU Increase [t_2,t_5], Memory Sustain [t_0,t_4]}

As you can see simply sorting the real time values and numbering them starting from 0 yields the relative times.

Level 1 Candidate Generation. In Apriori the first level of candidate item sets is generated by simply creating an item set of size one for each of the items appearing in the data set. These item sets are added to a list of candidate item sets. This basic process is used in ASAS.

Regular items are treated the same. A candidate item set is not generated for each event item. Event items have a type, usually a name that combines the original time sequence attribute name and the event detected in the sequence. There are usually multiple event items of a single type in a data set. Each event item may have a unique begin and end time. If a candidate item set were created with these event items with unique begin and end times they would be too specific and a frequent item set containing them would not likely be found. Instead, a representative event item for each event type is created. This new item is added to a candidate item set.

Level 1 Counting Support. The support of an item set is the percentage of instances in the data set that contain the item set. The weight of an item set is the number of instances that contain it. The weight of an item set is counted and then used to calculate support.

In Apriori determining if an item set is included in an instance is a straight forward procedure. The list of items in the item set is compared to the list of items in each data set instance. If all items in the item set are found in the instance that instance counts towards the weight of the item set.

For regular items the Apriori algorithm remains the same. Event items require special attention. During the counting of support the value, that is the name of the event and the time during which it occurs, must be evaluated instead of relying solely on the integer representation used by Apriori.

Determining if an item set is included in a data set instance in ASAS is not as straightforward. In an instance there may be many event items of one type. The event items in the instance that match the event items in the item set must be chosen so the temporal relationships between them are the same. When an item set containing event items is included in an instance their exists a mapping between the matching event items from the item set to the instance.

Mapping event items is trivial when the item set contains a single event item. This is because the relative begin and end times are always t_0 and t_1, respectively. As long as the instance contains an event item of the same type a mapping is guaranteed. This is the case for all item sets of size one containing one event item.

Level 2 Candidate Generation. In Apriori, candidate item sets of size two are generated by combining each pair of frequent item sets of size one. Combining item sets of size one is a simple matter since each pair results in a valid candidate. The same is true for ASAS. The difference is that for each pair of event items there exists thirteen different item sets that represent the different temporal relationships [7] those two event items can have together.

Level 2 (and up) Counting Support. First, the instance must contain all the regular items in the item set. If they are all present, inclusion of the event items are checked. If the item set contains only one event item the checking is trivial as described above in Level 1 Counting Support. If the one event item is found, the weight of the item set is incremented.

If the candidate item set contains two event items (or more in subsequent levels) then a mapping must be made from event items in the item set and event items in the instance. The relative begin and end times of event items in an item set and an instance are stored for quick reference in an array. Each index of the array represents that relative time along the time line. In each array cell is a list of begin and end labels for each event with that corresponding begin and end time. A mapping between relative times in the item set and in the instance represents a partial possible mapping between the begin and end labels of the event items, and hence a mapping between the event items themselves.

A mapping is made between an item set's relative times and an instance's real times by starting at time t_0 on the item set's time line. Starting at t=0 in the instance, the earliest time index containing a list of begin labels for the same type of events as those in the item set is found. Once this corresponding time index is found it is noted as a possible map for that relative time in the item set. The remaining time indexes in the item set are mapped in the same way, always starting at the next relative time in the instance.

Once a possible map is created it is tested using the end time labels. Begin and end time labels are paired together, belonging to a single event item in the item set. The corresponding begin and end labels in the instance must belong to a single event item as well. Furthermore, an event item in the instance can only be mapped to a single event item in the item set as the begin and end labels must.

Once the map is validated the weight of the item set is incremented. If the map is found to be invalid the last relative time used in the instance is skipped and another possible map of relative times is created and tested. This continues until there are no more relative times left to try in the instance.

Level 3 (and up) Candidate Generation. At the third level of Apriori's candidate item set generation, generating possible frequent item sets of size 3, there are ways to eliminate some of the candidates before counting support. The items in an item set are sorted in ascending order according to their item number. To combine two item sets to form a new item set one size larger there are a list of conditions that must be met. They must have the same number of items. The list of items in both item sets must be the same except for the last item which must be different. Furthermore, the last item in the second item set must have a higher item number than the last item in the first item set. If two item sets do not meet these criteria they are not combined to generate a candidate item set. The candidate item set formed by combining two item sets simply has the superset of the items in each. This is easily done by adding the last item of the second item set to the first item set.

Combining item sets has been modified for ASAS. All of the conditions still hold for regular items. The event items have to be handled as a special case. Since there is the possibility of there being more than one temporal relationship between the same set of event items the item number alone cannot be used to decide if two item sets should not be joined. Rather, a mapping must exist from all the event items in the first item set to all the event items in the second item set. This excludes event items that are the last item in an item set. Once this mapping is found the item sets can be combined.

As for level two candidate generation, it is possible for more than one candidate item set to be generated for each pair of frequent item sets that are combined. The algorithm for generating these possibilities is more involved than the straight forward iteration of thirteen possible temporal relationships. When more than two event items are present the number of temporal combinations grows. This number is limited during candidate generation by using the relative temporal information contained in each item set.

A: { Disk Increase [t_0,t_2], CPU Increase [t_1,t_3] }
B: { Disk Increase [t_1,t_2], Memory Sustain [t_0,t_3] }

Consider combining the item sets A and B shown above. These can be combined since they have the same number of items, a mapping exists between the Disk Increase event in item set A and the Disk Increase event in item set B, and the event items listed last in A and B are different. The temporal relationship between the CPU Increase event in A and the Memory Sustain event in B is not known. Some of the possible relationships can be eliminated by inferring information from the fact that the Disk Increase event in both A and B is the same event.

Combining the two item sets is done by adding the last item from the first item set to the second item set. In this example the event item Memory Sustain will be added to item set A. First, the begin time of Memory Sustain event to be added to item set A must be determined. In item set B the Memory Sustain event began before the Disk Increase event began. This temporal relationship must hold for the item sets being generated. In the item set A there is nothing before the begin time of Disk Increase. Therefore the only relative time Memory Sustain can begin is one time line value before time 0.

Next the end time of the Memory Sustain event must be determined. From item set B it can be seen that the Disk Increase event ended before the Memory Sustain event ended. This means that each relative time in the item set A that occurs after the end of the Disk Increase event is potentially when the end of the Memory Sustain event occurs. This includes the times noted in the time line for A and each time in between those marked times.

The possible begin and end time pairs determined are in relation to item set A's existing relative time line. A candidate item set is generated for each pair. In these candidate item sets the relative time line will be renumbered starting from 0 to include the Memory Sustain event.

{Disk Increase [t_1,t_3], CPU Increase [t_2,t_5], Memory Sustain [t_0,t_4]}
{Disk Increase [t_1,t_3], CPU Increase [t_2,t_4], Memory Sustain [t_0,t_4]}
{Disk Increase [t_1,t_3], CPU Increase [t_2,t_4], Memory Sustain [t_0,t_5]}

By inferring temporal relationships from the event items in common between the item sets that are being combined many candidate item sets representing different temporal relationships can be eliminated. An important observation to make is that this reasoning is only necessary when each item set has more than one event item. If each item set contains exactly one event item each the generation of candidates follows the procedure described in Section 3 Level 2 Candidate Generation.

Algorithm

```
given a data set of instances $DS$, and minimum weight $minW$
for all regular items $i$ in $DS$ do
    create candidate item set $c$ of size $k = 1$
    add $i$ to $c$ and add $c$ to candidate list $C$
end for
for all event items $e$ in data set do
    if event type of $e$ not present in event type list $ET$ then
        create candidate item set $c$ of size $k = 1$
        create a new event item $ei$ with event type of $e$ and begin time = 0 and end time = 1
        add $ei$ to $c$, add $c$ to $C$, and add $e$ to $ET$
    end if
end for
for all instances $I$ in $DS$ do
    for all $c$ in $C$ do
        if $I$ contains the item in $c$ then
            increment weight of $c$
        end if
    end for
end for
for all $c$ in $C$ do
    if weight of $c \geq minW$ then
        add $c$ to frequent item sets of size $k$ list
    end if
end for
remove all from $C$
while frequent item set of size $k$ list is not empty do
    k++
    for all pairs of item sets $f1$ and $f2$ in the frequent item sets of size $k$-1 list do
        if $f1$ and $f2$ contain event items then
            generate 13 candidates, 1 for each possible temporal relationship between the event items $f1$ and $f2$ do not have in common
        else
            generate 1 candidate by combining $f1$ and $f2$
        end if
        add generated candidate(s) to $C$
    end for
    for all instances $I$ in $DS$ do
        for all $c$ in $C$ do
            if all regular items $i$ in $c$ are included in $I$ then
                if mapping exists between all event items $ei$ in $c$ to event items in $I$ such that all temporal relationships are the same then
                    increment weight of $c$
                end if
            end if
        end for
    end for
    for all $c$ in $C$ do
        if weight of $c \geq minW$ then
            add $c$ to frequent item sets of size $k$ list
        end if
    end for
    remove all from $C$
end while
```

4. ASAS DETAILS

Some details of ASAS were left out of the basic algorithm explantation for clarity. Here are some important details, enhancement features and performance improvements over the algorithm presented in Section 3.

Multiple Events Of The Same Type. The items in the candidate item sets of size one generated in the first level of ASAS (see Section 3) are the only items that will appear in the frequent item sets and association rules generated from them. If multiple event items of the same type are to appear in a rule they must be present in the first level of candidate item sets. This requirement changes the Basic ASAS algorithm described in Section 3. Instead of creating one event item for each event type multiple event items of the same type are created.

Instead of just: item # 1. Memory Sustain [t_0,t_1]
we create:
item # 1. Memory Sustain [t_0,t_1]
item # 2. Memory Sustain [t_0,t_1]
so that later when item sets containing items 1 and 2 are combined we can get:
Memory Sustain [t_0,t_1], Memory Sustain [t_2,t_3]
Memory Sustain [t_0,t_2], Memory Sustain [t_1,t_3]
...

The number of event items of each event type are counted. For each event type a number of new event items are created. This number is the maximum number of event items of the same type specified by the user or the number of event items found in the data set of that type, whichever is smaller.

The more event items of the same type allowed, the more event items there are to count support for and use to generate new candidate item sets. The higher the maximum is set the more work there is to mine impacting performance negatively. The higher the maximum is set the more potentially expressive the rules found could be.

Duplicate Item Sets. To mitigate the work imposed by enabling the discovery of rules containing multiple events of the same type, ASAS modifies how support is counted. Instead of one list of candidate item sets there are two. The count–candidates list holds the item sets that will be counted in the data set. The all–candidates list holds all item sets including those that do not have to be counted.

An item set is considered a duplicate of another when both contain the same regular items and each event item has a corresponding event item of the same type with the same begin and end times. It is trivial to identify these duplicate item sets when generating candidates of size one. When a duplicate item set is generated it is not added to the count–candidates list. It gets added to the all–candidates list. A hash table stores the map of duplicate item sets in the all–candidates list to item sets in the count–candidates list.

Only the item sets in the count–candidates list are compared to the instances in the data set. After the support counting process is done the duplicate item sets in all–candidates are assigned the weight of their corresponding item set in the count–candidates list.

Calculating Confidence. Traditionally confidence is defined for a rule A ⇒ B as the percentage of instances that contain A that also contain B. This is usually calculated as S(AB) / S(A), where S is support.

Take a data set that has one time sequence; $<a,b,a,a,a,a>$. Consider a rule A ⇒ B where both A and B contain one event item each, $a[t_0,t_1]$ and $b[t_2,t_3]$ respectively. The event item a begins at time 0 and ends at time 1. The event item b begins at time 2 and ends at time 3. Since there is one instance in our data set and it contains the item set {A, B} as described, the support of the item sets {A}, {B}, and {A, B} are 1. If support was used to calculate the confidence of the rule A ⇒ B it would be 1. This implies that in the data set from which the rule was mined that 100 % of the time a appears, b follows. Looking at the time sequence, only 20 percent of the time is a followed by b.

Let's use event weight to define confidence. Event weight is a count of how many times the events in an item set appear in all the instances in which the item set has been found. It is possible that the pattern described by an item set's events occur multiple times inside a single instance.

Confidence of a rule containing event items in both the antecedent and consequent of the rule is defined here as EW(A⇒B) / EW(A), where EW is event weight and EW(A⇒B) is the number of occurrences of A which can be extended to {A, B}. The event weight of the item set {A} is 5. Of those 5 only 1 can be extended to {A, B}. The confidence for A ⇒ B is 0.2, or 20 percent. Clearly this more accurately represents the data set.

Since event weight is counted in relation to the antecedent of a rule being extended to encompass the consequent of a rule, the event weight counting is done during rule generation and not frequent item set mining.

5. EMPIRICAL EVALUATION

Stock Market Data. The data used consists of ten years worth of closing prices from 7 technology companies from 1992 to 2002 obtained from Yahoo! Finance. Additionally, events such as new product releases, awards received, negative press releases, and expansions or mergers from each company were obtained from each respective company's web site. Each instance in this data set represents a single quarter year. There are 24 instances in the data set. All 10 years are not represented because information on the additional events listed above were not available for all years.

Before mining, the sequences of closing prices for a quarter for each company are filtered for events. For each predefined event and closing price sequence, a new attribute is created indexing where this event type occurs in the sequence. The closing price sequence attributes are then removed from the data set. The financial events detected include rounded top, selling climax, ascending triangle, broadening top, descending triangle, double bottom, double top, head & shoulders, inverse head & shoulders, panic reversal, rounded bottom, triple bottom, triple top, sustain, increase, and decrease [21]. This data was compiled by [18].

5.1 Rules

The associations rules presented here are in the form $A \Rightarrow B$, where A and B are each a set of items. Furthermore, events are annotated with their begin (t_1), and end (t_2) times as such: $[t_1,t_2]$. This is followed by the confidence, support, and event weight of the rule. The minimum and maximum possible length of each event type appearing in the rule is specified. These rules were obtained by applying AprioriSetsAndSequences to the data set described in Section 5. Let's look at a fairly simple rule containing events.

CSCO Increase $[t_0,t_1]$ ⇒ CSCO Sustain $[t_2,t_3]$
[Conf: 0.8, Sup: 0.42, Event Weight: 16]

CSCO Increase 6-11 days, CSCO Sustain 6-11 days

This rule reads: The closing stock price of Cisco Systems Inc increased in value for 6 to 11 days. With a confidence of 80% the closing price of Cisco will remain fairly constant for 6 to 11 days sometime after Cisco's closing price stops increasing. It will do this during the same quarter year the increase took place. There is no overlap in time between the two events of this rule. With a support of about 42% this happens in about 10 of the quarters represented in the data set. In these 10 instances this behavior is found 16 times as noted by the event weight. Let's look at a rule similar to the one described above.

NXTL Selling Climax $[t_0,t_1]$ ⇒ NXTL Sustain $[t_2,t_3]$
[Conf: 0.8, Sup: 0.67, Event Weight: 24]

NXTL Selling Climax 6-12 days, NXTL Sustain 6-12 days

Here the company of interest is Nextel Communications Inc. The relationship between the events are the same. The support is higher with this rule being found in 16 quarters. In the 16 instances representing those quarters this rule is found 24 times. This rule and the previous one are examples of predicting future values in a single sequence. This is just one form of the rules ASAS can find.

Diagnostic rules concerned with a single sequence can also be found. Rather than predicting an event to begin or end after the events in the antecedent, a diagnostic rule describes an association of events beginning or ending before those in the antecedent. This includes events in the consequent which occur between or during events in the antecedent.

INTC Increase $[t_2,t_3]$ ⇒ INTC Selling Climax $[t_0,t_1]$
[Conf: 0.87, Sup: 0.46, Event Weight: 13]

INTC Increase 6-8 days, INTC Selling Climax 6-13 days

Intel Corporation's closing stock price increases for 6 to 8 days. Before Intel's stock price increases it goes through a selling climax that lasts 6 to 13 days during the same quarter. Let's look at a pair of rules. These have the same events in them but one has a predictive form and the other has a diagnostic form.

CSCO Expand Merge [t_4,t_5]
& AMD Ascending Triangle [t_0,t_1] ⇒ SUNW Sustain [t_2,t_3]
[Conf: 0.91, Sup: 0.42, Event Weight: 10]

AMD Ascending Triangle [t_0,t_1] & SUNW Sustain [t_2,t_3]
⇒ CSCO Expand Merge [t_4,t_5]
[Conf: 1.0, Sup: 0.42, Event Weight: 11]

CSCO Expand Merge 1-7 days,
AMD Ascending Triangle 6-30 days, SUNW Sustain 6-13 days

Advanced Micro Devices Inc's closing stock prices exhibits a pattern known as an ascending triangle for 6 to 30 days. Sometime after but during the same quarter Sun Microsystems Inc's closing stock price remains fairly constant for 6 to 13 days. Sometime after in the same quarter Cisco goes through a period of expansion or merger for 1 to 7 days. The predictive form of the rule has a 100% confidence. In any quarter in the data set, every time AMD and Sun exhibit the behaviors described in these rules, Cisco expands or merges.

This example shows rules where events identified in different numeric sequences and symbolic events are related in time. The temporal relationships effect the confidence of the rules, even when they have the same events. This example also shows ASAS can find diagnostic forms of association rules.

AMD Sustain [t_0,t_2] & INTC Sustain [t_1,t_3]
⇒ MSFT Sustain [t_4,t_5]
[Conf: 0.78, Sup: 0.42, Event Weight: 7]

AMD Sustain 6-9 days, INTC Sustain 6-10 days,
MSFT Sustain 6-13 days

AMD's closing stock price remains fairly constant for 6 to 9 days. During this 6 to 9 days interval Intel Corporation's closing stock price begins to remain fairly constant. Intel's sustain period last for 6 to 10 days. During this 6 to 10 day period AMD's sustain ends. Sometime after Intel's sustain period ends but in the same quarter year, Microsoft Corporation's closing stock price remains fairly constant for 6 to 13 days.

Events overlapping in time make the temporal relationships between events more precise and may make it more interesting. These rules can be found by ASAS in predictive and diagnostic forms.

An interesting characteristic of some of the rules found in our dataset with overlapping events was that they described the events themselves in terms of other events. A company's closing stock price could exhibit a complicated behavior like a selling climax. A portion of this complicated behavior may be similar to a much simpler behavior such as a sustain in stock closing price. ASAS can find rules that state a sustain event and selling climax event can overlap in the same sequence as illustrated in the rule below.

AMD Selling Climax [t_0,t_2] ⇒ AMD Sustain [t_1,t_3]
[Conf: 0.61, Sup: 0.42, Event Weight: 14]

AMD Selling Climax 6-15 days, AMD Sustain 6-9 days

Parts of complicated events can also be found similar.

SUNW Inverse Head & Shoulders [t_1,t_3]
⇒ SUNW Descending Triangle [t_0,t_2]
[Conf: 0.71, Sup: 0.42, Event Weight: 12]

SUNW Inverse Head & Shoulders 6-14 days, SUNW Descending Triangle 6-24 days

5.1.1 Multiple Occurrences of Event Type

AprioriSetsAndSequences algorithm allows rules to be found containing multiple occurrences of events with the same type.

CSCO Sustain [t_0,t_1] ⇒ CSCO Sustain [t_2,t_3]
[Conf: 0.81, Sup: 0.54, Event Weight: 38]

CSCO Sustain 6-11 days

Cisco's closing stock price remains fairly constant for 6 to 11 days. With a confidence of 81% we can say later in the same quarter Cisco's value will experience another 6 to 11 days period of sustain. It is interesting to compare rules which predict the same repeating patterns for different companies. Below are the confidence, support and event weight for repeating sustain rules of the noted companies.

SUNW [Conf: 0.87, Sup: 0.58, Event Weight: 39]
MSFT [Conf: 0.86, Sup: 0.63, Event Weight: 36]
NXTL [Conf: 0.83, Sup: 0.63, Event Weight: 40]
AMD [Conf: 0.81, Sup: 0.54, Event Weight: 35]

Let's compare two rules in which one an event repeats twice and the other the same event repeats three times.

INTC Sustain [t_0,t_1] ⇒ INTC Sustain [t_2,t_3]
[Conf: 0.92, Sup: 0.63, Event Weight: 44]

INTC Sustain [t_0,t_1] ⇒
INTC Sustain [t_2,t_3] & INTC Sustain [t_4,t_5]
[Conf: 0.71, Sup: 0.42, Event Weight: 34]

INTC Sustain 6-10 days

Notice the reduced confidence in predicting Intel's stock price will undergo a sustain period twice more in the same quarter. Repeating events also occur in events that were not obtained from a sequence. The rule below shows a repeating Cisco Expand Merge event.

CSCO Expand Merge [t_0,t_1]
⇒ CSCO Expand Merge [t_2,t_3]
[Conf: 0.9, Sup: 0.46, Event Weight: 36]

CSCO Expand Merge 1-7 days

Rules with repeating events are not limited to just one type of event. Below is an example of a rule containing both a singular event and a repeating event.

CSCO Expand Merge [t_4,t_5] & NXTL Sustain [t_0,t_1]
⇒ NXTL Sustain [t_2,t_3]
[Conf: 0.92, Sup: 0.42, Event Weight: 24]

NXTL Sustain [t_2,t_3] & NXTL Sustain [t_0,t_1]
⇒ CSCO Expand Merge [t_4,t_5]
[Conf: 0.67, Sup: 0.42, Event Weight: 16]

CSCO Expand Merge 1-7 days, NXTL Sustain 6-12 days

As shown above rules with repeating events can be found in predictive and diagnostic forms.

5.2 ASAS Performance

Figure 2 shows the seconds used to mine rules per frequent item set found and other metrics for slightly differing data sets from the stock market domain. The total time it takes to mine appears to be insensitive to the number of event attributes, the number of event occurrences, and the average length of the time line. It seems only the number of frequent item sets found in a data set greatly increases mining time. The time spent finding each frequent item set seems related to the number of event occurrences and the number of event attributes in the data set.

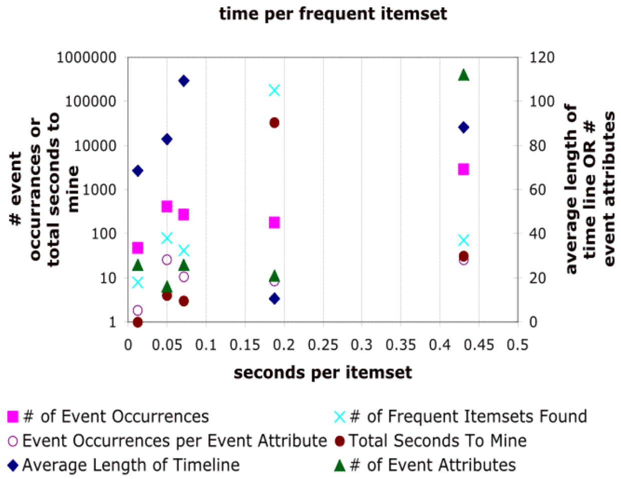


Figure 2: Various Metrics

Figure 3 shows the results of varying the maximum number of events with the same type that can appear in a rule. This was done with a support setting of 49%. 16 rules containing 2 events of the same type were found. Beyond a maximum of 2 more time is spent per frequent item set with no additional rules found to justify the cost. The lower the percentage of new rules found by increasing the maximum number of events of the same type allowed, the more time per frequent item set will be spent during mining. Figure 4 shows results using a support of 40%. This tradeoff is easily seen. Even though more rules are found due to the lower support, more time is spent per frequent item set.

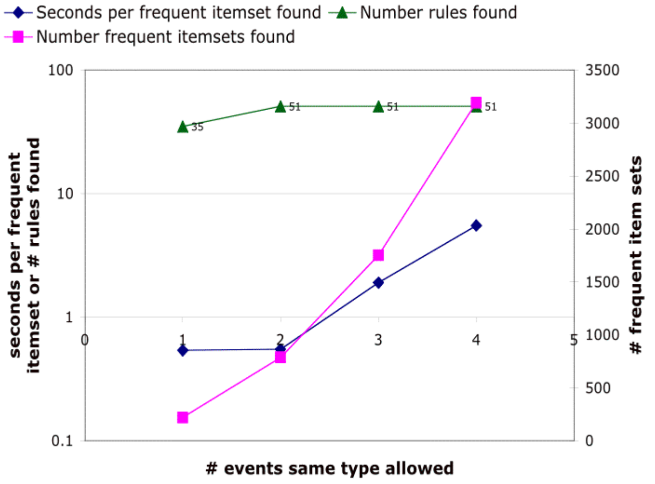


Figure 3: 49 Percent Support

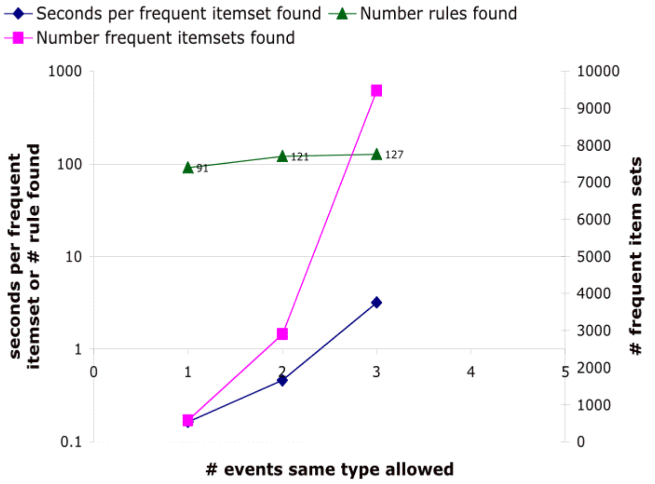


Figure 4: 40 Percent Support

6. RELATED WORK

There has been a great deal of interest in devising approaches to mine associations from sequential data. These approaches can be roughly divided into two groups. The first group contains approaches that extend the Apriori algorithm to sequences. These approaches assume data instances that are sequences of commercial transactions. A commercial transaction is called an *event*. These approaches mine frequent patterns from those data instances. Among others, the work by Srikant and Agrawal [6] and by Zaki [33] and collaborators belong to this group. They use the notions of *time window* and *max/min gaps* to address the complexity of the mining task. Zaki [33] considers item set constraints for this same purpose. One difference between our work and the approaches in this group is that our notion of *event* is a non–trivial time interval and theirs is a point in time (instantaneous events). This has a profound impact on the expressiveness of our association rules and on the complexity of the mining process, as in our case the possible orderings of two single events is 13 while for them that number of orderings in only 3. Another important difference is that in our approach we consider data instances that are combinations of several attribute types, while their instances are sequences of transactions.

The second group of association rule mining approaches to sequential mining includes the work by Mannila et al. [24, 23, 13]. They consider *episodes* of events, where events are once again points in time. Episodes are collections of partially ordered events that occur close to each other in time. This constraint addresses the complexity of the search in a way similar to the time window approach described above. Our work extends theirs by allowing events that are time intervals. This enhances the collection of partial orders that are applicable to a set of events and thus the expressiveness of the mined patterns.

Roddick and Spiliopoulou [26] provide an excellent survey of temporal knowledge discovery. Rainsford and Roddick [25] report efforts on extending association rules with temporal information. Their work is similar to ours in that they also consider the 13 possible ways in which two temporal events can be ordered in time. However, the expressiveness of their association rules is very restricted in comparison with ours. Bettini et al. [10] describe an approach to mine temporal associations that allows the user to define a rule template, and their algorithm finds valid instantiations of the rule template in the data set. Our approach is more general than theirs in that the user is not restricted to use just one temporal template for each mining task, as our algorithm considers all possible temporal patterns that are frequent. Also, we can explore several time–granularities during the same mining task, just by defining an event–based attribute for each relevant time–granularity and letting them "intersect" with other events of interest. The events in these time–granularity attributes would simply define intervals of the time granularity it represents. For a real time line whose units are days the time–granularity attribute for weeks would identify an event for every seven days along the real time line. Other approaches that employ user–defined temporal templates are those described by Han and collaborators [29, 22]. Their multidimensional intertransaction association rules are particular cases of our complex temporal association rules.

7. CONCLUSIONS AND FUTURE WORK

We introduce an algorithm for mining expressive temporal relationships from complex data sets in which a single data instance may consist of a combination of attribute values that are nominal sequences, time series, sets, and traditional relational values. Our mining algorithm is close in spirit to the two-stage Apriori algorithm. Our work contributes the the investigation of prune strategies and efficient data structures to effectively handle the added data complexity and the added expressiveness of the temporal patterns.

Several alternative methods to Apriori's item set generation have been proposed in the literature. Those alternative methods differ from Apriori in their strategy to generate *frequent* item sets, or in the item sets that they consider *interesting.* Some approaches attempt to increase the performance of the mining algorithm over certain types of data [35]; compute frequent item sets very efficiently [16, 32, 20, 17]; or utilize parallel computing environments [5, 34]. Work by Webb [30], Agarwal, Aggarwal, and Prasad [1, 2], and Bayardo, Agrawal, and Gunopoulos [8] address some of the efficiency concerns by means of novel and judicious search techniques for item set generation. Modifications and generalizations of association rules that are more suitable to certain application domains are investigated by Brin, Motwani, and Silverstein [11] and by Cohen et al. [12]. Additional work on using statistical measures of item set interestingness other than support and lift are investigated by DuMouchel and Pregibon [14] and by Wu, Barbará and Ye [31].

In this paper we focus on the necessary extensions of the traditional support and confidence framework to handle the desired complex associations. Zhen, Kohavi, and Mason [36] show experimentally that although some well–known alternative association rule mining approaches discussed above outperform Apriori over artificial data sets, they do not over real-world data sets. The work described here provides a foundation for future investigation and comparison of alternative measures of item set interestingness and alternative search techniques such as those discussed above but in the context of complex data.

8. REFERENCES

[1] R. C. Agarwal, C. C. Aggarwal, and V. Prasad. Depth first generation of long patterns. In *Proc. of the Sixth ACM SIGKDD Conference on Knowledge Discover y and Data Mining*, pages 108–118, Boston, MA, Aug. 2000. ACM.

[2] R. C. Agarwal, C. C. Aggarwal, and V. V. V. Prasad. A tree projection algorithm for generation of frequent item sets. *Journal of Parallel and Distributed Computing*, 61(3):350–371, 2001.

[3] R. Agrawal, C. Faloutsos, and A. Swami. Efficient similarity search in sequence databases. In *Foundations of Data Organization and Algorithms (FODO) Conference*, Evanston, Illinois, Oct. 1993.

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

[5] R. Agrawal and J. C. Shafer. Parallel mining of association rules. *IEEE Trans. On Knowledge And Data Engineering*, 8:962–969, 1996.

[6] R. Agrawal and R. Srikant. Mining sequential patterns. In *International Conference on Database Engineering*, pages 3–14. ieee, 1995.

[7] J. Allen. Maintaining knowledge about temporal intervals. *Communications of the ACM*, 26(11), 1983.

[8] R. J. Bayardo, R. Agrawal, and D. Gunopulos. Constraint-based rule mining in large, dense databases. *Data Mining and Knowledge Discovery*, 4(2/3):217–240, July 2000.

[9] D. Berndt and J. Clifford. Finding patterns in time series: a dynamic programming approach. In U. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, editors, *Advances in Knowledge Discovery and Data Mining*, pages 229–248. AAAI Press/ MIT Press, 1995.

[10] C. Bettini, X. Sean Wang, and S. Jajodia. Testing complex temporal relationships involving multiple granularities and its application to data mining. In ACM, editor, *Proceedings of the Fifteenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, PODS 1996, Montréal, Canada, June 3–5, 1996*, volume 15, pages 68–78, New York, NY 10036, USA, 1996. ACM Press.

[11] S. Brin, R. Motwani, and C. Silverstein. Beyond market baskets: Generalizing association rules to dependence rules. *Data Mining and Knowledge Discovery*, 2:39–68, 1998.

[12] E. Cohen, M. Datar, S. Fujiwara, A. Gionis, P. Indyk, R. Motwani, J. D. Ullman, and C. Yang. Finding interesting associations without support pruning. In *Proc. of Int'l. Conf. on Data Engineering (ICDE2000)*, pages 489–499, 2000.

[13] G. Das, K.-I. Lin, H. Mannila, G. Renganathan, and

P. Smyth. Rule discovery from time series. In *Proc. of the 4th Int. Conf. on Knowledge Discovery and Data Mining*, pages 16–22. ACM, 1998.

[14] W. DuMouchel and D. Pregibon. Empirical bayes screening for multi-item associations. In *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 67–76. ACM Press, 2001.

[15] M. R. Faloutsos and Y. Manolopoulos. Fast subsequence matching in time-series databases. In *Proc. ACM SIGMOD*, pages 419–429, Minneapolis, MN, May 1994.

[16] J. Han, J. Pei, and Y. Yin. Mining frequent patterns without candidate generation. In *Proc. Int'l. Conf. of Management of Data (SIGMOD2000)*, pages 1–12, May 2000.

[17] C. Hidber. Online association rule mining. Technical Report CSD-98-1004, Dept. of Electrical Engineering and Computer Science, Univ. of California at Berkeley, May 1998.

[18] S. Holmes and C. Leung. Exploring temporal associations in the stock market. Undergraduate Graduation Project (MQP). Worcester Polytechnic Institute, April 2003.

[19] P. Laxminarayan. Exploratory analysis of sleep data. Master's thesis, Department of Computer Science, Worcester Polytechnic Institute, January 2004.

[20] D.-I. Lin. *Fast Algorithms for Discovering the Maximum Frequent Set*. PhD thesis, Dept. of Computer Science. New York University, 1998.

[21] J. Little and L. Rhodes. *Understanding Wall Street*. Liberty Hall Press and McGraw-Hill Trade, 3rd edition, 1991.

[22] H. Lu, L. Feng, and J. Han. Beyond intratransaction association analysis: mining multidimensional intertransaction association rules. *ACM Transactions on Information Systems (TOIS)*, 18(4):423–454, 2000.

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

[24] H. Mannila, H. Toivonen, and A. I. Verkamo. Discovering Frequent Episodes in Sequences. In U. M. Fayyad and R. Uthurusamy, editors, *Proceedings of the First International Conference on Knowledge Discovery and Data Mining (KDD-95)*, Montreal, Canada, 1995. AAAI Press.

[25] C. Rainsford and J. Roddick. Adding temporal semantics to association rules. In *Proceedings of the Third European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD'99)*, pages 504–509, 1999.

[26] J. Roddick and M. Spiliopoulou. A survey of temporal knowledge discovery paradigms and methods. *IEEE Transactions on Knowledge and Data Engineering*, 14(4):750–767, 2002.

[27] C. Shoemaker and C. Ruiz. Association rule mining algorithms for set-valued data. In J. Liu, Y. Cheung, and H. Yin, editors, *Proc. of the Fourth International Conference on Intelligent Data Engineering and Automated Learning*, pages 669–676. Lecture Notes in Computer Science. Vol. 2690. Springer-Verlag, June 2003.

[28] R. Srikant and R. Agrawal. Mining sequential patterns: Generalizations and performance improvements. In *Proc. of the Fifth Int'l Conference on Extending Database Technology (EDBT)*, Avignon, France, March 1996.

[29] A. K. Tung, H. Lu, J. Han, and L. Feng. Efficient mining of intertransaction association rules. *IEEE Transactions On Knowledge And Data Engineering*, pages 43–56, Jan./Feb. 2003.

[30] G. I. Webb. Efficient search for association rules. In *Proc. of the Sixth ACM SIGKDD Conference on Knowledge Discover y and Data Mining*, pages 99–107, Boston, MA, Aug. 2000. ACM.

[31] X. Wu, D. Barbará, and Y. Ye. Screening and interpreting multi-item associations based on log-linear modeling. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 276–285. ACM Press, 2003.

[32] Y. Xiao and M. H. Dunham. Considering main memory in mining association rules. In *Proceedings of the First International Conference on Data Warehousing and Knowledge Discovery*, pages 209–218, November 1999.

[33] M. Zaki. Sequence mining in categorical domains: Incorporating constraints. In *Proc. Int. Conference on Information and Knowledge Management (CIKM)*, pages 422–429. ACM, 2000.

[34] M. J. Zaki, M. Ogihara, S. Parthasarathy, and W. Li. Parallel data mining for association rules on shared-memory multi-processors. In *CD-ROM Proceedings of Supercomputing'96*, Pittsburgh, PA, Nov. 1996. IEEE.

[35] M. J. Zaki, S. Parthasarathy, M. Ogihara, and W. Li. New algorithms for fast discovery of association rules. In D. Heckerman, H. Mannila, D. Pregibon, and R. Uthurusamy, editors, *Proceedings of the Third International Conference on Knowledge Discovery and Data Mining (KDD-97)*, page 283. AAAI Press, 1997.

[36] Z. Zheng, R. Kohavi, and L. Mason. Real world performance of association rule algorithms. In *Proceedings of the Seventh ACM-SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2001.