Procedural Steps for Knowledge Mining in Time Series

Kaustuva Chandra Dev
Dept. of MCA
Trident Academy of Creative Technology
BPUT, Odisha, India.

Sibananda Behera
Dept. of MCA
TACT, BPUT
Bhubaneswar, Odisha, India

ABSTRACT

Symbolic intervals which form temporal patterns are usually formulated through Allen's interval relations that originate in temporal reasoning. But this representation is not advantages for knowledge discovery. The Hierarchical Time series Knowledge Representation (HTKR) is the hierarchical language which expresses the temporal aspects of coincidence and partial order, for interval patterns. We present mining procedural steps which are more efficient, effective and based on item set techniques. Pruning of the search space minimizes the mining result size considerably, thereby speeding up the procedural steps and easing the interpretations. When applied on the real data set, HTKR can provide the explanation of underlying temporal phenomena, but whereas the numerous Allen's relation patterns only explains fragmented data.

General Terms

Algorithms.

Keywords

Data mining, time series, knowledge mining, temporal relations, phrases.

1. INTRODUCTION

Temporal information is related to changes and the times of the changes [13].An important data format which discovers temporal knowledge is symbolic interval time series. Various methods used for converting numerical time series to symbolic interval time series are segmentation, discretization and clustering. Patterns extracted from symbolic interval data can explain the underlying temporal behavior.

Allen's interval relations [1] forms the main basis for the unsupervised pattern discovery in interval time series. Originally theses relations were developed in the temporal reasoning context, where the incomplete exact data and the temporal constraints are usually the inputs to the process. Typical problems include answering scenario queries which satisfy all the constraints. But in the context of data mining, incorrect and noisy interval data are inputted to search for understandable and meaningful patterns. Winarko [14] proposed an algorithm named ARMADA which is based on an efficient sequential pattern mining algorithm, MEMISP [15], to mine frequent temporal patterns.

We think that Allen's relations are not fit for interval time series pattern discovery due to their severe limitations. As an alternative we proposed the Hierarchical Time series Knowledge Representation (HTKR), the hierarchical language for creation of interval times series based temporal knowledge, which extends the Unification based Temporal Grammar (UTG) [9], [10]. We present efficient procedural steps for pattern mining expressed with HTKR using item set techniques.

2. RELATED WORK AND MOTIVATION

The temporal patterns from interval data are usually framed through using the 13 interval relations of Allen [1], such as before, after, starts, startedby, meets, metby, contains, during, overlaps, overlappedby, finishedby, finishes and equals. Variants of the Apriori algorithm usually perform the unsupervised rule mining with using Allen's relations. The combination of two intervals or existing patterns along with a single relation, constructs the required interval pattern [3], [6]. All the pair wise interval relations contained within a pattern are listed through a representation [5]. Allen's relations are not advantageous for pattern discovery from interval time series, as shown by the following examples.

(i) Patterns extracted from the noisy interval data expressed through Allen's interval relations lack robustness:

Most of Allen's relations require two or more interval end points to be equal. It creates patterns where a similar relationship between intervals is fragmented into different relations due to small disturbances in interval end points. Several almost equal intervals are shown in Figure 1.

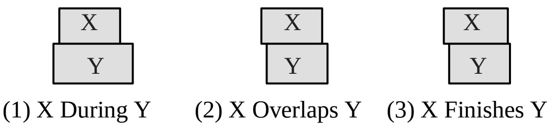


Figure 1: Different patterns which are fragments of same approximated relation almost equal.

(ii) Patterns extracted and expressed with Allen's interval relations show ambiguity:

Various different situations can be visually and intuitively represented through Allen's same relation. Different versions of overlaps relation are shown in Figure 2.

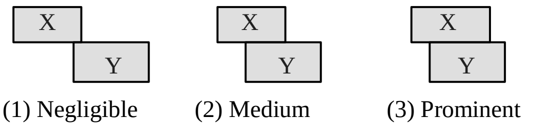


Figure 2: Different patterns of Allen's overlaps relation.

(iii) Patterns extracted and expressed with Allen's interval relations are difficult to understand:

The patterns represented through Allen's relations do not follow the suggestions of Gricean maxims representation of knowledge discovery for humans [8]. For instance, the suggestion regarding maxim of quantity is violated. To avoid ambiguity due to the compact format, the list of all intervals' pair wise relation patterns need to be expressed [5], which grows very quickly with the number of intervals.

Ultsch had proposed the UTG [9], [10], which is a hierarchical pattern language for the temporal knowledge discovery, and he had also implemented it [4]. Temporal abstraction is increased through each level of the hierarchy. Events and Sequences form the core element of the patterns. Almost simultaneous intervals are described by Events while the total orders of the events are described by Sequences. The UTG is comparatively more robust than Allen's relations, but the later are more expressible [7],[17]. The UTG's hierarchical structure offers new relevant feedback possibilities during the process of knowledge discovery and result analysis [10]. In order obtain a temporal description of the multivariate data, basic temporal abstractions (extracted from each variable in the previous step) are combined to form complex temporal patterns. For example, a domain expert may want to describe a pattern in a time series data such as: "an increase in variable X is followed by a decrease in variable Y". This is the idea behind the temporal abstraction framework of Shahar [16]. These core ideas are extended in HTKR to achieve greater robustness and ability to express.

3. REPRESENTING KNOWLEDGE IN TIME SERIES

Temporal knowledge present in the interval data is expressed through the hierarchical language HTKR. Each level describes the duration, coincidence and partial order temporal aspects successively.

Let Σ be a collection of finite set of symbols s.

Definition 1. A symbolic time interval is a set consisting of {*s*, *start, end*} where s Є Σ, pair {*start, end*} is a time interval, *start* ≤ *end*, {*start, end*} Є *T* X *T* i.e. T^2 and

$T = \{1, 2 \ldots n\}$ and $N \supset \{1, 2 \ldots n\}$.

Definition 2. Time interval {*start, end*} ⊆ {*start', end'*} if *start'* ≤ *start* and *end'* ≤ *end* and

time interval {*start, end*} = {*start', end'*} if *start'* = *start* and *end'* = *end*.

Definition 3. A symbolic interval duration is denoted as *duration* ({*start, end*}) = *end* – *start* +1.

Definition 4. Two time intervals {*start, end*}and {*start', end'* } overlap if {*start*, … *end* } ∩ {*start'*, … *end'* } ≠ ∅.

Tones basically represent duration in HTKR, which consists of a label, a symbol and a symbolic interval series. Chords are the simultaneously occurring tones, which represent coincidence.

Definition 5. A chord pattern *C* describes a time interval where *t* chords coincide, for $t > 0$.

Definition 6. Chord C_i describes a superset coincidence of tones from C_j, then $C_i \supset C_j$.

Definition 7. The merged chord of C_i and C_j is denoted as C_i U C_j.

Definition 8. The numbers of tones that coincide are $|C|$.

Definition 9. The support of a chord $Support_d(C)$ is the maximal observation intervals having minimum duration of *d*.

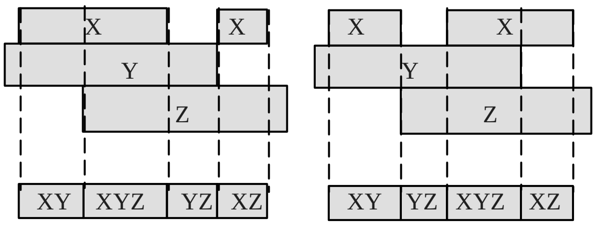
Definition 10. The chord C_i is marginally closed w.r.t. a threshold *tr* ($tr < 1$) if no super chords exist with approximately same support. $\forall C_j \supset C_i$, $[Support\ (C_j)\ /\ Support\ (C_i)] < 1 - tr$.

Phrases are formed by the collection of several chords connected through a partial order.

Definition 11. A phrase pattern is a partial order of *p* chords ($p > 1$), starting with the first chord and ending with the last chord and no overlapping is allowed within the chords of a phrase.

Definition 12. Phrase P_i describes partial order of the superset of the chords of P_j, then $P_i \supset P_j$, and the same partial order exists for all common chords.

Definition 13. The support of a phrase *Support* (*P*) is the number of observations.



(1) Phrase instance (2) Similar phrase instance

Figure 3: Summarizing overlapping tones in chords

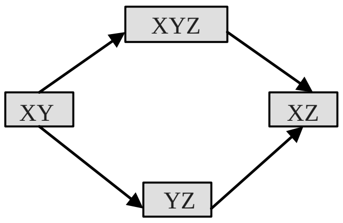


Figure 4: Partial order of chords within phrase.

4. HIERARCHICAL TIME SERIES KNOWLEDGE MINING

We discuss the procedural steps for mining the coincidence and the partial order from a given symbolic interval data.

4.1 Coincidence mining

A set of tones, forming the chords, act as input for the coincidence mining process. A chord is taken into consideration only when there is a coincidence of $Size_{min}$ different tones having a minimum duration of *d*. A chord is considered as frequent only if the total duration of all the intervals where it has been considered, is greater than the minimum support $Support_{min}$. Thus the mining process involves selection of a subset from all the tones and comparison with all the super chords to find margin closed chords. Since it is similar to the mining of closed frequent item sets, we therefore follow the CHARM [12] algorithm.

Algorithm 4.1 applies a depth first search method for finding the margin closed chords. It consists of repetitive recursive steps, involving a prefix chord $Chord_{prefix}$ and the set of super chords *Superchords*. The algorithm is initially started by taking an empty chord and all trivial frequent chords as anticipated extensions. All combinations of the super chords from *Superchords* extends the prefix chord $Chord_{prefix}$. The initial super chord extension by $Chord_i$ is stored in a variable $Chord_i$*'*. The minimum support factor filters the extensions along with another chord $Chord_j$. The idea of support comparison between the super chord ($Chord_i$*'* U $Chord_j$) and both the individual sub chords $Chord_i$*'* and $Chord_j$ forms the core calculation part of the algorithm. Its conditions are based on the generalized view of the four item set properties [12]. The addition of the current chord $Chord_i$*'* to the set of margin closed chords occurs, only if none of the previously found margin closed chords subsumes it. The recursion continues with the super chords *Superchords'* of current chord *Chord$_i$'* until the maximum chord size is reached. The process stops when set of extensions *Superchords* becomes empty.

CHARM algorithm [12] scales up linearly with the quantity of transactions, and this algorithm also follows that behavior. The minimum chord duration *d* has to be carefully chosen w.r.t. the application domain and should be long enough for the consideration of coincidence. The number of chords retrieved is directly controlled by the minimum support (*Support$_{min}$*) and the margin closedness threshold (*threshold*). Thus the upper ranges of these parameters are provided and the algorithm tries to slowly squeeze these values, if the previous found results need refinement.

Algorithm 4.1 Mining margin closed chords using DFS

```
Input
  The set of given tones Tones
  The minimum duration of the chords d
  The minimum support of the chords $Upper\_Support_{min}$
        (only upper range is provided)
  The minimum size of the chord $Size_{min}$
  The maximum size of the chord $Size_{max}$
  The margin closedness threshold Upper_threshold
        (only upper range is provided)
Output
  The set of margin closed chords Return

Algorithm
1.  $Support_{min} = Upper\_Support_{min}$
    and threshold = Upper_threshold
2.  Loop
3.     Update $Support_{min}$
4.     Loop
5.       Update threshold
6.       $Superchords = \{t \in Tones | Support_{min} \leq Support(t)\}$
7.       Return =Extension ($\emptyset$,Superchords,$\emptyset$,$Support_{min}$, threshold)
8.       If Return is already refined then
9.          Output Return and Stop
10.      End If
11.      Decrement threshold by 0.1
12.    End Loop
13.    Decrement $Support_{min}$ by 0.01
14. End Loop
Algorithm
Extension ($Chord_{prefix}$, Superchords, $Support_{min}$, threshold)
1.  Loop for each $Chord_i \in Superchords$
2.     $Chord_i' = $ Concat ($Chord_{prefix}$, $Chord_i$)
       $Superchords' = \emptyset$
3. Loop for each $Chord_j \in Superchords$ and i < j
4. If $Support_d(\text{Union}(Chord_i', Chord_j)) >= Support_{min}$
5.  If $\frac{Support_d(\text{Union}(Chord_i', Chord_j))}{\text{Max}(Support_d(Chord_i'), Support_d(Chord_j))} \geq 1-threshold$
6.          $Chord_i' = \text{Union}(Chord_i', Chord_j)$
            $Superchords = \text{Diff}(Superchords, Chord_j)$
7.  Else If $\frac{Support_d(\text{Union}(Chord_i', Chord_j))}{Support_d(Chord_i')} \geq 1-threshold$
      and $\frac{Support_d(\text{Union}(Chord_i', Chord_j))}{Support_d(Chord_j)} < 1-threshold$
8.          $Chord_i' = \text{Union}(Chord_i', Chord_j)$
9.  Else If $\frac{Support_d(\text{Union}(Chord_i', Chord_j))}{Support_d(Chord_i')} < 1-threshold$
      and $\frac{Support_d(\text{Union}(Chord_i', Chord_j))}{Support_d(Chord_j)} \geq 1-threshold$
10.       $Superchords' = \text{Union}(Superchords', \text{Union}(Chord_i', Chord_j))$
          $Superchords = \text{Diff}(Superchords, Chord_j)$
11.  Else
        $Superchords' = \text{Union}(Superchords', \text{Union}(Chord_i', Chord_j))$
12.  End If
13. End If
14. End Loop
15, If $| Chord_i' | \geq Support_{min}$
      and $\forall Chord \in Return$ with $Chord_i' \subseteq Chord$,
      $\frac{Support_d(Chord)}{Support_d(Chord_i')} < 1 - threshold$
16.   $Return = \text{Union}(Return, Chord_i')$
17. End If
18. If $| Chord_i' | < Support_{max}$ then
19. $Return = \text{Extension}(Chord_i', Superchords', Support_{min}, threshold)$
20. End If
21. End Loop.
```

4.2 Partial order mining

Phrases describe the partial order of time intervals. The steps for margin closed phrases mining are similar to the steps for closed partial order mining [2].

Algorithm 4.2 tries to find margin closed phrases. The item set interval series are first formed from the interval sequences. For each interval having minimum duration *d$_{min}$* where no changes of chords occur, it creates one item set, which contains all currently active chords' symbols. Shorter sub series are formed by creating a sequence of intervals from a single item set interval series. Next the standard closed sequence mining algorithm CLOSPAN [11], can be used with a restriction which excludes overlapping chords. This results in obtaining pairs (*CSpattern*, *TWindow*) i.e. closed sequential pattern *CSpattern* in transaction window *TWindow*. In the margin closed partial orders' mining the pairs, formed consist of set of *CSpattern* and the transaction list where they all occur, are needed to be maximal. Partial order is then converted from each group of closed sequences. Again the algorithm is controlled by the minimum duration (*d$_{min}$*) and the margin closedness threshold (*threshold*). Thus the upper ranges of these parameters are again provided and the algorithm tries to slowly squeeze these values, if the previous found results need refinement. The required conditions for the conversion of the set of sequences into partial order form the last step of the algorithm [2], which ultimately constructs the final phrases representation.

Algorithm 4.2 Finding margin closed phrases

```
Input
  A set of chords Chords
  Chords' minimum duration in phrase $Upper\_d_{min}$
        (only upper range is provided)
  The minimum support of the phrase $Support_{min}$
        (default value is 1)
  The minimum size length in phrase $Size_{min}$
  The margin closedness threshold Upper_threshold
        (only upper range is provided)
  Transaction Window
        $Window = \{(CSpattern_i, e_i) | i=1\ldots w\}$
Output
  A set of phrases
Algorithm
1.  $d_{min} = Upper\_d_{min}$ and threshold = Upper_threshold
2.  Loop
3.     Update $d_{min}$
4.     Loop
5.        Update threshold
6.        Chords conversion to item set interval series
          by using $d_{min}$
```

```
7.        Find pairs (CSpattern, TWindow), consisting
          of closed sequential patterns CSpattern
          occurring within transaction window TWindow
          ⊆ Window, using $Support_{min}$ and $Size_{min}$
8.        Creation of margin closed maximal pairs
          ($CSpattern_{max}$, TWindow),
           $CSpattern_{max}$ is set of all closed sequential
           patterns occurring in all transaction windows
           TWindow ⊆ Window, using $Support_{min}$ and
           threshold
9.        If margin closed maximal pairs are already
           refined then
10.            Building of partial order chords for each
               set $CSpattern_{max}$ and Stop
11.        End If
12.        Decrement threshold by 0.1
13.      End Loop
14.      Reduce $d_{min}$ by half
15. End Loop
```

5. DISCUSSION

All algorithms for mining Allen’s relations are based on the A-priori rule. The patterns are formed following a breadth first technique [5]. The breadth first technique is far inferior to the depth first technique, on the basis of performance measure. The margin closedness concept helps to reduce the complexity by pruning the search space. The coarse results obtained can be further refined, if required, by further continuing that loop. Phrases and sequential patterns have a great degree of structural similarities, involving highly redundant data. So this conversion to item set intervals reduces the redundancy to a great extent.

6. SUMMARY

We presented the unsupervised mining algorithms for HTKR, a temporal pattern language, which were quite efficient, effective and tailored from the item set and sequential pattern mining. The number of patterns produced was greatly reduced through the pruning of the search space, with the help of margin closedness which ultimately made the mining process faster. The mining algorithm was further enhanced by embedding the extra features of result refinements. Mining the patterns of HTKR was found to have deeper meaning, greater efficiency and higher effectiveness then Allen’s relations.

7. ACKNOWLEDGMENTS

Our thanks to all the experts who have contributed towards development of the template.

8. REFERENCES

[1] J. F. Allen. 1983 Maintaining knowledge about temporal intervals. Communications of the ACM, 26(11):832–843,

[2] G. Casas-Garriga. 2005. Summarizing sequential data with closed partial orders. In Proc. SDM’05, pages 380–391.

[3] P. R. Cohen. 2001. Fluent learning: Elucidating the structure of episodes. In Proc. IDA’01, pages 268–277.

[4] G. Guimar˜es and A. Ultsch. 1999. A method for temporal knowledge conversion. In Proc. IDA’99, pages 369–380.

[5] F. H¨ppner. 2003. Knowledge Discovery from Sequential Data. PhD thesis, Technical University Braunschweig, Germany.

[6] P.-S. Kam and A. W.-C. Fu. 2000. Discovering temporal patterns for interval-based events. In Proc. DaWaK’00, pages 317–326.

[7] F M’’rcchen. 2006. Time Series Knowledge Mining. PhD thesis, hilipps-university Marburg, Germany.

[8] S. G. Sripada, E. Reiter, and J. Hunter. 2003. Generating English summaries of time series data using the Gricean maxims. In Proc. KDD’03, pages 187–196.

[9] A. Ultsch. 1996. Eine unifikationsbasierte Grammatik zur Beschreibung von komplexen Mustern in multivariaten Zeitreihen. personal notes. German.

[10] A. Ultsch. 2004. Unification-based temporal grammar. Technical Report 37, CS Dept., Philipps-University Marburg, Germany.

[11] X. Yan, J. Han, and R. Afshar. 2003. CloSpan: Mining closed sequential patterns in large datasets. In Proc. SDM’03, pages 166–177.

[12] M. J. Zaki and C.-J. Hsiao. 2005. Efficient algorithms for mining closed itemsets and their lattice structure. IEEE TKDE, 17(4):462–478.

[13] W. Li, K. F. Wong and C. Yuan,2001. Toward automatic Chinese temporal information extraction, *JASIST*, 52(9)pages 748–762.

[14] E. Winarko and J.F Roddick. 2007. ARMADA-An algorithm for discovering richer relative temporal association rules from interval-based data. *Data & Knowledge Engineering*, vol. 3, issue 1, pp. 76-90.

[15] M. Lin and S. Lee. 2005. Fast Discovery of Sequential Patterns by Memory Indexing and Database Partitioning. *Journal of Information Sciences and Engineering*, Vol. 21, No. 1, pp. 109-128.

[16] Shahar, Y. 1997. A Framework for Knowledge-Based Temporal Abstraction. Artificial Intelligence, 90:79-133.

[17] Buono, P., Aris, A., Plaisant, C., Khella, A., Shneiderman, B., Hochheiser, H. Schneiderman, B. 2005.Interactive Pattern Search in Time Series. Proceedings of Conference on Visualizaion and Data Analysis, VDA , SPIE. CA.