ABSTRACT

Temporal Data Mining is a rapidly evolving area of re-search that is at the intersection of several disciplines, in-cluding statistics, temporal pattern recognition, temporal databases, optimisation, visualisation, high-performance com-puting, and parallel computing. This paper is first intendedto serve as an overview of the temporal data mining in re-search and applications.

INTRODUCTION

Temporal Data Mining is a rapidly evolving area of re-search that is at the intersection of several disciplines, in-cluding statistics (e.g., time series analysis), temporal pat-tern recognition, temporal databases, optimisation, visual-isation, high-performance computing, and parallel comput-ing. This paper is intended to serve as an overview of thetemporal data mining in research and applications. In addi-tion to providing a general overview, we motivate the impor-tance of temporal data mining problems within Knowledge

Discovery in Temporal Databases (KDTD) which include formulations of the basic categories of temporal data mining methods, models, techniques and some other related areas.

DEFINITION AND TASKS OF TEMPO-RAL DATA MINING

The temporal data mining component of the KDTD process is concerned with the algorithmic means by which tempo-ral patterns are extracted and enumerated from temporal

data. Some problems for temporal data mining in temporaldatabases include questions such as: How can we provideaccess to temporal data when the user does not know how to describe the goal in terms of a specific query? How canwe find all the time related information and understand a large temporal data set? and so on.

DEFINITION

Definition:

*Temporal Data Mining is a single step in the process of Knowledge Discovery in Temporal Databases that enumerates structures (temporal patterns or models)*

*over the temporal data, and any algorithm that enumerates temporal patterns from, or fits models to, temporal data is a Temporal Data Mining Algorithm.*

TEMPORAL DATA MINING ALGORITHMS

The goal of temporal data mining is to find hidden relations between given sequence of events. An efficient approach to mining such relations is sequence mining. It involves three steps:

1) Transformation: converting given data into suitable form.

2) Similarity Measure: defining the similarity measure to be used.

3) Mining Operation: applying mining operation to get desired results.

*GSP Algorithm*

GSP stands for Generalized Sequential Pattern. It is used for sequence mining. It is based on the Apriori algorithm. We first discover all the frequent items level-wise by counting the occurrences of all singleton elements in the data set. The transactions are then filtered. Non frequent items are removed. After this step, each transaction consists of only the frequent elements. This is the input to the algorithm.

GSP Algorithm makes multiple passes. In the 1st pass, all single items are counted. A set of candidate 2-sequences are formed from the frequent items, and one more pass is made to find out their frequency. Candidate 3-sequences are generated from frequent 2-sequences. This process is repeated until no more frequent sequences are found. Two main steps in the algorithm are: Candidate Generation: The candidates for the next pass are generated by joining F(k-1) with itself. Pruning is done in order to eliminate any sequence at least one of whose subsequences is not frequent. Support Counting: A hash-tree based search is used for

counting support efficiently. Non-maximal frequent sequences are removed.

Algorithm:

F1 = the set of frequent 1-sequence

k=2,

do while F(k-1)!= Null;

Generate candidate sets Ck (set of candidate ksequences);

For all input sequences s in the database D

do

Increment count of all a in Ck if s supports a

Fk = {a Є Ck such that its frequency exceeds the threshold}

k= k+1;

Result = Set of all frequent sequences is the union of all Fks

End do

End do.

*SPADE*

SPADE stands for Sequential Pattern Discovery using Equivalence Classes. SPADE is based GSP. SPADE uses a vertically structured database. SPADE initiates from the bottom-most element of the lattice and works in a bottom-up fashion to generate all frequent sequences. It maintains the vertical structure as it proceeds from the less elements to more

elements.

Algorithm:

SPADE (min\_sup, D):

F1 = {frequent items or 1-sequences};

F2 = {frequent 2 sequences};

E = {Equivalence Class [X]θ1};

for all [X] Є E do Enumerate-Frequent-Seq([X]);

Enumerate-Frequent-Seq(S):

for all atoms Ai Є S do

Ti = φ ;

for all atoms Aj Є S with j>=i do

R = Ai v Aj;

if (Prune(R) == FALSE) then

L(R) =L(Ai ) ∩ L(Aj);

if σ(R) >= min\_sup then

Ti = Ti U {R}; F|R| = F|R| U {R};

end

if(Depth-First-Search) then Enumerate-Frequent-Seq(Ti );

end

if(Breadth-First-Search) then

for all Ti != φ do Enumerate-Frequent-Seq(Ti);

Prune (β):

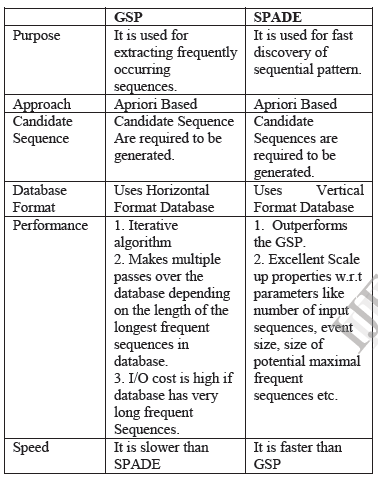
for all (k-1)-subsequences, α < β do

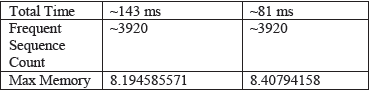
if([ α1] has been processed, and α not Є F(k-1) then

return true;

return false;

GSP AND SPADE COMPARISON





CONCLUSION

TDM has an important use in many other areas of knowledge. There are a lot of TDM techniques, classified considering the similarity measures they used, their applications. A significant number of algorithms are present for TDM.

REFERENCES

[1] Han J, Kamber (2001), Data mining: Concepts and techniques (San Fransisco, CA: Morgan Kauffmann)

[2] Srivatsan Laxman and P S Sastry (2006), A survey of temporal data mining, Sadhana, Vol. 31, Part 2, pp. 173–198.

[3] Claudia M. Antunes and Arlindo L. Oliveira, Temporal Data Mining: an overview. Lecture Notes in Computer Science.