AN OVERVIEW: TEMPORAL-SIDE OF SEQUENTIAL PATTERNS DISCOVERY

Dr Ahmed Aburodes Assaid Alkilany

Department of Computer Science, Faculty of Science, Sebha University
aassaid@gmail.com

ABSTRACT :

Temporal data means a data which have incorporated with the concept of time, to maintain past, present and future data. A Stream of data has may contain time, In other words it can be named as a sequence of data. In this paper temporal data mining concepts and tasks and mining sequential patterns algorithms are discussed and evaluated. The motivation behind this paper is to give preliminary knowledge about a temporal data mining ,as well as presenting and evaluation of the most known algorithms for discovering Temporal-side of Sequential Patterns which will helps new arrival to Temporal Data Mining arena.

KEYWORDS : *data mining, Temporal Data Mining, Temporal Sequential Patterns.*

1. INTRODUCTION

Temporal databases incorporate the concept of time to maintain past, present and future data[1], [2], [3]. They store time-varying information. As most database applications are temporal in nature, e.g. financial applications such as portfolio management, accounting, and banking, record-keeping applications such as personnel, medical-record and inventory management and scientific applications, such as weather monitoring. in the past decade, the study of temporal databases has been an active field of research [4].

Generally, a temporal database supports three distinct types of time attributes which are valid time, transaction time and user-defined time. Valid time stores the time when an event takes place with start time and end time values. Transaction time is the time when the event is recorded in the database and user-defined time is time domains whose semantics are user-defined and depend on the particulars of the application. Since valid time describes the occurrence pattern of events stored in the database, it promises greater utility as a source of domain knowledge than transaction time.

Each record stores the start time and end time during which the tuple is valid. Data is collected in the form of event time sequences where each event lasts for a certain time interval. For instance, in hospital information systems laboratory examinations or clinical records are stored for medical diagnosis of patients' behavior over a certain monitoring period. Records like "patient A had surgery from 10:30 to 13:00 on 14 June" are stored. The temporal nature of data provides us a better understanding of trend or pattern over time as to find any valuable information. For example, we can find patterns like "60% of patients who took medicine A and then took medicine B after an hour, got a fever the following day". The frequent temporal patterns exhibited by patients may identify some correlations between drugs for further diagnosis. Other temporal data are about telecommunication networks, weather and marketing in which by analyzing sequences

DOI : 10.5121/ijdkp.2013.3101

of time-stamped data, we can have a better understanding of the data, which changes over time. Knowledge discovery in temporal databases thus catches the attention of researchers[5], [6].
Furthermore, recent research in temporal databases has made important contributions in characterizing the semantics of temporal information and in providing expressive and efficient means to model, store, and query temporal data[3], [7]. For instance, an extended SQL standard, TSQL2, has been developed for temporal databases[8]. Within the TSQL2 standard, time is widely represented by intervals defined between start time (t_s) and end time (t_e) points. There has been a significant amount of investigation for the development of temporal databases, such as temporal data structures[9], temporal algebraic operators[10], query processing[7], [11], indexing[12], etc.

2. TEMPORAL CONCEPTS

Time is continuous in nature. This continuous nature of time is important in the changing world since changes are possible with time. The sun moving across the sky or advances of program counters all involve changes with time[13]. Authors of [13], [14] have defined some of the basic concepts of time.

2.1 Event

An event is an instantaneous happening, i.e. something occurring at an instant an event is supposed to occur at a time t if it occurs at any instant during t. Moreover, the event occurrence time is the instant at which the event occurs in the real-world. The valid time associated with the event is the time t to which the event occurrence time belongs[13], [14].

2.2 Valid time

The valid time of a fact is the time when the fact is true in the modeled reality. A fact may have associated with number of instants and time intervals, with single instants and intervals being important special cases. Valid times are usually supplied by the user[13].

2.3 Transaction Time

A database fact is stored in a database at some point in time, and after it is stored, it is current until logically deleted. The transaction time of a database fact is the time when the fact is current in the database and may be retrieved. As a consequence, transaction times are generally not time instants, but have duration[13].

2.4 Valid Time vs. Transaction Time

Time has received some attention in database research[15], [16]. This has led to the development of a temporal data model[16], p. 502) that *"… would store, along with information on entities and relationships, both when information was valid in the real world and when that information was recorded in the database."* Both valid time and transaction time can employ fields capturing the start or finish of activities, or equivalent representation.

Valid time, is when the information is valid, e.g., when an event occurred. Whereas, transaction time is when the event was captured, or the resource, agent or location information was established. [15]referred to transaction time as extrinsic time. Valid time was developed before transaction time[16]. A database that supports both valid time and transaction time can be called *bi*Temporal database[17], [18], [19].

2.5 Timestamp

Timestamp (TS) is a time value associated with some timestamped object, for example, an attribute values or a table. The concept can be specialized to valid timestamp, transaction timestamp, interval timestamp, event timestamp. In [17] Devlin has divided timestamp to two types depending on the nurture of events, which may be using a simple timestamp or using two timestamps.

As stated in[19], timestamps may be associated with events as a whole. In this view, a timestamp exists for the lifespan of an event, i.e., from the beginning to the end of its relevance. Timestamps may also refer to evolving properties of the event, expressing the beginning and the end of relevance of certain attribute values.

2.6 User Defined Time

User Defined Time (UDT) is an uninterrupted attribute domain of date and time. User-defined time is parallel to domains such as money and integer, unlike transaction time and valid time; it has no special query language support. It may be used for attributes such as “birth day" and “hiring date"

2.7 Duration

The duration of event can be defined as the absolute distance between start time and end time. And the Duration can be defined formally as

$D(E)=|t_e - t_s|$, where D is the duration of event (E), t_s and t_e are event (E) time start and time end, respectively.

For example, the duration of viewing a webpage P is when a user starts viewing the page P until he switches to another page or stop viewing that page[13], [14].

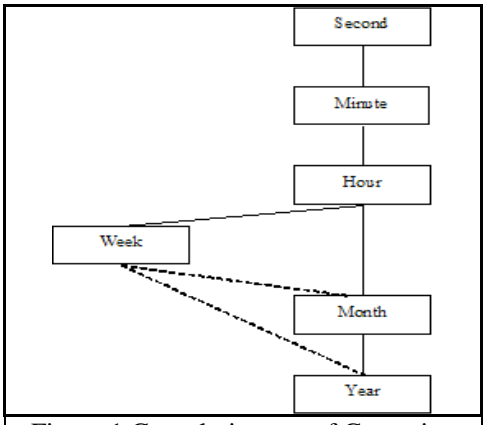


Figure 1 Granularity tree of Gregorian Calendar [20]

2.8 Calendar

A calendar is a human abstraction of physical time space[1], and the measurements of time vary in different calendar systems that are used in different cultural environments. Some familiar

calendar systems are the Gregorian, Islamic and Oriental-lunar calendars. In particular, calendars determine the mapping between human-meaningful time values and an underlying time-line. However, calendars are most often cyclic, allowing human-meaningful time values to be expressed succinctly. For example, dates in the common Gregorian calendar may be expressed in the form < day month year> where each of the fields month, day, and year cycle as time passes. Figure 1 shows the granularity tree of Gregorian, which includes basic granularities of the physical time

Furthermore, a calendar is modeled as a totally ordered set (sequence) of intervals with additional semantics and represented by a tuple *(granularity, pattern, period, start time, end time),* where[20]:

Granularity is the default time unit used in a calendar.
Pattern is a subsequence of time units expressed as a temporal element. If a calendar is periodic, the pattern with respect to one period is specified. Otherwise, the whole sequence of a calendar is specified. If there is more than one possible pattern, they are separated by "|" which means *or.*
Period is the length of a time interval during which a *pattern* occurs repeatedly. It is expressed in terms of the number of time units of a particular granularity in that period, or by a particular granularity.
Start time is the time unit from which a calendar starts.
End time is the time unit at which a calendar ends.

2.9 Granularity

The discovery of temporal patterns or relationships that involve multiple granularities is addressed in[21]. It is stressed that events occurring in the same day, or happening within k weeks from a specific day may capture our attention. With the use of an event structure, which is a set of temporal constraints on a set of variables representing events, we target for patterns of events that match the even structure[21].

2.10 Time intervals

Time interval or calendar interval refers to any specific set of consecutive time units with a given start-time, end-time and granularity, for instance the time interval [1, 4] with granularity of minute is an ordered set {1, 2, 3, 4}, representing minutes[20]. Moreover, according to [22] a calendar unit can also consist of a set of consecutive intervals. For, any calendar unit can be considered as a set of subinterval with smaller granularities. Consequently, based on that approach, the interval between January 1 and January 15 forms a time interval of the calendar units “months” with granularity days, whilst the calendar units “days” also consists of subintervals with granularity hours etc.

2.11 Periodicity

Periodicity refers to regular repetition of a certain event within a specific time interval[20], [22]. For instance, the event “spring” is repeated once within the time interval of a year. Such events are called periodic events, such intervals are called periodic intervals. Each periodicity refers to a specific time period which can be represented in terms of calendar units of a specific granularity (i.e. year with granularity months) or merely in terms of specific granularity, i.e. months. Periodicity is a very important temporal feature frequently identified among the transactions in temporal databases[20], [22], [23].

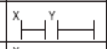
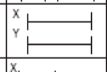
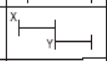
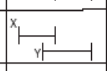
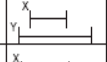
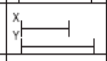
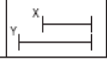
Relation	Symbol	Relation for Inverse	Pictorial Example	Endpoints Constraint
X before Y	b	Y after X		$X.t_s < Y.t_s$
X equal Y	e	Y equal X		$X.t_s = Y.t_s$ $X.t_e = Y.t_e$
X meets Y	m	Y met-by X		$X.t_e = Y.t_s$
X overlaps Y	o	Y overlapped-by X		$X.t_s < Y.t_s$ $X.t_e > Y.t_s$ $X.t_e < Y.t_e$
X during Y	d	Y contains X		$X.t_s > Y.t_s$ $X.t_e < Y.t_e$
X starts Y	s	Y started-by X		$X.t_s = Y.t_s$ $X.t_e < Y.t_e$
X finishes Y	f	Y finished-by X		$X.t_s > Y.t_s$ $X.t_e = Y.t_e$

Table 1 Allen's Taxonomy [24]

3. TEMPORAL REASONING AND TEMPORAL SEMANTICS

Considerable research effort has been directed towards temporal aspects of information systems. One of the areas is temporal reasoning, which involves the issues of time modeling and the representation of temporal relationships based on the underlying temporal domain. There are two primitive notion of temporal data, *time point and time interval*, which temporal reasoning systems are based on. Time points are assumed to be linear and an ordering relation is defined. Intervals are expressed in a pair of start time and end time points, (I^-, I^+), with $(I^- < I^+)$ such that the ordering relations are expressed in terms of relations between their endpoints. Some authors define algebras of temporal relationships according to a classical point of view. One of the most commonly used interval-based formalisms is Allen's interval algebra[24]. It models the relationship between any two intervals as a subset of a set of thirteen basic relations, including before, *meets, overlaps, starts, during and finishes,* together with *their inverses*, plus the relation *equal*. Table 1 illustrate Allen's taxonomy. In corporation with knowledge discovery process, most widely used Allen's interval algebra, as well as first order temporal logic.

4. TEMPORAL DATA MINING

Temporal data constitutes a large portion of data collected in daily operations. In general, temporal data can be loosely defined as any data that contain temporal information. Examples include financial databases for stock price indexes, telecommunications and medical databases. Searching for similar patterns in a temporal database is useful in many applications as we can discover and predict the risks, causality and trends associated with a specific pattern. The accommodation of time into mining techniques provides a window into the temporal arrangement of events and thus an ability to suggest cause and effect or trends in rule sets. Temporal data mining is thus an important extension as it offers the capability to infer causal and temporal proximity relationships that non-temporal data mining is not able to do.

The time component we capture helps in analyzing the changes of the data over time of the system. We may find any causal relationships from the ordering of occurrences of events such as the first condition, which is followed by the second one, is identified as cause and effect relationship, other than association, if no knowledge of time is known. Likewise, the time component may assists in identifying the validity of rules like "Hiking Boots $\rightarrow$ Outerwear", "Years. Months(3:5) during {Years (1990), Years (1995)}". This reveals that every springtime from 1990 to 1995 the customers who buy hiking boots also buy outerwear. Such a rule may not be valid before 1990 or after 1995. We observe that by adding the temporal semantics to the rule

set, more accurate and clear information is obtained[25]. In addition, by discovering the change in knowledge obtained in the underlying data, it is possible to know how quickly the domain is likely to change which helps in better marketing strategies. For example, by identifying frequently or unexpected occurring patterns over event sequences such as stocks with similar price movements, customers' purchasing patterns over seasons as well as rare events happening for fraud detection, we gain more information from the sequences of records. In general, a set of historical data is collected in the form of event time sequences.

Current temporal data mining techniques can be broadly classified into two categories: categorical and numerical data analysis. The former focuses on the discovery of causal relationships among temporally-oriented events. Most of the events concerned are point-based categorical events where only the time when the transaction takes placed is recorded like sales records, telecommunication network alarms, etc. Some of the categorical data are interval-based events the valid times of which are supported by the system such as patient database, scientific databases in geophysics and astronomy areas, etc. The ordering of data is a valuable source of information, which can direct future operations. Numerical data analysis concerns the discovery of similar patterns within the same time sequence or among different time sequences[26], [27], [28]. Numerical values of the sequences are taken into consideration as a comparison for trend discovery and prediction and it is known as time series analysis[29], [30]. Different shapes of the changes of data over time are analyzed[4].

Previous work in knowledge discovery in temporal data mainly concentrates on sequential pattern such as[31], [32], [33], [34], [35]. Although potential knowledge can be extracted, these techniques only treat data as a series in chronological order. They consider the ordering of a string of events and thus mainly support point-based events. Hence, most of these algorithms ignore time intervals, whereas some events sometime are compulsory stamped by time.

4.1 Temporal Data Mining Tasks

A set of relevant and important data mining techniques can be applied on a temporal database. According to techniques of data mining and theory of statistical time series analysis, the theory of temporal data mining may involve the following areas of investigation, since a general theory for this purpose is yet to be developed[36]:

1. Temporal data mining tasks include:

   • Temporal data characterization and comparison,
   • Temporal clustering analysis,
   • Temporal classification,
   • Temporal association rules,
   • Temporal sequential pattern,
   • Temporal pattern analysis, and
   • Temporal prediction and trend analysis.

2. A new temporal data model (supporting time granularity and time-hierarchies) may need to be developed based on:

   • Temporal data structures, and
   • Temporal semantics.

In addition, temporal data mining needs to include an investigation of tightly related issues such as temporal data warehousing, temporal OLAP, computing temporal measurements, etc.

5. MINING SEQUENTIAL PATTERNS

Sequential pattern mining was first introduced by Agrawal and Srikant[32]. They also presented three algorithms for solving the problem of identifying sequential patterns. The AprioriAll algorithm was shown to perform better than the other two approaches. In [37] the same authors proposed the *GSP* algorithm, which is 20 times faster than AprioriAll. They also introduced maximum gap, minimum gap and sliding windows. This branch of data mining attempts to discover frequent subsequences as patterns in a sequence database[38]. One can mine only the maximal frequent subsequences, or all frequent subsequences.

Mannila et al. presented the problem of finding frequent episodes in only one long sequence of events[33]. An episode is defined as a set of events occurring with a partially defined order and within a given time bound. In a later work[34], they generalized their work to allow one to express arbitrary unary conditions on the individual event attributes, or to give binary conditions on the pairs of event attributes. Their experiments were performed using a Web server-level logs file. Oates and Cohen in [39] introduced the problem of detecting strong dependencies among multiple streams of data. Their measure of dependency strength is based on the statistical measure of non independence. [32]provides the following definition: An *itemset* is a non-empty set of items. A *sequence* is an ordered list of itemsets. Without loss of generality, one can assume that the set of items is mapped to a set of contiguous integers.

A record *supports a sequence s* if *s* is contained in it. The support count is incremented only once per record. The support for a sequence is defined as the fraction of the whole data set that contains this sequence. If this support $\geq$ *min_sup*, then the sequence is frequent.

Algorithms for mining frequent itemsets and sequences are useful for discovering Web structure and access patterns. For instance, by putting each Web page as transaction and URLs as items, frequent itemsets result in groups of related URLs that are frequently referenced together. Similarly, frequent sequences in Web log data yield information about user access patterns (that is, the sequence of URLs frequently traversed) that are of immense value to advertisers, Web site designers etc[40]. additionally, in a web access database at a particular site, the discovered patterns are the sequences of the most frequently accessed pages at that site. This kind of information can be used to restructure the website, or to insert dynamically relevant links in web pages based on user access patterns.

5.1 Sequence Data constraints

Unfocused approaches to sequential pattern mining suffer from two major drawbacks, which can briefly be stated as follows[41]:

1) Disproportionate computational cost for selective users.
2) Overwhelming volume of potentially useless results

.

Here selective users can be identified as users searching for a specific pattern rather than frequent patterns, as in the case of gene sequence studies. So there is a need for novel pattern mining solutions that enable the incorporation of user-controlled focus in the mining process.

To achieve this goal, more constraints on data other than the time-window can be integrated to the levels of the data mining process. These are discussed in further detail in[35]. User defined constraints for determining interestingness of found patterns is an important portion of finding frequent sequences and useful rules. Constraints mentioned in related work are as follows:

a) As mentioned above, time constraints are the first basic measurements over sequences of events. Research over temporal databases involves time constraints more densely. Studies by Mannila et al include some user-defined time-window constraints[33], [34], [42]. Basically, the episodes are considered over these time intervals, so that, events occurring far apart from each other will not be counted for further investigation of sequences. Note that, in the time-window constraint, the whole sequence must appear within the time-window in order to be counted as frequent.
b) Length and width restrictions.
c) Minimum time gap between sequence elements.
d) Maximum time gap between sequence elements.
e) Item constraints for including or excluding some items, forming super items, etc.
f) Finding sequences characteristic in at least one class i.e., a special attribute value pair, that we are interested in predicting.
g) And a *flexible constraint specification language* that allows users to express the specific family of patterns that they are interested in. The abstract goal here is to prune the computational cost and ensure system performance with the level on user focus. Providing regular expressions is a promising method of expressing this flexible constraint specification language that is introduced to sequence mining in[41].

5.2 Sequential mining parameters

There are several parameter settings which may strongly influence the results of sequential pattern mining[43].

- The **duration** of a time sequence may be the entire available sequence in the database, or a user-specified subsequence such as that corresponding to the year 2001. Notice that a longer duration means a longer sequence to be mined, which makes the mining task more complex. Durations may also be defined as sets of partitioned sequences, such as every year or every one thousand take-offs and landings, or every time there is a large aircraft crash. In such cases, *periodic patterns* can be discovered.

- When a set of events occur within a specified period of time they can be viewed as occurring together in the same **event folding windo**w, *w*. If *w* is set to the same length as *duratio*n, it will find time-insensitive patterns i.e. association patterns without care as to which item was bought first. For example, *"In 2001, customers who bought a PC bought a digital camera as well*". If *w* is set to 0, sequential patterns found contain events that occur at distinct unrelated time instances, such as *"A customer who bought a PC and then a memory chip is likely to buy a CD-ROM later o*n". If *w* is set to be something in between, then those transactions that occur within one event window are taken as one transaction in the dataset to be mined.

- The third parameter is the time **interval** *int*, between events in the discovered pattern. If *int* = 0, there is no interval gap allowed, so patterns found consist of strictly consecutive sequences, such as $a_{i-1}a_ia_{i+1}$, where a_i is an event occurring at time i. For example, if *int*=0 and the event folding window is set to one week, patterns found will be frequent over consecutive weeks. If instead we want to find separated patterns by at least *min-interval* but at most *max_interval*, we might choose to specify *int* as $min\_interval \leq int \leq max\_interval$. For example, *"If a person rents movie A, he/she will likely rent movie B within 30 day*s" implies $1\ day \leq int \leq 30$ days. The final option is to find patterns carrying an exact interval *int*, where $int = c \neq 0$. For example, the query *"Every time the Dow Jones drops more than 5%, what will exactly happen two days later*?" will search for sequential patterns with *int = 2 day*s.

5.3 Sequential Patterns Algorithms

As we mentioned earlier, several sequential pattern mining algorithms have been introduced to confront the problem of discovering sequential patterns. From the literature, most well known algorithms can be classified into three categories: Apriori-based, Projection-based and SPADE-based.

5.3.1 Apriori-based Algorithms

GSP was introduced in[44]. It extends the algorithms AprioriAll and AprioriSome introduced in [32] by adding some time constraints like minimal and maximal gaps between occurrences of elements (min-gap, max-gap), sliding time windows and taxonomies. Problem definition of the algorithm is: given a database D of data-sequences, a taxonomy T, user-specified min-gap and max-gap time constraints, and a user-specified sliding window-size, to find all sequences whose support is greater than the user-specified minimum support.

Apriori-Based algorithms make multiple passes over the data. Each subsequent pass starts with a seed set: the frequent sequences found in the previous pass. The seed set is used to generate candidate sequences. The support (frequency) for these candidate sequences is found during the pass over the data. The algorithm terminates when there are no frequent sequences at the end of a pass. PSP[45], another Apriori-based algorithm, was developed to improve the way in which GSP stored candidate patterns - in every other respect its process of finding sequential patterns mirrors GSP.

5.3.1 Projection-based Algorithms

FreeSpan [38] was developed to substantially reduce the expensive candidate generation and testing of Apriori, while maintaining its basic heuristics. In general, *FreeSpan* uses frequent items to recursively project the Sequence database into projected databases while growing subsequence fragments in each projected database. Each projection partitions the database and confines further testing to progressively smaller and more manageable units. The trade-off is a considerable amount of sequence duplication as the same sequence could appear in more than one projected database. However, the size of each projected database usually (but not necessarily) decreases rapidly with recursion. PrefixSpan *[46]* was developed to address the costs of *FreeSpan*. Its general idea is that, instead of projecting sequence databases by considering all the possible occurrences of frequent subsequences, the projection is based only on frequent prefixes because any frequent subsequence can always be found by growing a frequent prefix.

5.3.2 SPADE-based Algorithms

SPADE (**S**equential **PA**ttern **D**iscovery using **E**quivalence Classes) algorithm was presented recently in[47], [48]. It uses vertical id-list database format where each sequence is associated with a list of objects in which the sequence occurs along with the timestamps. It decomposes the search space lattice into sub-lattices and processes them independently in the main memory. Three database scans are needed to compared with the multiple scans of data in other approaches. Two different search strategies, breadth-first search and depth-first search are used for enumerating frequent sequences. *c*SPADE is again a study of Zaki et al presented in[35]. It is mainly the extension of SPADE algorithm with the constraints in GSP. There are other algorithms, which have been introduced and are based on SPADE algorithms, such as webSPADE[49] and Go-SPADE.[50].

5.4 Evaluation of Sequential Patterns discovery Algorithms

As stated in the research which conducted by [51] as well as stated in [35], [48] the experimental results show that SPADE outperforms GSP by a factor of two, and by an order of magnitude with pre-processed data. It has two important advantages:

1) Reducing the I/O costs by reducing the database scans and, more importantly
2) Decomposing the problem into sub-problems which can be solved independently, so that it makes parallel computing possible.

Authors of GSP and *c*SPADE have implemented some further constraints like item constraints for including or excluding certain items and class predictions, while excluding the taxonomy presentation of GSP. Results obtained show that the difference between cSPADE and GSP narrows when min-gap and max-gap constraints are considered, but overall again cSPADE outperforms GSP. But this is a natural result as SPADE also has outperformed GSP.

1. Table 2 shows the evaluation of most known sequential patterns algorithms based on the literature of mining sequential patterns, which we have illustrated earlier. Each algorithm is based on one of three categorises: Apriori-based, Projection-based and SPADE-based. We explained how the data in these algorithms are laid out, the type of data structure used by each category and how may times the algorithm passes over the database.

Type	Algorithms	Data Layout	Data structure	Passing on data
Apriori Based	AprioriAll AprioriSome GSP	Horizontal formatting	Hash tree	Several times scan
	PSP		Prefix-tree	
Projection Based	FreeSpan PrefixSpan	Projection Data		
SPADE Based	SPADE cSPADE webSPADE GO-SPADE	Vertical formatting	Lattice Based	Three times scan

Table 2 Sequential patterns discovery algorithms valuation

6. CONCLUSION

In this paper, concepts of temporal-side aspect are explained, and temporal data mining and its tasks are discussed. However, we precisely concentrate on mining temporal-side sequential patterns. In recent years, Several approaches for mining sequential data have been proposed.

In this paper, The concept of mining sequential patterns has been discussed and some of the well known mining sequential pattern algorithms have been evaluated which have concentrated on temporal side. As result of that mining algorithms have been categorized into three types, in relation to layout of data, Data structure as well as passing on data. which helps a researcher to choose the type of algorithms may suite their criteria of research.

REFERENCES

[1] Tansel, A.U., et al., Temporal Databases: Theory Design and Implementation. Benjamin-Cummings, 1993.
[2] Wu, Y., S. Jajodia, and X.S. Wang, Temporal database bibliography update. In Temporal Databases - Research and Practice Lecture Notes in Computer Science, 1998: p. 338-366.
[3] Jensen, C.S. and R.T. Snodgrass, Temporal data management. IEEE Transactions on Knowledge and Data Engineering, 1999. 11(1): p. 36-43.
[4] Agrawal, R., G. Psaila, and E.L. Wimmers, Querying shapes of histories. In Proceedings of the 21st International Conference on Very Large Databases, 1995: p. 502-514.
[5] Saraee, M. and C. Theodoulidis, Knowledge discovery in temporal databases. In Proceedings of IEE Colloquium on Knowledge Discovery in Databases, 1995: p. 1-4.

[6] Chen, X. and I. Petrounias, An architecture for temporal data mining. In IEE Colloquium on Knowledge Discovery and Data Mining, 1998: p. 8/1-8/4.
[7] Bettini, C., S. Wang, and S. Jajodia, Temporal semantic assumptions and their use in databases. IEEE Transactions on Knowledge and Data Engineering, 1998. 10(2): p. 277-296.
[8] Snodgrass, R.T., A.G.e. al, and A. Ilsoo, The TSQL2 Temporal Query Language. Kluwer Academic Publishers, 1995.
[9] Amagasa, T., M. Aritsugi, and Y. Kanamori, Implementing time-interval class for managing temporal data. In Proceedings of the 9th International Workshop on Database and Expert Systems Applications, 1998: p. 843-849.
[10] Tuzhilin, A. and J. Clifford, A temporal relational algebra as a basis for temporal relational completeness. In Proceedings of the 16th International Conference on Very Large Data Bases, 1990: p. 13-23.
[11] Toman, D., Point vs. interval-based query languages for temporal databases. In Proceedings of the 5th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, 1996: p. 58-67.
[12] Elmasri, R., G.T.J. Wuu, and Y.J. Kim, The time index: An access structure for temporal data. In Proceedings of the 16th International conference on Very Large Data Bases, 1990.
[13] Dyreson, C., et al., A consensus glossary of temporal database concepts, in Temporal Databases - Research and Practice, O. Etzion, S. Jajodia, and S. Sripada, Editors. 1998, Springer-Verlag: Berlin Heidelberg. p. 367-405.
[14] Chountas, P., Uncertain & conflicting Information in Databases & Information systems, in Department of Computation. 2002, University of Manchester Institute of science and Technology: Manchester.
[15] Bubenko, J. and J. A., The temporal dimension in information modeling. In Architecture and Models in Data Base Management Systems., 1977. Amsterdam North-Holland.
[16] McKenzie, E. and R.T. Snodgrass, Supporting Valid Time in an Historical Relational Algebra: Proofs and Extensions, in Technical Report TR 91-15. 1991, Department of Computer Science, University of Arizona: Tucson, AZ.
[17] Devlin, B., Managing Time in the Data Warehouse. infoDB, 1997. 11(1).
[18] Abellَ, A. and C. Martيn. A Bitemporal Storage Structure for a Corporate Data Warehouse. in ICEIS 2003, Proceedings of the 5th International Conference on Enterprise Information Systems. 2003. Angers, France.
[19] Knolmayer, G.F. and T. Myrach. Concepts of Bitemporal Database Theory and the Evolution of Web Documents. in Proceedings of the 34th Hawaii International Conference on System Sciences - 2001. 2001. Hawaii.
[20] Lee, J.Y., R. Elmasri, and J. Won, An integrated temporal data model incorporating time series concept. Data Knowledge Eng. 24(3), 1998: p. 257-276.
[21] Bettini, C., et al., Discovering frequent event patterns with multiple granularities in time sequences. IEEE Transactions on Knowledge and Data Engineering, 1998. 10(2): p. 222-236.
[22] Chen, X. and I. Petrounias, A development framework for temporal data mining, in Knowledge discovery and data mining, M.A. Bramer, Editor. 1999, IEE Stevenage, UK: Univ. of Ports Mouth, Ports Mouth, UK. p. 87 - 113.
[23] Chen, X., Temporal Data Mining: algorithms Language and System for Temporal Association Rules. 1999, the Manchester Metropolitan University: Manchester.
[24] Allen, J.F., Maintaining knowledge about temporal intervals. Communications of the ACM, 1983. 26(11): p. 832-843.
[25] Chen, X. and I. Petrounias, Language support for temporal data mining. In Proceedings of the 2nd European Symposium on Principles of Data Mining and Knowledge Discovery, 1998: p. 282-290.
[26] Faloutsos, C., M. Ranganathan, and Y. Manolopoulos, Fast Subsequence Matching in Time-Series Databases. ACM SIGMOD Int. Conf. on Management of Data (1994), 1994: p. 419-429.
[27] Agrawal, R., et al., Fast similarity search in the presence of noise, scaling and translation in time-series databases. In Proceedings of the 21st International Conference on Very Large Databases, 1995: p. 490-501.
[28] Das, G., D. Gunopulos, and H. Mannila, Finding Similar Time Series. In Proc. 1997 1st European Symposium on Principles of Data Mining and Knowledge Discovery (PKDD'97), Trondheim, Norway., 1997: p. 88-100.
[29] Das, G., et al., Rule discovery from time series. In Proceedings of the 2nd European Symposium on Principles of Data Mining and Knowledge Discovery, 1998: p. 16-22.

[30] Guralnik, V. and J. Srivastava, Event detection from time series data. In Proceedings of the 5th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 1999: p. 33-42.
[31] Agrawal, R., C. Faloutsos, and A. Swami, Efficient similarity search in sequence database. In Proceedings of the 4th International Conference on Foundations of Data Organization and Algorithms, 1993: p. 69-84.
[32] Agrawal, R. and R. Srikant, Mining Sequential Patterns. Proc. of the 11th Int'l Conference on Data Engineering (ICDE'95), Taipei, Taiwan, 1995: p. 3 - 14.
[33] Mannila, H., H. Toivonen, and A.I. Verkamo, Discovering Frequent Episodes in Sequences. In Proc. of the First Int'l Conference on Knowledge Discovery and Data Mining (KDD'95), Montreal, Quebec, 1995: p. 210-215.
[34] Mannila, H. and H. Toivonen., Discovering generalized episodes using minimal occurrences. In Proc. of the Second Int' l Conference on Knowledge Discovery and Data Mining, 1996. (KDD' 96): p. 146-151, Portland, Oregon.
[35] Zaki, M.J., Sequence Mining in Categorical Domains: Incorporating Constraints. to appear ,in 9th International Conference on Information and Knowledge Management Washington, DC, 2000: p. 422-429.
[36] Lin, W., M.A. Orgun, and G.J. Williams. An Overview of Temporal Data Mining. in Proceedings of the 1st Australian Data Mining Workshop (ADM02). 2002. Canberra, Australia: University of Technology, Sydney.
[37] Srikant, R. and R. Agrawal, Mining Sequential Patterns: Generalizations and Performance Improvements. IBM Research Report RJ 9994, 1995.
[38] Han, J., et al., FreeSpan: Frequent Pattern-Projected Sequential Pattern Mining. In Proc. 2000 Int. Conf. Knowledge Discovery and Data Mining (KDD'00) Boston, MA, 2000: p. 355-359.
[39] Oates, T. and P.R. Cohen, Searching for structure in multiple streams of data. In Proc. Of Thirteenth Int'l Conference on MachineLearning (ICML'96) Bari, Italy, 1996: p. 346-354.
[40] Garofalakis, M., et al., Data Mining and the Web: Past, Present and Future. In Workshop on Web Information and Data Management, 1999: p. 43-47.
[41] Garofalakis, M., R. Rastogi, and K. Shim., SPIRIT: Sequential Pattern Mining with Regular Expression Constraints. In 25th Intl. Conf. Very Large Databases, 1999.
[42] Mannila, H., H. Toivonen, and A.I. Verkamo, Discovery of Frequent Episodes In Event Sequences. Technical Report C-1997-15, Department of Computer Science, University of Helsinki, 1997.
[43] Han, J. and M. Kamber, Data Mining: Concepts and Techniques. SanFrancisco: Morgan Kaufmann Publishers, 2001.
[44] Srikant, R. and R. Agrawal, Mining Sequential Patterns: Generalizations and Performance Improvements. In Proc. of the Fifth Int' l Conference on Extending Database Technology, Avignon, France, 1996.
[45] Masseglia, F., F. Cathala, and P. Poncelet. The PSP Approach for Mining Sequential Patterns. in Proceedings of the 2nd European Symposium on Principles of Data Mining and Knowledge Discovery (PKDD'98). 1998. Nantes, France.
[46] Pei, J., et al. PrefixSpan: Mining Sequential Patterns Efficiently by Prefix-Projected Pattern Growth. in Proceedings of Int. Conf. Data Engineering (ICDE'01). 2001. Heidelberg Germany.
[47] Zaki, M.J., Efficient Enumeration of Frequent Sequences. 7th International Conference on Information and Knowledge Management. Washington DC, 1998: p. 68-75.
[48] Zaki, M.J., SPADE: An Efficient Algorithm for Mining Frequent Sequences. In Proc. of Machine Learning Journal, special issue on Unsupervised Learning (Doug Fisher, ed.), 2001. Vol. 42 Nos. 1/2: p. 31-60.
[49] Demiriz, A. and M.J. Zaki, webSPADE: A Parallel Sequence Mining Algorithm to Analyze Web Log Data. SIGKDD '02 Edmonton, Alberta CA, 2002.
[50] Leleu, M., et al. GO-SPADE: Mining Sequential Patterns over Datasets with Consecutive Repetitions. in Proceedings of the International Conference on Machine Learning and Data Mining (MLDM 2003). 2003. Berlin Heidelberg: Springer.
[51] Ahola, J., Mining Sequential Patterns (MOSE). 2001, VTT INFORMATION TECHNOLOGY: Espoo, Finland.