Sequence Pattern mining

By

S. Jeba Priya

Sequence Pattern Mining

- Sequence database – consists of sequences of ordered elements or events (with or without time)
- Sequential Pattern Mining is the mining of frequently occurring ordered events or subsequences as patterns
- Example:
  - Customer shopping sequences:
    - First buy computer, then CD-ROM, and then digital camera, within 3 months.
  - Web access patterns, Weather prediction
- Usually categorical or symbolic data
  - Numeric data analysis – Time Series Analysis

Sequence Pattern Mining

- $I = \{I_1, I_2, \ldots I_p\}$ – Set of items
- Sequence s = $<e_1\ e_2\ e_3 \ldots\ e_l>$
    - Ordered list of events
        - Each event is an element of the sequence
        - In Shopping databases, event – shopping trip in which customers bought items $(x_1\ x_2 \ldots x_k)$
    - All trips by customer form a sequence
        - Item can occur at most once in an event, but several times in a sequence
    - Sequence with length l : l-sequence
    - A sequence $\alpha = <a_1 a_2 \ldots a_n>$ is a subsequence of $\beta = <b_1 b_2 .. b_m>$ denoted as $\alpha \subseteq \beta$ if there exists integers $j_1, j_2, \ldots j_n$ between 1 and m such that $a_1 \subseteq b_{j1}$, $a_2 \subseteq b_{j2}, \ldots a_n \subseteq b_{jn}$
        - $\alpha = <(ab),d>$ and $\beta = <(abc),(de)>$ α is a sub-sequence of β

Sequence Pattern Mining

- A sequence database, S, is a set of tuples, <SID,s>, where SID is a sequence ID and s is a sequence
- A tuple <SID, s> is said to contain a sequence a, if a is a subsequence of s
- The support of a sequence α in a sequence database S is the number of tuples in the database containing α
- Given the minimum support threshold, a sequence a is frequent in sequence database S if support $_S$(a) >= min sup
- A frequent sequence is called a sequential pattern
- A sequential pattern with length l is called an l-pattern

Sequence Patterns

SID	Sequence
1	<a(abc)(ac)d(cf)>
2	<(ad)c(bc)(ae)>
3	<(ef)(ab)(df)cb>
4	<eg(af)cbc>

- Length of Sequence 1 – 9
- It contains ‘a’ multiple times but sequence 1 will contribute only one to the support of <a>
- Sequence <a(bc)df> - Subsequence of Sequence 1
- Support for <(ab)c> is 2 (Present in 1 and 3)
  - Frequent as it satisfies minimum support of 2

Scalable Methods for Mining Sequential Patterns

- Full Set Vs Closed Set
  - A sequential patterns s is closed if there exists no s' where s' is a proper super-sequence of s and s' has same support as s
    - Subsequences of a frequent sequence are also ferquent
    - Mining closed sequential patterns avoids generation of unnecessary sub-sequences
- GSP – Candidate generate and test approach on horizontal data format
- SPADE – Candidate generate and test approach on vertical data format
- PrefixScan – does not require candidate generation
- All approaches exploit Apriori property – every non empty subsequence of a sequential pattern is a sequential pattern

GSP – Sequential Pattern Mining

- GSP (Generalized Sequential Patterns)
  - Multi-pass, Candidate generate and test approach proposed by Agrawal and Srikant
- Outline of the method
  - Initially, every item in DB is a candidate of length-1
  - for each level (i.e., sequences of length-k) do
    - Scan database to collect support count for each candidate sequence
    - Generate candidate length-(k+1) sequences from length-k frequent sequences using Apriori
      - A *(k-1)*-sequence w_1 is merged with another *(k-1)*-sequence w_2 to produce a candidate *k*-sequence if the subsequence obtained by removing the first event in w_1 is the same as the subsequence obtained by removing the last event in w_2
  - repeat until no frequent sequence or no candidate can be found
- Major strength: Candidate pruning by Apriori
- Weakness : Generates large number of candidates

SPADE – Sequential Pattern Mining on Vertical data format

- SPADE – Sequential PAttern Discovery using Equivalent classes
- Vertical data format <itemset: (sequence_ID, event_ID)>
- ID_list : Set of (sequence_ID, event_ID) pairs for a given itemset
- Mapping from horizontal to vertical format requires one scan
- Support of k-sequences can be determined by joining the ID lists of (k-1) sequences
  - To find candidate 2-sequences, all single items are joined if they are frequent, if they share the same sequence identifier and if their event identifier follows a sequential ordering
  - Patterns are grown similarly

Vertical Data Format

SID	EID	itemset
1	1	a
1	2	abc
1	3	ac
1	4	d
1	5	cf
2	1	ad
2	2	c
2	3	bc
2	4	ae
3	1	ef
3	2	ab
3	3	df
3	4	c
3	5	b
4	1	e
4	2	g
4	3	af
4	4	c
4	5	b
4	6	c

(a) vertical format database

a		b		···
SID	*EID*	*SID*	*EID*	···
1	1	1	2	
1	2	2	3	
1	3	3	2	
2	1	3	5	
2	4	4	5	
3	2			
4	3			

(b) ID_lists for some 1-sequences

ab			ba			···
SID	*EID(a)*	*EID(b)*	*SID*	*EID(b)*	*EID(a)*	···
1	1	2	1	2	3	
2	1	3	2	3	4	
3	2	5				
4	3	5				

(c) ID_lists for some 2-sequences

aba				···
SID	*EID(a)*	*EID(b)*	*EID(a)*	···
1	1	2	3	
2	1	3	4	

(d) ID_lists for some 3-sequences

SPADE

- Reduces scans of the sequence database
- As the length of the frequent sequence increases, the size of ID_list decreases – results in fast joins
- But large set of candidates are still generated

Prefix Span

- Given a sequence $\alpha = <e_1 e_2 \ldots e_n>$ (where each e_i corresponds to a frequent event), a sequence $\beta = <e'_1 e'_2 \ldots e'_m>$ ($m<=n$) is called a prefix of α iff
  - $e'_i = e_i$ for $i<=m-1$
  - $e'_m \subseteq e_m$
  - All frequent items in $(e_m - e'_m)$ are alphabetically after those in e'_m
- Sequence $\gamma = <e''_m e_{m+1} \ldots e_n>$ is called the suffix of α wrt prefix β denoted as $\gamma = \alpha/\beta$ where $e''_m = (e_m - e'_m)$

<a>, <aa>, <a(ab)> and <a(abc)> are prefixes of sequence <a(abc)(ac)d(cf)>

Prefix	Suffix (Prefix-Based Projection)
<a>	<(abc)(ac)d(cf)>
<aa>	<(_bc)(ac)d(cf)>
<a(ab)>	<(_c)(ac)d(cf)>

Prefix Span

- Mining Sequential patterns:
  - $\{<x_1>,<x_2>,\ldots<x_n>\}$ – complete set of length-1 sequential patterns.
    - The complete set of Sequential patterns in S can be partitioned into n disjoint subsets.
    - i^{th} subset is the set of sequential patterns with prefix $<x_i>$
  - α : length-l sequential pattern and $\{\beta_1, \beta_2,\ldots \beta_m\}$ – set of all length (l+1) sequential patterns with prefix α.
    - The complete set of sequential patterns with α as a prefix can be partitioned into m disjoint subsets
    - j^{th} subset is the set of patterns prefixed with β_j

Prefix Span - Example

- Step 1: find length-1 sequential patterns
  - <a>, <b>, <c>, <d>, <e>, <f>
- Step 2: divide search space. The complete set of seq. pat. can be partitioned into 6 subsets:
  - The ones having prefix <a>;
  - The ones having prefix <b>;
  - …
  - The ones having prefix <f>

SID	sequence
10	<a(abc)(ac)d(cf)>
20	<(ad)c(bc)(ae)>
30	<(ef)(ab)(df)cb>
40	<eg(af)cbc>

PrefixSpan : Example

SID	sequence
10	<a(abc)(ac)d(cf)>
20	<(ad)c(bc)(ae)>
30	<(ef)(ab)(df)cb>
40	<eg(af)cbc>

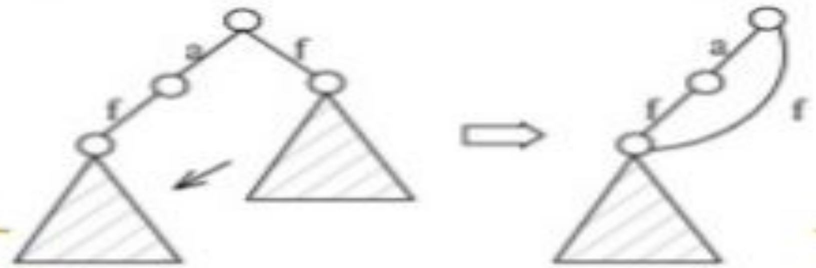
- Only need to consider projections w.r.t. <a>
  - <a>-projected database: (only first occurrence of a is considered) <(abc)(ac)d(cf)>, <(_d)c(bc)(ae)>, <(_b)(df)cb>, <(_f)cbc>
  - In <a> projected database – frequent items are a:2, b:4, _b:2, c:4, d:2 and f:2
- Find all the length-2 seq. pat. Having prefix <a>: <aa>, <ab>, <(ab)>, <ac>, <ad>, <af>
  - Further partition into 6 subsets
    - Having prefix <aa> - {< (_bc)(ac)d(cf)>, <(_e)>} – No frequent subsequences
    - Having prefix <(ab)> - <(_c)(ac)d(cf)> and <(df)cb>
      - Frequent patterns: <c> <d> <f> <dc> : <(ab)c> <(ab)d> <(ab)f> <(ab)dc>
    - Having prefix <ac> - <(ac)d(cf)>, <(bc)(ae)>, <b>, <bc>
      - Frequent patterns: <a> <b> <c> : <aca> <acb> <acc>

Prefix Span

- No candidate sequence needs to be generated
- Projected databases keep shrinking
- Major cost of PrefixSpan: constructing projected databases
  - Can be improved by pseudo-projections
- Pseudo-projection
  - Registers the index of the corresponding sequence and the starting position of the projected suffix in the sequence instead of physical projection
  - Avoids physically copying postfixes
  - Efficient in running time and space when database can be held in main memory
  - For large data combination of physical and pseudo projection

Sequential Pattern Mining

- Performance rating : PrefixSpan, SPADE, GSP
- All three are slow when there is a large number of frequent subsequences
- Closed Sequential Patterns
  - Closed Subsequences – contain no super sequence with the same support
  - Reduces number of sequences considered
  - CloSpan
    - Based on equivalence of projected databases
    - Two projected sequence databases are equivalent iff the total number of items match
    - CloSpan prunes non-closed sequences whenever two projected databases are exactly the same by stopping the growth of one
    - Requires Post-processing to eliminate any remaining non-closed sequential patterns
    - BIDE (Bidirectional Search) algorithm avoids additional checking



Sequential Pattern Mining

- Multi-dimensional, multi-level Sequential patterns
  - Additional information maybe associated with Sequence ID – Customer age, address, group and profession
  - Additional information associated with items – item category, brand, model type, model number, place…
  - Example: "Retired customers who purchase a digital camera are likely to purchase a color printer within a month"
  - Additional information can be attached with Sequence ID / Item ID

Constraint based Mining of sequential pattern

- Un-focused mining reduces the efficiency and usability of frequent-pattern mining
- Constraint based mining incorporates user-specified constraints to reduce the search space
  - Regular expressions can be used to specify pattern templates
  - Helps to improve efficiency of mining and interestingness of patterns

Constraint based Mining of sequential pattern

- Constraints related to duration T
  - Constraints related to maximal or minimal length – anti-monotonic / monotonic constraints
  - Anti-monotonic constraint: T<= 10
  - Monotonic : T > 10
  - Succinct : T = 2005
  - Periodic patterns – related to sets of partitioned sequences such as every two weeks before and after an earthquake

Constraint based Mining

- Event folding window w
  - specifies the periodicity for treating events as occurring together
  - w=0 – No event sequence folding
  - w=T – time-insensitive frequent patterns
- Gap between events
  - Gap=0 – Strictly consecutive sequential patterns
  - min_gap and max_gap
  - Exact gaps and approximate gaps
- Serial Episodes
  - Set of events occurring in total order
- Parallel Episodes
  - Occurrence order is trivial
- Examples
  - (A|B)C*(D|E) – A and B first occur (relative order is unimportant) followed by any number of C events followed by D and E (in any order)
  - C = <a*{bb|(bc)d|dd}> a-projected databases followed by SuffixSpan

Periodicity Analysis for Time-Related Sequential Data

- Periodicity analysis – mining of periodic patterns – searching for recurring patterns in time-related sequence data
  - Seasons, tides, traffic patterns, power consumption
  - Often performed over time-series data
  - Full periodic pattern
    - Every point in time contributes (precisely or approximately) to the periodicity
      - Example: All days in a year – contribute to season cycle
  - Partial periodic pattern
    - Specifies periodic behavior of a time-related sequence at some but not all points of time
      - Example: ABC reads the paper between 7:00-7:30 am every week day

Periodicity Analysis for Time-Related Sequential Data

- Synchronous periodicity – event occurs at a relatively fixed offset in each "stable" period
  - 3 pm every day
- Asynchronous periodicity – event fluctuates in loosely defined period
- Precise or approximate – depending on data value or offset within a period
- Mining partial periodicity – leads to the discovery of cyclic or periodic association rules (rules that associate a set of events that occur periodically)
  - Example: If tea sells well between 3 – 5 pm dinner will also sell well between 7 – 9 pm on weekends