Unit-V

Mining sequence patterns

Dr. K.RAGHAVA RAO

Professor of CSE

Dept. of MCA

KL University

krraocse@gmail.com

http://datamining.blog.com

Sequence Databases & Sequential Patterns

- Transaction databases, time-series databases vs. sequence databases
- Frequent patterns vs. (frequent) sequential patterns
- Applications of sequential pattern mining
  - Customer shopping sequences:
    - First buy computer, then CD-ROM, and then digital camera, within 3 months.
  - Medical treatments, natural disasters (e.g., earthquakes), science & eng. processes, stocks and markets, etc.
  - Telephone calling patterns, Weblog click streams
  - Program execution sequence data sets
  - DNA sequences and gene structures

What Is Sequential Pattern Mining?

- Given a set of sequences, find the complete set of *frequent* subsequences

A *sequence* : < (ef) (ab) (df) c b >

An element may contain a set of items. Items within an element are unordered and we list them alphabetically.

A *sequence database*

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

<a(bc)dc> is a *subsequence* of <a(abc)(ac)d(cf)>

Given *support threshold* *min_sup* =2, <(ab)c> is a *sequential pattern*

Challenges on Sequential Pattern Mining

- A huge number of possible sequential patterns are hidden in databases
- A mining algorithm should
  - find the complete set of patterns, when possible, satisfying the minimum support (frequency) threshold
  - be highly efficient, scalable, involving only a small number of database scans
  - be able to incorporate various kinds of user-specific constraints

Sequential Pattern Mining Algorithms

- Concept introduction and an initial Apriori-like algorithm
  - Agrawal & Srikant. Mining sequential patterns, ICDE'95
- Apriori-based method: GSP (Generalized Sequential Patterns: Srikant & Agrawal @ EDBT'96)
- Pattern-growth methods: FreeSpan & PrefixSpan (Han et al.@KDD'00; Pei, et al.@ICDE'01)
- Vertical format-based mining: SPADE (Zaki@Machine Leanining'00)
- Constraint-based sequential pattern mining (SPIRIT: Garofalakis, Rastogi, Shim@VLDB'99; Pei, Han, Wang @ CIKM'02)
- Mining closed sequential patterns: CloSpan (Yan, Han & Afshar @SDM'03)

The Apriori Property of Sequential Patterns

- A basic property: Apriori (Agrawal & Sirkant'94)
  - If a sequence S is not frequent
  - Then none of the super-sequences of S is frequent
  - E.g, <hb> is infrequent → so do <hab> and <(ah)b>

Seq. ID	Sequence
10	<(bd)cb(ac)>
20	<(bf)(ce)b(fg)>
30	<(ah)(bf)abf>
40	<(be)(ce)d>
50	<a(bd)bcb(ade)>

Given *support threshold* *min_sup* =2

GSP—Generalized Sequential Pattern Mining

- GSP (Generalized Sequential Pattern) mining algorithm
  - proposed by Agrawal and Srikant, EDBT'96
- 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
  - repeat until no frequent sequence or no candidate can be found
- Major strength: Candidate pruning by Apriori

Finding Length-1 Sequential Patterns

- Examine GSP using an example
- Initial candidates: all singleton sequences
  - <a>, <b>, <c>, <d>, <e>, <f>, <g>, <h>
- Scan database once, count support for candidates

Cand	Sup
<a>	**3**
<b>	**5**
<c>	**4**
<d>	**3**
<e>	**3**
<f>	**2**
~~<g>~~	1
~~<h>~~	1

min_sup =2

Seq. ID	Sequence
10	<(bd)cb(ac)>
20	<(bf)(ce)b(fg)>
30	<(ah)(bf)abf>
40	<(be)(ce)d>
50	<a(bd)bcb(ade)>

GSP: Generating Length-2 Candidates

51 length-2 Candidates

	<a>	<b>	<c>	<d>	<e>	<f>
<a>	<aa>	<ab>	<ac>	<ad>	<ae>	<af>
<b>	<ba>	<bb>	<bc>	<bd>	<be>	<bf>
<c>	<ca>	<cb>	<cc>	<cd>	<ce>	<cf>
<d>	<da>	<db>	<dc>	<dd>	<de>	<df>
<e>	<ea>	<eb>	<ec>	<ed>	<ee>	<ef>
<f>	<fa>	<fb>	<fc>	<fd>	<fe>	<ff>

	<a>	<b>	<c>	<d>	<e>	<f>
<a>		<(ab)>	<(ac)>	<(ad)>	<(ae)>	<(af)>
<b>			<(bc)>	<(bd)>	<(be)>	<(bf)>
<c>				<(cd)>	<(ce)>	<(cf)>
<d>					<(de)>	<(df)>
<e>						<(ef)>
<f>						

Without Apriori property, 8*8+8*7/2=92 candidates

Apriori prunes 44.57% candidates

The GSP Mining Process

5th scan: 1 cand. 1 length-5 seq. pat.

4th scan: 8 cand. 6 length-4 seq. pat.

3rd scan: 46 cand. 19 length-3 seq. pat. 20 cand. not in DB at all

2nd scan: 51 cand. 19 length-2 seq. pat. 10 cand. not in DB at all

1st scan: 8 cand. 6 length-1 seq. pat.

<(bd)cba>

Cand. cannot pass sup. threshold

<abba> <(bd)bc> …

Cand. not in DB at all

<abb> <aab> <aba> <baa> <bab> …

<aa> <ab> … <af> <ba> <bb> … <ff> <(ab)> … <(ef)>

<a> <b> <c> <d> <e> <f> <g> <h>

min_sup =2

Seq. ID	Sequence
10	<(bd)cb(ac)>
20	<(bf)(ce)b(fg)>
30	<(ah)(bf)abf>
40	<(be)(ce)d>
50	<a(bd)bcb(ade)>

Candidate Generate-and-test: Drawbacks

- A huge set of candidate sequences generated
  - Especially 2-item candidate sequence
- Multiple Scans of database needed
  - The length of each candidate grows by one at each database scan
- Inefficient for mining long sequential patterns
  - A long pattern grow up from short patterns
  - The number of short patterns is exponential to the length of mined patterns

The SPADE Algorithm

- SPADE (Sequential PAttern Discovery using Equivalent Class) developed by Zaki 2001
- A vertical format sequential pattern mining method
- A sequence database is mapped to a large set of
  - Item: <SID, EID> sequence id and eventid
- Sequential pattern mining is performed by
  - growing the subsequences (patterns) one item at a time by Apriori candidate generation

The SPADE Algorithm

SID	EID	Items
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		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			

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				

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

Bottlenecks of GSP and SPADE

- A huge set of candidates could be generated
  - 1,000 frequent length-1 sequences generate s huge number of length-2 candidates! $1000 \times 1000 + \frac{1000 \times 999}{2} = 1{,}499{,}500$
- Multiple scans of database in mining
- Breadth-first search
- Mining long sequential patterns
  - Needs an exponential number of short candidates
  - A length-100 sequential pattern needs 10^{30} candidate sequences! $\sum_{i=1}^{100} \binom{100}{i} = 2^{100} - 1 \approx 10^{30}$

Prefix and Suffix (Projection)

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

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

Mining Sequential Patterns by Prefix Projections

- 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>

Finding Seq. Patterns with Prefix <a>

- Only need to consider projections w.r.t. <a>
  - <a>-projected database:
    - <(abc)(ac)d(cf)>
    - <(_d)c(bc)(ae)>
    - <(_b)(df)cb>
    - <(_f)cbc>

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

- Find all the length-2 seq. pat. Having prefix <a>: <aa>, <ab>, <(ab)>, <ac>, <ad>, <af>
  - Further partition into 6 subsets
    - Having prefix <aa>;
    - ...
    - Having prefix <af>

Completeness of PrefixSpan

SDB

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

Length-1 sequential patterns
<a>, <b>, <c>, <d>, <e>, <f>

Having prefix <a>

Having prefix <b>

Having prefix <c>, ..., <f>

<u><a>-projected database</u>
<(abc)(ac)d(cf)>
<(_d)c(bc)(ae)>
<(_b)(df)cb>
<(_f)cbc>

Length-2 sequential patterns
<aa>, <ab>, <(ab)>, <ac>, <ad>, <af>

<u><b>-projected database</u>

…

… …

Having prefix <aa>

Having prefix <af>

<u><aa>-proj. db</u> … <u><af>-proj. db</u>

Efficiency of PrefixSpan

- No candidate sequence needs to be generated
- Projected databases keep shrinking
- Major cost of PrefixSpan: Constructing projected databases
  - Can be improved by pseudo-projections

Speed-up by Pseudo-projection

- Major cost of PrefixSpan: projection
  - Postfixes of sequences often appear repeatedly in recursive projected databases
- When (projected) database can be held in main memory, use pointers to form projections
  - Pointer to the sequence
  - Offset of the postfix

s=<a(abc)(ac)d(cf)>

↓ <a>

s|<a>: (, 2) <(abc)(ac)d(cf)>

↓ <ab>

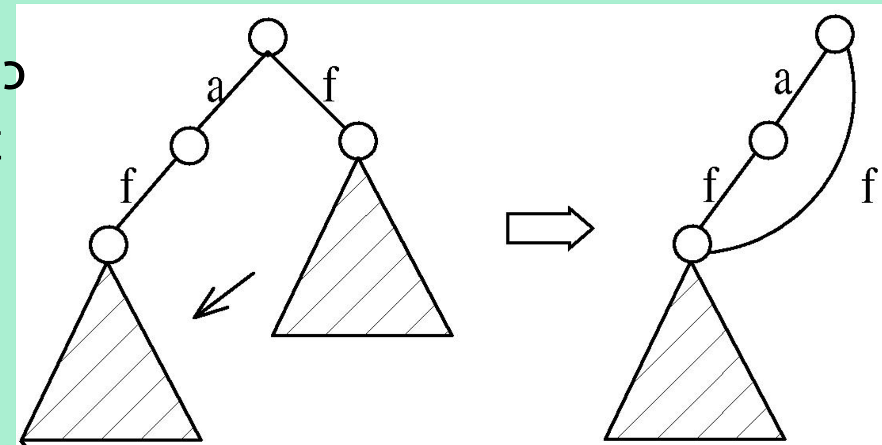
s|<ab>: (, 4) <(_c)(ac)d(cf)>

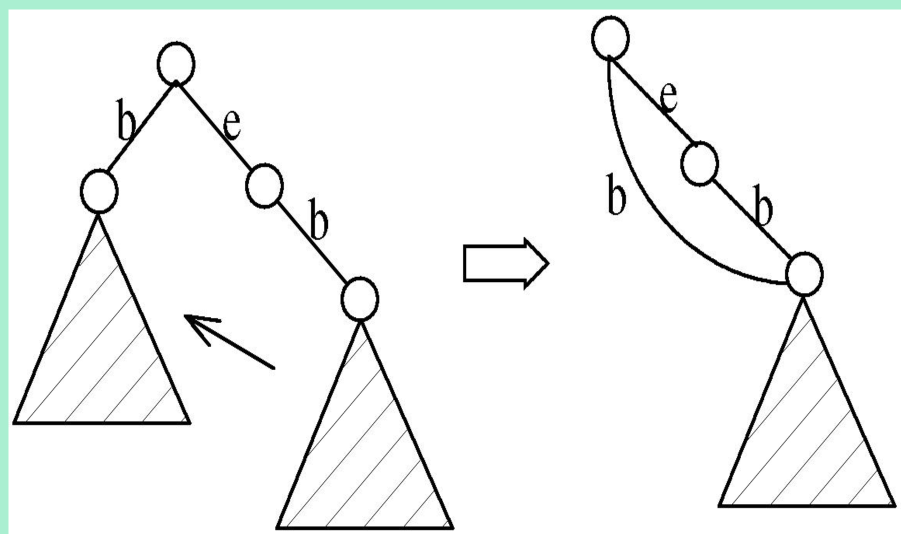
Pseudo-Projection vs. Physical Projection

- Pseudo-projection avoids physically copying postfixes
  - Efficient in running time and space when database can be held in main memory
- However, it is not efficient when database cannot fit in main memory
  - Disk-based random accessing is very costly
- Suggested Approach:
  - Integration of physical and pseudo-projection
  - Swapping to pseudo-projection when the data set fits in memory

CloSpan: Mining Closed Sequential Patterns

- A closed sequential pattern *s*: there exists no superpattern *s'* such that *s'* ɔ *s*, and *s'* and *s* have the same support
- Which one is closed? <abc>: 20, <abcd>:20, <abcde>: 15
- Why mine close seq. patterns?
  - Reduces the number of (redundant) patterns but attains the same expressive power
- Property: If *s'* ɔ *s*, closed iff two project DBs have the same size
- Using Backward Subpattern and Backward Superpattern pruning to prune redundant search space





Constraint-Based Seq.-Pattern Mining

- Constraint-based sequential pattern mining
  - Constraints: User-specified, for focused mining of desired patterns
  - How to explore efficient mining with constraints? — Optimization
- Classification of constraints
  - Anti-monotone: E.g., value_sum(S) < 150, min(S) > 10
  - Monotone: E.g., count (S) > 5, S $\supseteq$ {PC, digital_camera}
  - Succinct: E.g., length(S) $\geq$ 10, S $\in$ {Pentium, MS/Office, MS/Money}
  - Convertible: E.g., value_avg(S) < 25, profit_sum (S) > 160, max(S)/avg(S) < 2, median(S) – min(S) > 5
  - Inconvertible: E.g., avg(S) – median(S) = 0