PrefixSpan: Mining Sequential Patterns Efficiently by Prefix-Projected Pattern Growth

Authors:
Jian Pei, Jiawei Han, Behzad Mortazavi-Asi, Helen Pinto Qiming Chen,
Umeshwar Dayal, Mei-Chun Hsu

Presenter:
Wojciech Stach

Outline

- Mining Sequential Patterns
  - Problem statement
  - Definitions & examples
  - Strategies
- PrefixSpan algorithm
  - Motivation
  - Definitions & examples
  - Algorithm
  - Example
  - Performance study
- Conclusions

Sequential Pattern Mining

- Given
  - a set of sequences, where each sequence consists of a list of elements and each element consists of set of items
  - user-specified min_support threshold

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

<a(abc)(ac)d(cf)> - 5 elements, 9 items

<a(abc)(ac)d(cf)> - 9-sequence

<a(abc)(ac)d(cf)> = <a(cba)(ac)d(cf)>
<a(abc)(ac)d(cf)> ≠ <a(ac)(abc)d(cf)>

Sequential Pattern Mining

- Find all the frequent subsequences, i.e. the subsequences whose occurrence frequency in the set of sequences is no less than min_support

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

min_support = 2

Solution – 53 frequent subsequences

<a><aa> <ab> <a(bc)> <a(bc)a> <aba> <abc> <(ab)> <(ab)c> <(ab)d> <(ab)f> <(ab)dc> <ac> <aca> <acb> <acc> <ad> <adc> <af>

<b> <ba> <bc> <(bc)> <(bc)a> <bd> <bdc> <bf>

<c> <ca> <cb> <cc>

<d> <db> <dc> <dcb>

<e> <ea> <eab> <eac> <eacb> <eb> <ebc> <ec> <ecb> <ef> <efb> <efc> <efcb>

<f> <fb> <fbc> <fc> <fcb>

Subsequence vs. super sequence

- Given two sequences $\alpha=<a_1a_2\ldots a_n>$ and $\beta=<b_1b_2\ldots b_m>$
- α is called a subsequence of β, denoted as $\alpha\subseteq \beta$, if there exist integers $1\leq j_1<j_2<\ldots<j_n \leq m$ such that $a_1\subseteq b_{j1}$, $a_2 \subseteq b_{j2}, \ldots, a_n\subseteq b_{jn}$
- β is a super sequence of α

β =<a(abc)(ac)d(cf)>

α_1=<aa(ac)d(c)>

α_2=<(ac)(ac)d(cf)>

α_3=<ac>

~~β =<a(abc)(ac)d(cf)>~~

~~α_4=<df(cf)>~~

~~α_5=<(cf)d>~~

~~α_6=<(abc)dcf>~~

Sequence Support Count

- A sequence database is a set of tuples <sid, s>
- A tuple <sid, s> is said to contain a sequence α, if α is a subsequence of s, i.e., $\alpha \subseteq$s
- The support of a sequence α is the number of tuples containing α

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

α_1=<a> support(α_1) = 4

α_2=<ac> support(α_2) = 4

α_3=<(ab)c> support(α_3) = 2

Strategies

- Apriori-property based
  - AprioriSome (1995)
  - AprioriAll (1995)
  - DynamicSome (1995)
  - GSP (1996)

- Regular expression constraints
  - SPIRIT (1999)

- Data projection based
  - FreeSpan (2000)

Outline

- Mining Sequential Patterns
  - Problem statement
  - Definitions & examples
  - Strategies
- PrefixSpan algorithm
  - Motivation
  - Definitions & examples
  - Algorithm
  - Example
  - Performance study
- Conclusions

Motivation and Background

- Shortcomings of Apriori-like approaches
  - Potentially huge set of candidate sequences
  - Multiple scans of databases
  - Difficulties at mining long sequential patterns
- FreeSpan (**Fre**qu**e**nt pattern-projected **S**equential **pa**tter**n** mining) – pattern growth method
  - General idea is to use frequent items to recursively project sequence databases into a smaller projected databases and grow subsequence fragments in each projected database
- PrefixSpan (**Prefix**-projected **S**equential **pa**tter**n** mining)
  - Less projections and quickly shrinking sequences

Prefix

- Given two sequences $\alpha=<a_1a_2\ldots a_n>$ and $\beta=<b_1b_2\ldots b_m>$, $m\leq n$
- Sequence β is called a prefix of α if and only if:
  - $b_i = a_i$ for $i \leq m-1$;
  - $b_m \subseteq a_m$;
  - All the items in $(a_m - b_m)$ are alphabetically after those in b_m

α =<a(abc)(ac)d(cf)>

β =<a(abc)a>

~~α =<a(abc)(ac)d(cf)>~~

~~β =<a(abc)c>~~

Projection

- Given sequences α and β, such that β is a subsequence of α.
- A subsequence α' of sequence α is called a projection of α w.r.t. β prefix if and only if
  - α' has prefix β;
  - There exist no proper super-sequence α'' of α' such that α'' is a subsequence of α and also has prefix β

α =<a(abc)(ac)d(cf)>

β =<(bc)a>

α′ =<(bc)(ac)d(cf)>

Postfix

- Let $\alpha' =<a_1a_2\ldots a_n>$ be the projection of α w.r.t. prefix $\beta=<a_1a_2\ldots a_{m-1}a'_m>$ $(m \leq n)$
- Sequence $\gamma=<a''_m a_{m+1}\ldots a_n>$ is called the postfix of α w.r.t. prefix β, denoted as $\gamma= \alpha/ \beta$, where $a''_m=(a_m-a'_m)$
- We also denote $\alpha =\beta\cdot\gamma$

α′ =<a(abc)(ac)d(cf)>

β =<a(abc)a>

γ=<(_c)d(cf)>

PrefixSpan – Algorithm

- **Input**: A sequence database S, and the minimum support threshold min_sup
- **Output**: The complete set of sequential patterns
- **Method**: Call PrefixSpan(<>,0,S)
- **Subroutine** PrefixSpan(α, l, $S|_{\alpha}$)
- **Parameters**:
  - α: sequential pattern,
  - l: the length of α;
  - $S|_{\alpha}$: the α-projected database, if $\alpha \neq$<>; otherwise; the sequence database S.

PrefixSpan – Algorithm (2)

- **Method**

1. Scan $S|_{\alpha}$ once, find the set of frequent items b such that:
   a) b can be assembled to the last element of α to form a sequential pattern; or
   b) <b> can be appended to α to form a sequential pattern.
2. For each frequent item b, append it to α to form a sequential pattern α', and output α';
3. For each α', construct α'-projected database $S|_{\alpha'}$, and call PrefixSpan(α', l+1, $S|_{\alpha'}$).

PrefixSpan - Example

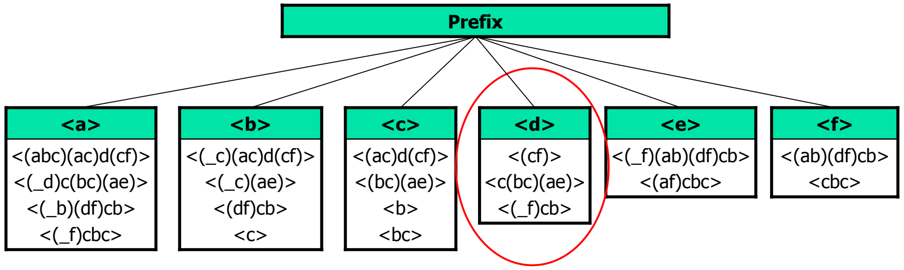
id	Sequence
10	<a(abc)(ac)d(cf)>
20	<(ad)c(bc)(ae)>
30	<(ef)(ab)(df)cb>
40	<eg(af)cbc>

min_support = 2

1. Find length-1 sequential patterns

<a>	<b>	<c>	<d>	<e>	<f>	<g>
4	4	4	3	3	3	1

2. Divide search space

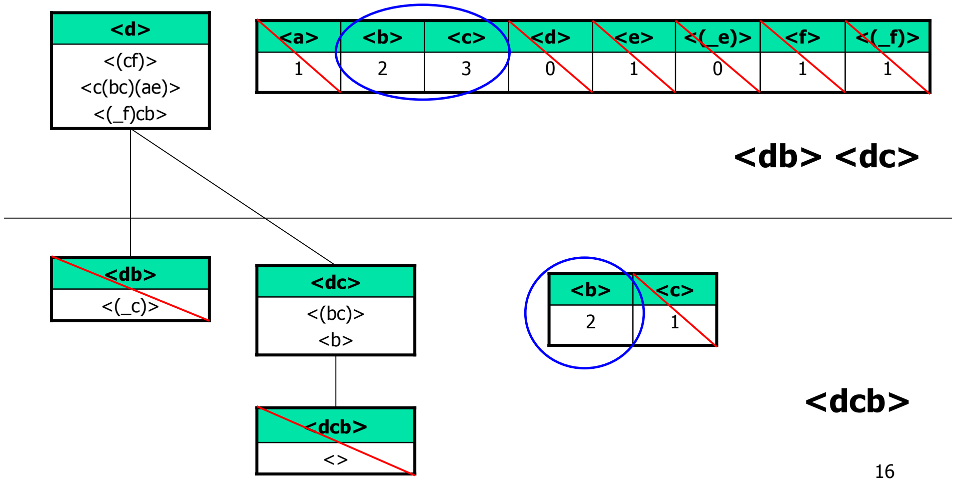


Prefix

<a>	<b>	<c>	<d>	<e>	<f>
<(abc)(ac)d(cf)> <(_d)c(bc)(ae)> <(_b)(df)cb> <(_f)cbc>	<(_c)(ac)d(cf)> <(_c)(ae)> <(df)cb> <c>	<(ac)d(cf)> <(bc)(ae)> <b> <bc>	<(cf)> <c(bc)(ae)> <(_f)cb>	<(_f)(ab)(df)cb> <(af)cbc>	<(ab)(df)cb> <cbc>

PrefixSpan – Example (2)

3. Find subsets of sequential patterns



<d>
<(cf)> <c(bc)(ae)> <(_f)cb>

<a>	<b>	<c>	<d>	<e>	<(_e)>	<f>	<(_f)>
1	2	3	0	1	0	1	1

<db> <dc>

<db>
<(_c)>

<dc>
<(bc)> <b>

<b>	<c>
2	1

<dcb>
<>

<dcb>

PrefixSpan - characteristics

- No candidate sequence needs to be generated by PrefixSpan
- Projected databases keep shrinking
- The major cost of PrefixSpan is the construction of projected databases
- ***How to reduce this cost?***

Different projection methods

- Bi-level projection
    - reduces the number and the size of projected databases
- Pseudo-Projection
    - reduces the cost of projection when projected database can be held in main memory

---

Bi-level Projection

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

min_support = 2

- Scan to get 1-length sequences
- Construct a triangular matrix instead of projected databases for each length-1 patterns

a	2					
b	(4,2,2)	1				
c	(4,2,1)	(3,3,2)	3			
d	(2,1,1)	(2,2,0)	(1,3,0)	0		
e	(1,2,1)	(1,2,0)	(1,2,0)	(1,1,0)	0	
f	(2,1,1)	(2,2,0)	(1,2,1)	(1,1,1)	(2,0,1)	1
	a	**b**	**c**	**d**	**e**	**f**

ALL length-2 sequential pattern

Support(<**ac**>) = 4
Support(<**ca**>) = 2
Support(<**(ac)**>) = 1

Support(<**cc**>) = 3

Bi-level projection (2)

- For each length-2 sequential pattern α, construct the α-projected database and find the frequent items
- Construct corresponding S-matrix

<ab>
<(_c)(ac)(cf)> <(_c)a> <c>

a	~~b~~	c	(_c)	~~d~~	~~(_d)~~	~~e~~	~~(_e)~~	~~f~~	~~(_f)~~
2	~~0~~	2	2	~~0~~	~~1~~	~~0~~	~~0~~	~~1~~	~~0~~

<aba> <abc> <a(bc)>

a	0		
c	(1,0,1)	1	
(_c)	(ϕ,2, ϕ)	(ϕ,1, ϕ)	ϕ
	a	**c**	**(_c)**

<a(bc)a>

Bi-level projection (3) - optimization

- "Do we need to include every item in a postfix in the projected databases?"
- NO! Item pruning in projected database by 3-way Apriori checking

<ac> is not frequent	**Any super-sequence of it can never be a sequential pattern**	c can be excluded from construction of <ab> - projected database
<a(bd)> is not frequent		To construct <a(bc)>-projected database, sequence <a(bcde)df> should be projected to <(_e)df> instead of <(_de)df>

Pseudo-Projection

- **Observation**: postfixes of a sequence often appear repeatedly in recursive projected databases
- **Method**: instead of constructing *physical* projection by collecting all the postfixes, we can use pointers referring to the sequences in the database as a pseudo-projection
- Every projection consists of two pieces of information: pointer to the sequence in database and offset to the postfix in the sequence

---

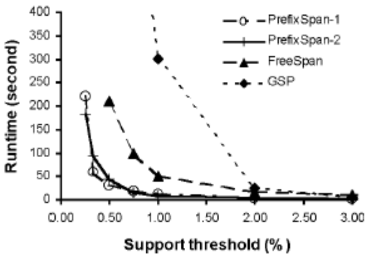
s1=<a(abc)(ac)d(cf)>

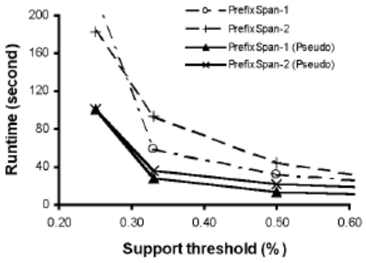
Pointer	Offset	Postfix
s1	2	<(abc)(ac)d(cf)>
s1	5	<(ac)d(cf)>
s1	6	<(_c)d(cf)>

Experimental Results

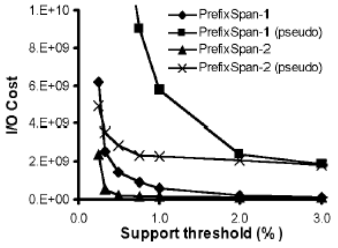
- Environment: 233MHz Pentium PC, 128 MB RAM, Windows NT, Visual C++ 6.0
- Reported test on synthetic data set: C10T8S8I8:
  - 1000 items
  - 10000 sequences
  - Average number of items within elements: 8
  - Average number of elements in a sequence: 8
- Competitors:
  - GSP
  - FreeSpan
  - PrefixSpan-1 (level-by-level projection)
  - PrefixSpan-2 (bi-level projection)

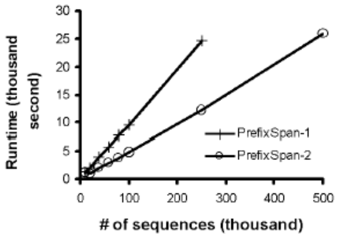
Runtime vs. support threshold





I/O costs vs. threshold and scalability





Outline

- Mining Sequential Patterns
  - Problem statement
  - Definitions & examples
  - Strategies
- PrefixSpan algorithm
  - Motivation
  - Definitions & examples
  - Algorithm
  - Example
  - Performance study
- Conclusions

Conclusions

- PrefixSpan
  - Efficient pattern growth method
  - Outperforms both GSP and FreeSpan
  - Explores prefix-projection in sequential pattern mining
  - Mines the complete set of patterns but reduces the effort of candidate subsequence generation
  - Prefix-projection reduces the size of projected database and leads to efficient processing
  - Bi-level projection and pseudo-projection may improve mining efficiency

References

- Pei J., Han J., Mortazavi-Asl J., Pinto H., Chen Q., Dayal U., Hsu M., PrefixSpan: Mining Sequential Patterns Efficiently by Prefix-Projected Pattern Growth, 17th International Conference on Data Engineering (ICDE), April 2001
- Agrawal R., Srikant R., Mining sequential patterns, Proceedings 1995 Int. Conf. Very Large Data Bases (VLDB'94), pp. 487-499, 1995
- Han J., Dong G., Mortazavi-Asl B., Chen Q., Dayal U., Hsu M.-C., Freespan: Frequent pattern-projected sequential pattern mining, Proceedings 2000 Int. Conf. Knowledge Discovery and Data Mining (KDD'00), pp. 355-359, 2000
- Srikant R., Agrawal R., Mining sequential pattern: Generalizations and performance improvements, Proceedings 5th Int. /conf. Extending Database Technology (EDBT'96), pp. 3-17, 1996
- Zhao Q., Bhowmick S. S., Sequential Pattern Mining: A Survey. Technical Report Center for Advanced Information Systems, School of Computer Engineering, Nanyang Technological University, Singapore, 2003

THANK YOU !!!

Any Questions?