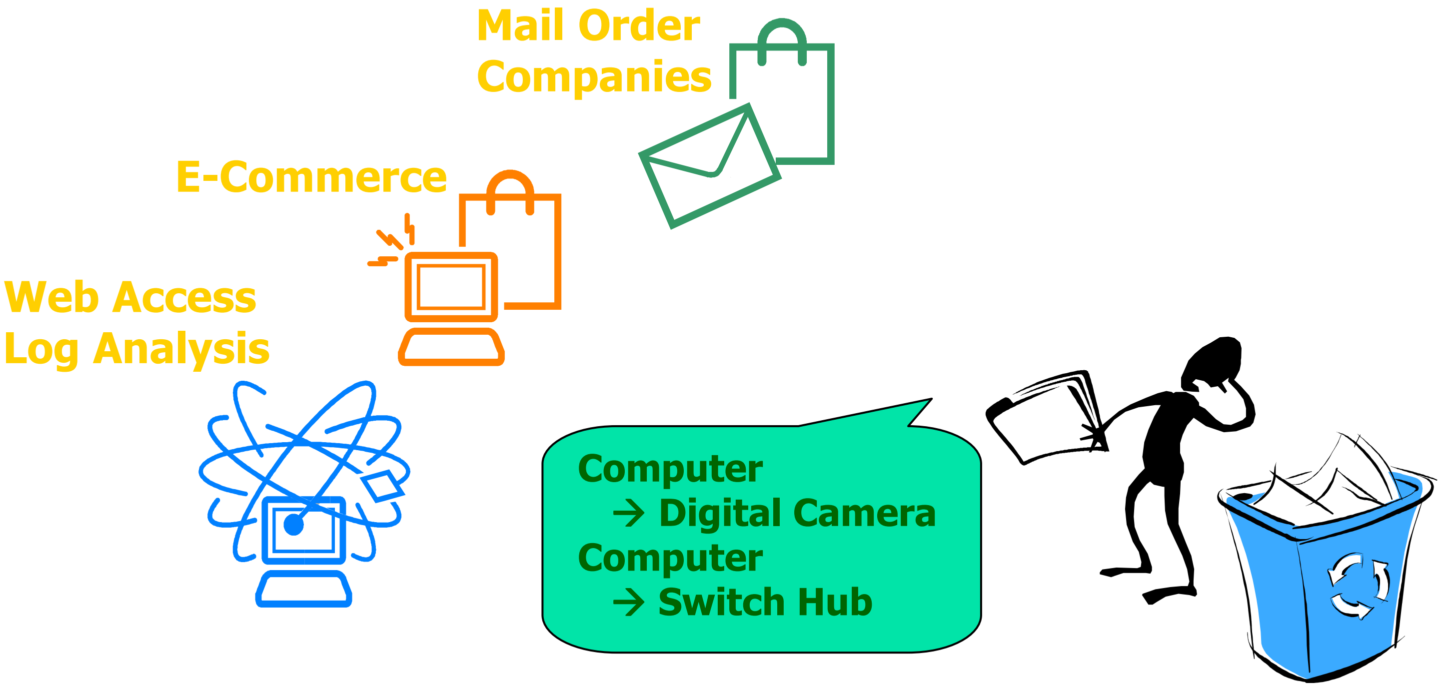
Association Rules and Sequential Pattern Mining

Kyuseok Shim
Seoul National University

Motivation

- Many applications require sequential pattern mining
  - Datasets typically include quantity information
  - However, traditional techniques cannot take it into account
  - Quantity information can provide useful insights to the users



Outline

- Association Rule Mining Algorithms
- Sequential Pattern Algorithms
- Summary

Association Rules

Association Rules

- Given:
  - A database of customer transactions
  - Each transaction is a set of items
- Find all rules $X \Rightarrow Y$ that correlate the presence of one set of items X with another set of items Y
  - Example: 98% of people who purchase diapers and baby food also buy beer.
  - Any number of items in the consequent/antecedent of a rule
  - Possible to specify constraints on rules (e.g., find only rules involving expensive imported products)

Association Rules

- Sample Applications
  - Market basket analysis
  - Attached mailing in direct marketing
  - Fraud detection for medical insurance
  - Department store floor/shelf planning

Problem Decomposition

1. Find all sets of items that have minimum support
   - Most expensive phase
   - Lots of research
2. Use the frequent itemsets to generate the desired rules
   - Generation is straight forward

Support and Confidence

- $X \rightarrow Y$ [support, confidence]

$$\text{support} = \frac{\#\text{ of transactions containing all the items in } X \cup Y}{\text{total }\#\text{ of transactions in the database}}$$

$$\text{confidence} = \frac{\#\text{ of transactions that contain both } X \text{ and } Y}{\#\text{ of transactions containing } X}$$

- For min_support = 50%, min_confidence = 50%
  - B => C with 50% support and 66% confidence

TID	Items
10	a, c, d
20	b, c, e
30	a, b, c, e
40	b, e

The Apriori Algorithm : Key Observation

- Every subset of a frequent itemset is also frequent itemset.
  - If {beer, diaper, nuts} is frequent, {beer, diaper} must be frequent.
- If there is any itemset which is infrequent, its superset will not be generated!
  - A powerful candidate set pruning technique.

An Apriori Example

Min_sup=2

Data base D

TID	Items
10	a, c, d, f
20	b, c, e
30	a, b, c, e,
40	b, e
50	a, f

Scan D

1-candidates

Itemset	Sup
a	3
b	3
c	3
d	1
e	3
f	2

Freq 1-itemsets

Itemset	Sup
a	3
b	3
c	3
e	3
f	2

2-candidates

Itemset
ab
ac
ae
af
bc
be
bf
ce
cf

Scan D

Itemset	Sup
ab	1
ac	2
ae	1
af	2
bc	2
be	3
bf	0
ce	2

Freq 2-itemsets

Itemset	Sup
ac	2
af	2
bc	2
be	3
ce	2

3-candidates

Itemset
acf
bce

pruning

3-candidates

Itemset
bce

Scan D

Itemset	Sup
bce	2

Freq 3-itemsets

Itemset	Sup
bce	2

- From [Agrawal, Srikant 1994]

The Apriori Algorithm

- C_k: Candidate itemset of size k
- L_k : frequent itemset of size k

- L_1 = {frequent items};
- for ($k = 1$; $L_k \mathrel{!}= \varnothing$; k++) do
  - C_{k+1} = candidates generated from L_k;
  - for each transaction t in database do increment the count of all candidates in C_{k+1} that are contained in t
  - L_{k+1} = candidates in C_{k+1} with min_support
- return $\cup_k L_k$;

Basic Sequential Pattern Algorithm

What is sequential pattern?

- Customers typically rent “Star Wars”, then “Empire Strikes Back”, and the “Return of the Jedi”.
- Useful time-related or ordered sequential pattern results apply to many scientific and business domains
  - Customer purchase behavior
  - Web access patterns
  - Scientific experiments
  - Disease treatments
  - DNA sequences

Problem Statement

Sequence Database

Customer ID	Customer Sequence
1	<(30) (90)>
2	<(10 20) (30) (40 60 70)>
3	<(30 50 70)>
4	<(30) (40 70) (90)>
5	<(90)>

<(30) (90)> ≡ <30 90>

Length = 6

Sequence : <(10 20) (30) (40 60 70)>

itemset/element/transaction(TID)

subsequence

super sequence

<30 90> ⊑ <30 (40 70) 90>

<30 (40 70) 90> contains <30 90>

- Support : the number(ratio) of tuples in database containing the sequence
- Min_support : user-specified support threshold
- Sequential pattern : the sequence is contained by at least min_support

Problem Definition

- Given a sequence database and a min_support, to find the complete set of frequent sequential patterns in the database.
- In the previous example,

Sequential patterns with support >= 20%
<30 90>
<30 (40 70)>

An Example

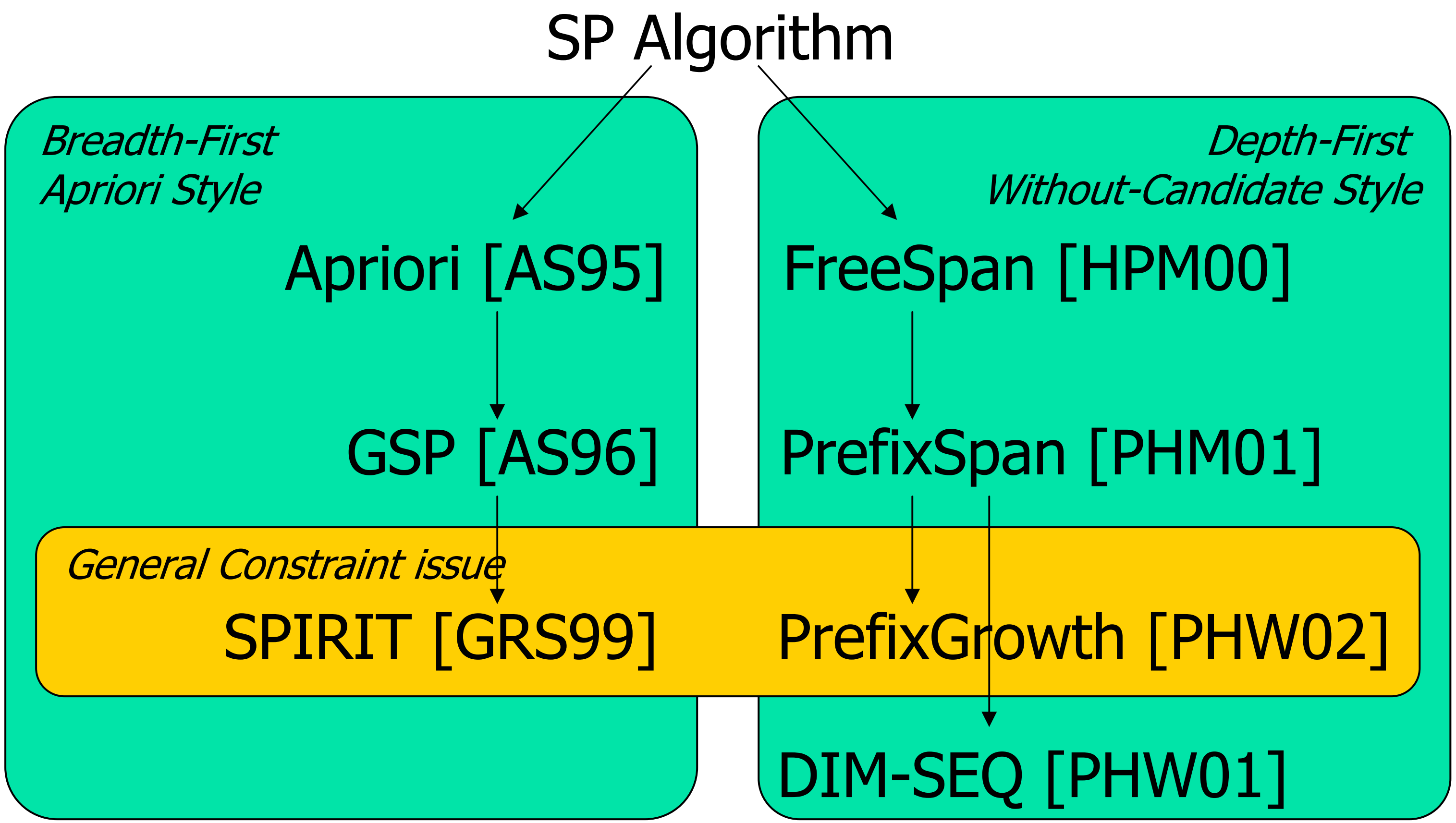
- Apriori heuristic
  - Any super-pattern of a non-frequent pattern cannot be frequent
- Breadth first algorithm (Given min_support = 50%)

SID	Sequence
1	<c c>
2	<(a b) c d>
3	<d c>
4	<c d>
5	<c a (b d) c>
6	<c d c>

Length	Candidates	Sequential patterns
1	Ø	<c>, <d>
2	<a a> <b b> <c c> <d d> <a b> <b a> <(a b)> <a c> <c a> <(a c)> <a d> <d a> <(a d)> <b c> <c b> <(b c)> <b d> <d b> <(b d)> <c d> <d c> <(c d)>	<c d> <c c> <d c>
3	<c c d> ...	Ø

Overview

- The Roadmap of topics discussed in this tutorial



GSP (1)

- General structure is similar to that of Apriori sequence phase.
- Key Operations
  - Candidate generation
  - Counting candidates
  - Processing taxonomies

GSP (2)

- Candidate generation
  - Join condition
    - If the subsequence obtained by dropping the first item of s_1 is the same as the subsequence obtained by dropping the last item of s_2
    - e.g.
      - < (10 20) 30 40 >, < 20 30 40 50 >
      - < 10 20 (30 40) >, < 20 (30 40 50) >
      - < 10 >, < 20 >
  - Join operation
    - The sequence s1 extended with the last item in s2.
    - The added item becomes a separate element if it was a separate element in s2, and part of the last element of s1 otherwise.
    - e.g.
      - < (10 20) 30 40 >, < 20 30 40 50 > → < (10 20) 30 40 50 >
      - < 10 20 (30 40) >, < 20 (30 40 50) > → < 10 20 (30 40 50) >
      - < 10 >, < 20 > → < 10 20 > , < (10 20) >   ∵<(N)20>, <(N 20)>

PrefixSpan (1)

- J. Pei, J. Han, B. Mortazavi-Asl, H. Pinto, Q. Chen, U. Dayal and M. Hsu [PHM01]
- Depth first & Divide and conquer algorithm

PrefixSpan (2)

- J. Pei, J. Han, B. Mortazavi-Asl, H. Pinto, Q. Chen, U. Dayal and M. Hsu [PHM01]
- Depth first & Divide and conquer algorithm
- PrefixSpan vs. FreeSpan
  - Only prefix-based projection : less projections and quickly shrinking the projected DB
- PrefixSpan vs. GSP
  - PrefixSpan makes no candidate.
  - The longer the sequence patterns, the larger the candidates GSP has.
  - However, PrefixSpan makes projections as many as frequent patterns, therefore the performance of PrefixSpan is dependant on projection cost.

PrefixSpan (3)

- Given a sequence α = <$e_1 e_2 \cdots e_n$>,
  - a sequence β = < $e'_1 e'_2 \cdots e'_m$ > ($m \leq n$) is a <u>prefix</u> of α if and only if $e'_i = e_i$ for ($i \leq m-1$) , $e'_m \subseteq e_m$, and all the items in ($e_m - e'_m$) are alphabetically after those in e'_m.
  - when $\beta \sqsubseteq \alpha$, subsequence α' of α is a <u>projection</u> of α w.r.t. prefix β if and only if α' has prefix β and there exists no proper super-sequence α'' of α' such that α'' is a subsequence of α and also has prefix β.
  - sequence γ = <$e_m e_{m+1} \cdots e_n$> is the <u>postfix</u> of α w.r.t. prefix β, where $e''_m = (e_m - e'_m)$, denoted as $\gamma = \alpha/\beta$ or $\alpha = \beta \cdot \gamma$

α = <a(abc)(bc)d(acf)>, β = <ab>
α' = the projection of α w.r.t. β = <a(bc)(bc)d(acf)>
$\gamma = \alpha'/\beta$ = <(_c)(bc)d(acf)>
β is a prefix of α', and γ is the postfix of α' w.r.t. β.

PrefixSpan (4)

- Outline of the method

PrefixSpan(α, l, $S|_{\alpha}$)

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.

For each frequent item b, append it to α to form a sequential pattern α', and output α';

For each α', construct α'-projected database $S|_{\alpha'}$, and call PrefixSpan(α', $l+1$, $S|_{\alpha'}$)

PrefixSpan (5)

- Example (Given min_support = 50%)

SID	Sequence Database
10	<(a b) c>
20	<b (c a)>
30	<d c (a b)>

1-frequent sequence
<a>, <b>, <c>

Having prefix <a>

SID	Projected Database
10	<(_b) c>
30	<(_b)>

1-frequent sequence
<_b>

Having prefix <(ab)>

SID	Projected Database
10	<c>

1-frequent sequence
∅

Having prefix <b>

SID	Projected Database
10	<c>
20	<(c a)>

1-frequent sequence
<c>

Having prefix <bc>

SID	Projected Database
20	<(_a)>

1-frequent sequence
∅

Having prefix <c>

SID	Projected Database
20	<(_a)>
30	<(a b)>

1-frequent sequence
∅

PrefixSpan (6)

- Scaling up techniques
  - Bi-level projection
    - To reduce the number of projection, bi-level algorithm construct projection not by 1-sequences, but by 2-sequences.
    - To get 2-sequences, bi-level use a GSP-like method.
    - This method reduces the number of projection, but it makes a number of candidates.
  - Pseudo-projection
    - If the projected database fits in main memory, instead of constructing a physical projection, pseudo-projection uses pointers as a pseudo-projection.

Summary

- Association rule mining and Sequential pattern mining have interesting applications
- Breadth-first and Depth first style algorithms are developed
- Maximal patterns were introduced for compact representations

References (1)

- [AS95] R. Agrawal and R. Srikant. “Mining sequntial patterns,” ICDE, 1995.
- [AS96] R. Agrawal and R. Srikant. “Mining Sequential patterns: Generalizations and performance improvements,” EDBT, 1996.
- [GRS99] M. Garofalakis, R. Rastogi, and K. Shim. “Spirit: Sequential pattern mining with regular expression constraints,” VLDB, 1999.
- [HPM00]J. Han, J. Pei , B. Mortazavi-Asl, H. Chen, U. Dayal and M. Hsu. “FreeSpan: frequent pattern-projected sequential pattern mining,” KDD, 2000.

References (2)

- [PHM01] J. Pei, J. Han, B. Mortazavi-Asl, H. Pinto, Q. Chen, U. Dayal and M. Hsu. “PrefixSpan: Mining Sequential Patterns Efficiently by Prefix-Projected Pattern Growth,” ICDE, 2001.
- [PHP01] H. Pinto, J. Han, J. Pei, K. Wang, Q. Chen, U. Dayal. “Multi-Dimensional Sequential Pattern Mining,” CIKM, 2001.
- [PHW02] J. Pei, J. Han, and W. Wang. “Mining sequential patterns with constraints in large databases,” CIKM, 2002