Mining Frequent Patterns, Association and Correlations: Basic Concepts and Methods

CIS 468: Spring 2015

What is Frequent Pattern Analysis

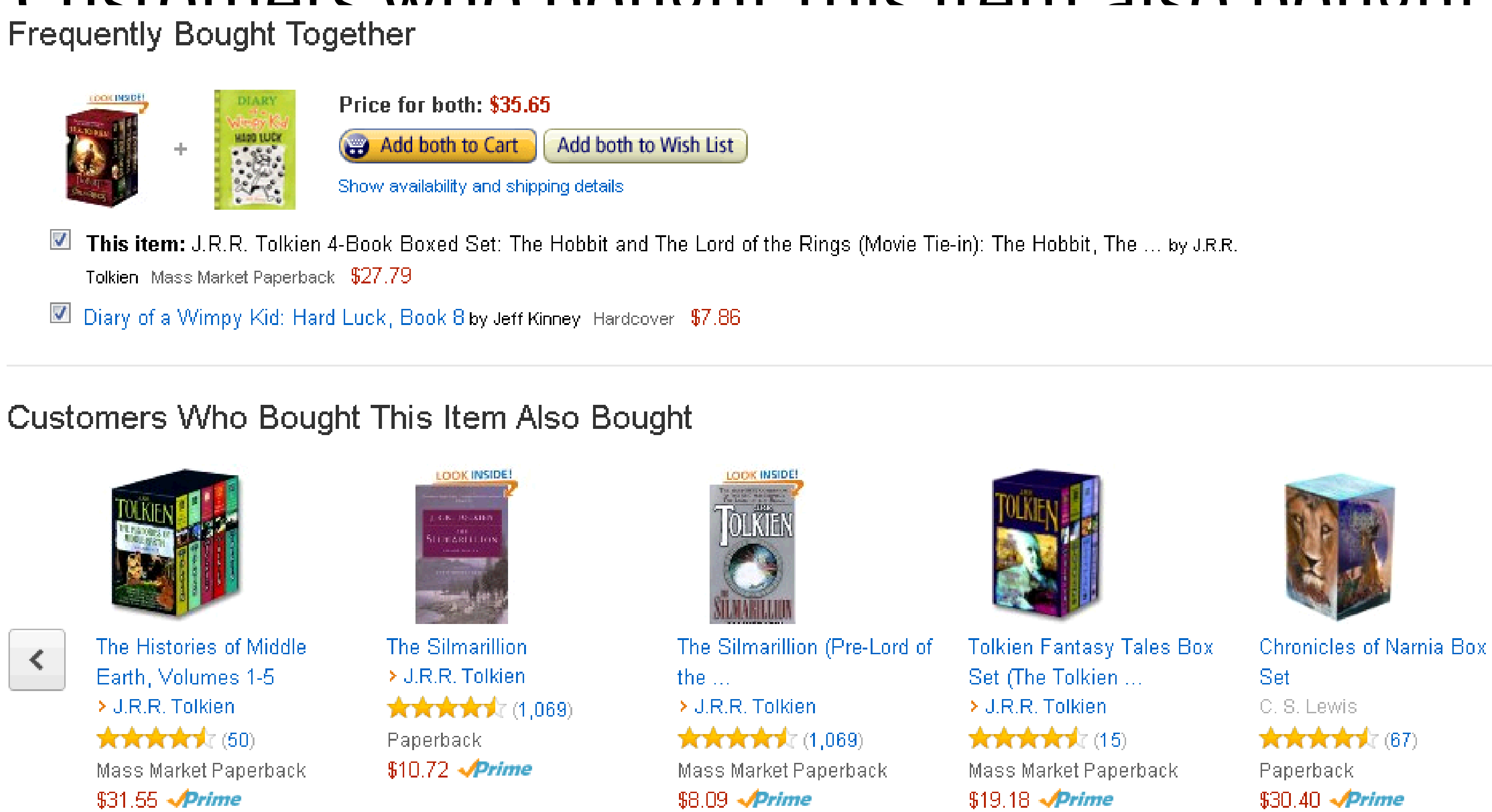
- Frequent pattern: a pattern (a set of items, subsequences, substructures, etc.) that occurs frequently in a data set
- First proposed by Agrawal, Imielinski, and Swami [AIS93] in the context of frequent itemsets and association rule mining
- Motivation: Finding inherent regularities in data
  - What products were often purchased together?— Beer and diapers?!
  - What are the subsequent purchases after buying a PC?
  - What kinds of DNA are sensitive to this new drug?
  - Can we automatically classify web documents?
- Applications
  - Basket data analysis, cross-marketing, catalog design, sale campaign analysis, Web log (click stream) analysis, and DNA sequence analysis.

Why is Frequent Pattern Mining Important?

- Freq. pattern: An intrinsic and important property of datasets
- Foundation for many essential data mining tasks
  - Association, correlation, and causality analysis
  - Sequential, structural (e.g., sub-graph) patterns
  - Pattern analysis in spatiotemporal, multimedia, time-series, and stream data
  - Broad applications

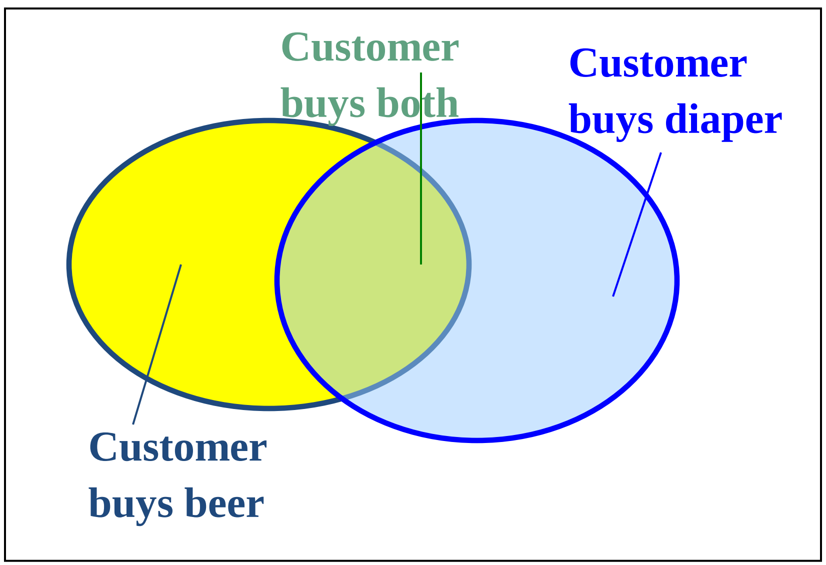
Frequent Itemsets

- Frequently Bought Together
- Customers who bought this item also bought



Basic Concepts: Frequent Patterns

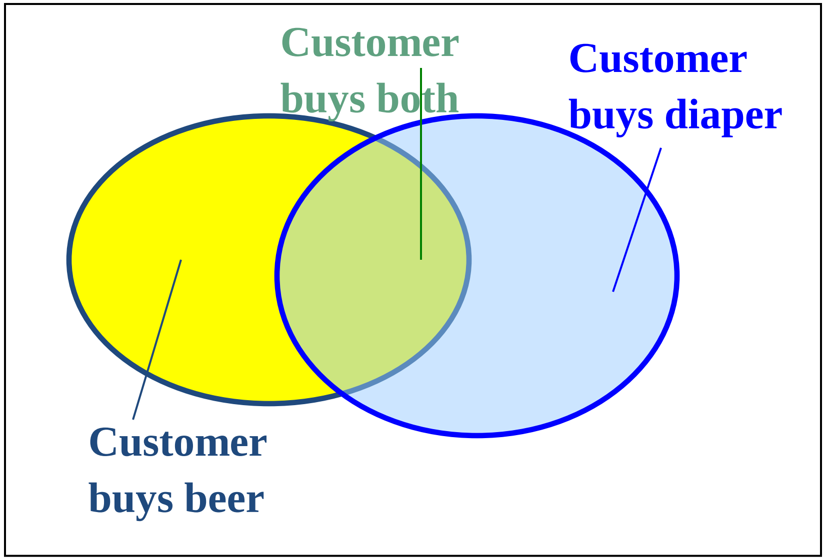
Tid	Items bought
10	Beer, Nuts, Diaper
20	Beer, Coffee, Diaper
30	Beer, Diaper, Eggs
40	Nuts, Eggs, Milk
50	Nuts, Coffee, Diaper, Eggs, Milk



- itemset: A set of one or more items
- k-itemset $X = \{x_1, \ldots, x_k\}$
- *(absolute) support*, or, *support count* of X: Frequency or occurrence of an itemset X
- *(relative) support*, *s*, is the fraction of transactions that contains X (i.e., the probability that a transaction contains X)
- An itemset X is *frequent* if X's support is no less than a *minsup* threshold

Basic Concepts: Association Rules

Tid	Items bought
10	Beer, Nuts, Diaper
20	Beer, Coffee, Diaper
30	Beer, Diaper, Eggs
40	Nuts, Eggs, Milk
50	Nuts, Coffee, Diaper, Eggs, Milk



- Find all the rules *X* → *Y* with minimum support and confidence
  - support, *s*, probability that a transaction contains X ∪ Y
  - confidence, *c*, conditional probability that a transaction having X also contains *Y*

Let minsup = 50%, minconf = 50%

Freq. Pat.: Beer:3, Nuts:3, Diaper:4, Eggs:3, {Beer, Diaper}:3

- Association rules: (many more!)
  - *Beer* → *Diaper* (60%, 100%)
  - *Diaper* → *Beer* (60%, 75%)

Closed Patterns and Max Patterns

- A long pattern contains a combinatorial number of sub-patterns
- Solution: *Mine closed itemsets and max-itemsets instead*
- An itemset X is closed if X is *frequent* and there exists *no super-pattern* $Y \supset X$, *with the same support* as X
- An itemset X is a max-itemset if X is frequent and there exists no frequent super-pattern $Y \supset X$
- Closed pattern is a lossless compression of freq. patterns
  - Reducing the # of patterns and rules

Closed Itemset

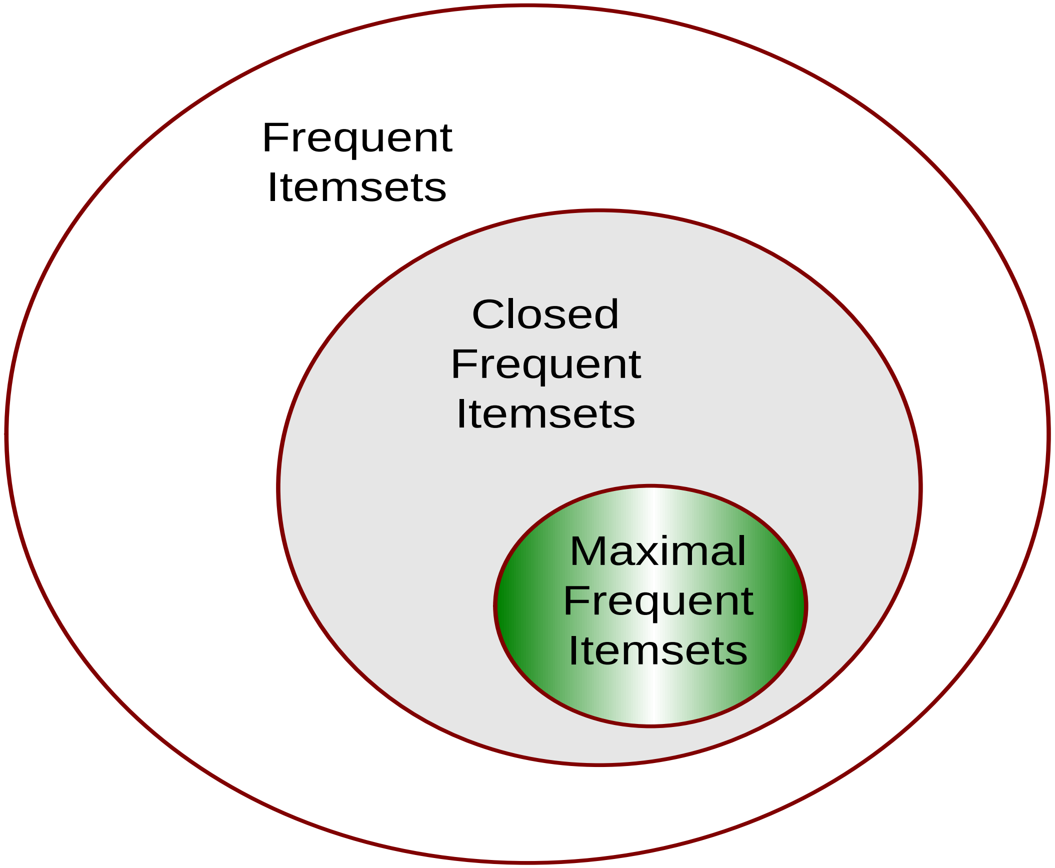
- Problem with maximal frequent itemsets:
  - Support of their subsets is not known – additional DB scans are needed
- An itemset is closed if none of its immediate supersets has the same support as the itemset

TID	Items
1	{A,B}
2	{B,C,D}
3	{A,B,C,D}
4	{A,B,D}
5	{A,B,C,D}

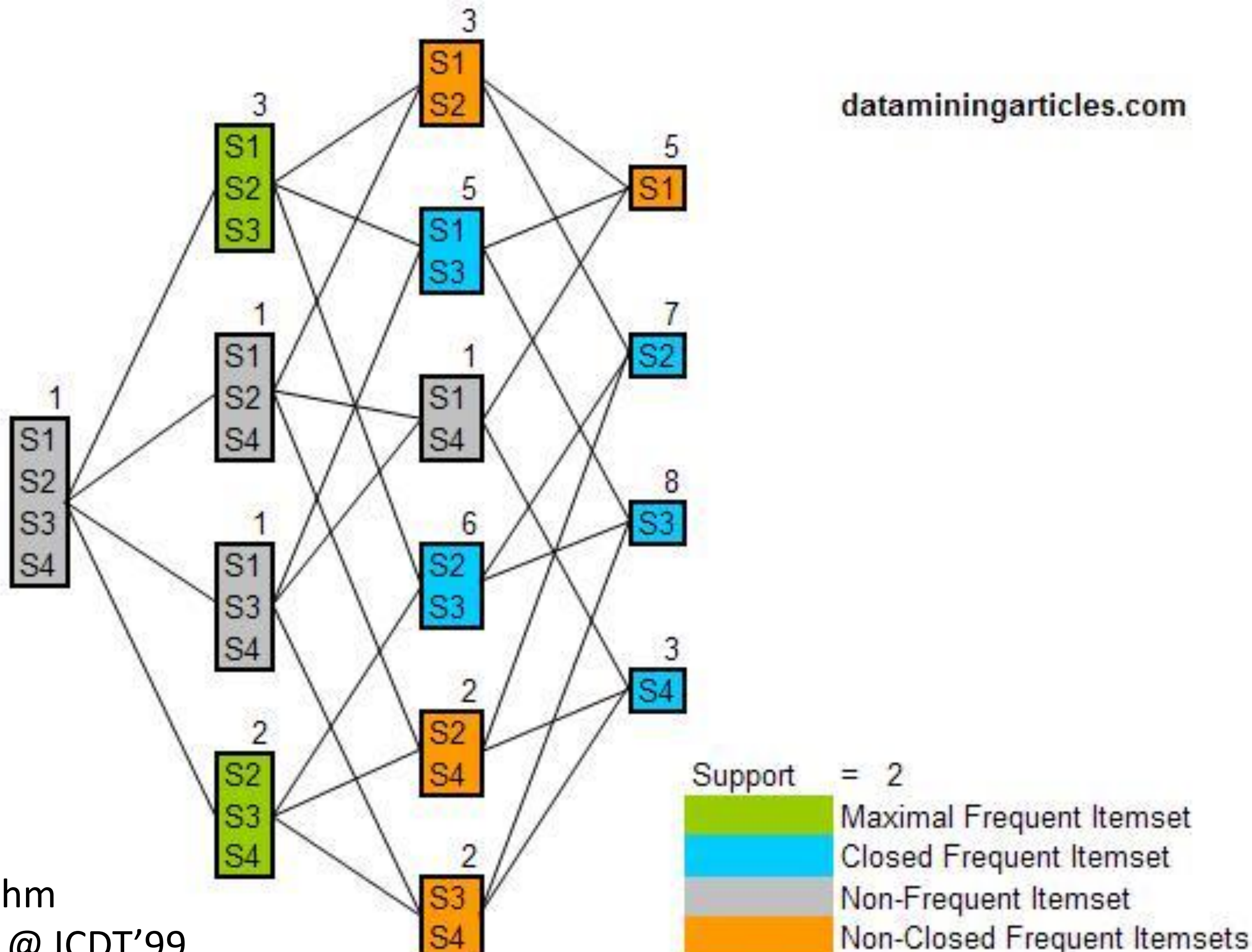
Itemset	Support
{A}	4
{B}	5
{C}	3
{D}	4
{A,B}	4
{A,C}	2
{A,D}	3
{B,C}	3
{B,D}	4
{C,D}	3

Itemset	Support
{A,B,C}	2
{A,B,D}	3
{A,C,D}	2
{B,C,D}	2
{A,B,C,D}	2

Maximal vs. Closed Itemsets



Maximal and Closed Frequent Itemsets



A-Close Algorithm
Pasquier, et al. @ ICDT'99

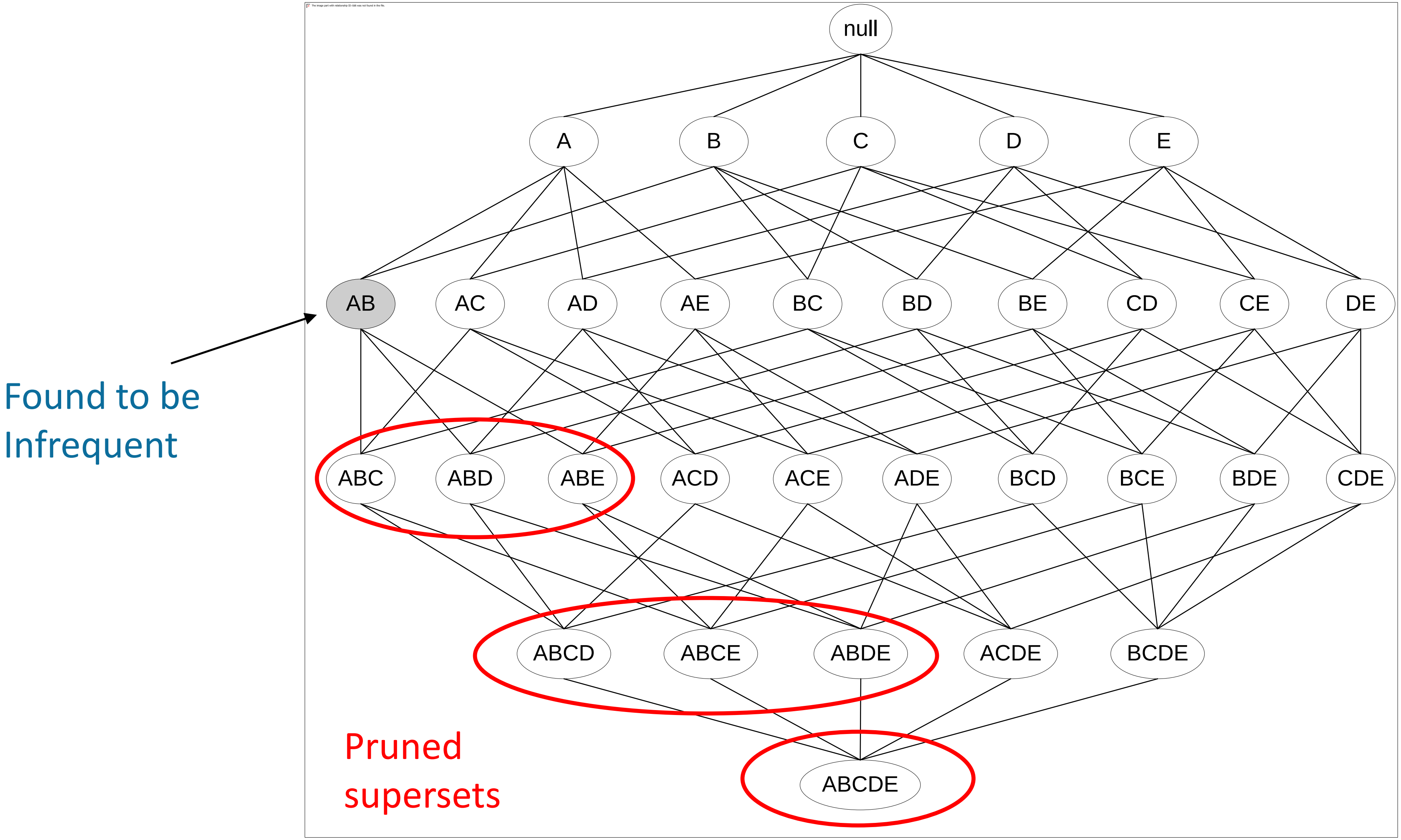
The Downward Closure Property and Scalable Mining Methods

- The downward closure property of frequent patterns
  - Any subset of a frequent itemset must be frequent
  - If {beer, diaper, nuts} is frequent, so is {beer, diaper} and {beer,nuts}
- The Apriori algorithm thrives on this property

Apriori: A Candidate Generation & Test Approach

- Apriori pruning principle: If there is any itemset which is infrequent, its superset should not be generated/tested! (Agrawal & Srikant @VLDB'94, Mannila, et al. @ KDD' 94)
- Method:
  - Initially, scan DB once to get frequent 1-itemset
  - Generate length (k+1) candidate itemsets from length k frequent itemsets
  - Test the candidates against DB
  - Terminate when no frequent or candidate set can be generated

Apriori Principle



Apriori Principle

Items (1-itemsets)

Item	Count
Bread	**4**
Cola	**2**
Milk	**4**
Beer	**3**
Diaper	**4**
Eggs	**1**

Pairs (2-itemsets)

(No need to generate candidates involving Cola or Eggs)

Itemset	Count
{Bread,Milk}	**3**
{Bread,Beer}	**2**
{Bread,Diaper}	**3**
{Milk,Beer}	**2**
{Milk,Diaper}	**3**
{Beer,Diaper}	**3**

Minimum Support = 3

Triplets (3-itemsets)

Itemset	Count
{Bread,Milk,Diaper}	**3**

If every subset is considered,

$$^6C_1 + {}^6C_2 + {}^6C_3 = 41$$

With support-based pruning,

$$6 + 6 + 1 = 13$$

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 !=$\emptyset$; k++) **do begin**
  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
  end
return $\cup_k L_k$;

The Apriori Algorithm (Example)

Mango – M
Onion – O
Nintendo – N
Key-chain – K
Eggs – E
Yo-yo – Y
Doll – D
Apple – A
Umbrella – U
Corn – C
Ice-cream - I

Transaction ID	Items Bought
T1	{M, O, N, K, E, Y }
T2	{D, O, N, K, E, Y }
T3	{M, A, K, E}
T4	{M, U, C, K, Y }
T5	{C, O, O, K, I, E}

The Apriori Algorithm (Step 1)

- Count the <u>number of transactions</u> in which each item (

Mango – M
Onion – O
Nintendo – N
Key-chain – K
Eggs – E
Yo-yo – Y
Doll – D
Apple – A
Umbrella – U
Corn – C
Ice-cream - I

Item	No of transactions
M	3
O	3
N	2
K	5
E	4
Y	3
D	1
A	1
U	1
C	2
I	1

Apriori Algorithm (Step 2)

- Let's say that an item is frequent if it occurs in 60% of the transactions.
- In step 2 we simply remove any items that are bought less than 3 times.

Mango – M
Onion – O
Nintendo – N
Key-chain – K
Eggs – E
Yo-yo – Y
Doll – D
Apple – A
Umbrella – U
Corn – C
Ice-cream - I

Item	Number of transactions
M	3
O	3
K	5
E	4
Y	3

Apriori Algorithm (Step 3)

- Start making pairs from the first item (M) and the second item (O) and so on...

Mango – M
Onion – O
Nintendo – N
Key-chain – K
Eggs – E
Yo-yo – Y
Doll – D
Apple – A
Umbrella – U
Corn – C
Ice-cream - I

Item pairs
MO
MK
ME
MY
OK
OE
OY
KE
KY
EY

Apriori Algorithm (Step 4)

- Count how many times each pair appears together in a transaction

Mango – M
Onion – O
Nintendo – N
Key-chain – K
Eggs – E
Yo-yo – Y
Doll – D
Apple – A
Umbrella – U
Corn – C
Ice-cream - I

Item Pairs	Number of transactions
MO	1
MK	3
ME	2
MY	2
OK	3
OE	3
OY	2
KE	4
KY	3
EY	2

Apriori Algorithm (Step 5)

- Remove all the L2 transaction pairs that occur less than 3 times and we are left with the following:

Mango – M
Onion – O
Nintendo – N
Key-chain – K
Eggs – E
Yo-yo – Y
Doll – D
Apple – A
Umbrella – U
Corn – C
Ice-cream - I

Item Pairs	Number of transactions
MK	3
OK	3
OE	3
KE	4
KY	3

Apriori Algorithm (Step 6)

- Form sets of three items using the self join rule.
- For each item pair we find two items with the same first item and join them
  - OK and OE = OKE

Item Set	Number of transactions
OKE	3
KEY	2

Association Rules

Body ==> *Consequent* [*Support* , *Confidence*]

- *Body*: represents the examined data.
- *Consequent*: represents a discovered property for the examined data.
- *Support*: represents the percentage of the records satisfying the *body* or the *consequent*.
- *Confidence*: represents the percentage of the records satisfying both the *body* and the *consequent* to those satisfying only the *body*.

Rule Generation

- Given a frequent itemset L, find all non-empty subsets $f \subset L$ such that $f \rightarrow L - f$ satisfies the minimum confidence requirement
  - If {O,K,E} is a frequent itemset, candidate rules:

    {O,K} $\rightarrow${E}, {O,E} $\rightarrow${K}, {K,E} $\rightarrow${O}, {K} $\rightarrow$ {O,E}, {E} $\rightarrow$ {O,K}, {O} $\rightarrow$ {K,E}, {O} $\rightarrow$ {K}, {O} $\rightarrow$ {E}, {K} $\rightarrow$ {O}, {K} $\rightarrow$ {E}, {E} $\rightarrow$ {O}, {E} $\rightarrow$ {K}

- If $|L| = k$, then there are $2^k - 2$ candidates association rules (ignoring $L \rightarrow \varnothing$ and $\varnothing \rightarrow L$)

Confidence and Association Rules

$\{O,K\} \rightarrow \{E\}$

$\{O,E\} \rightarrow \{K\}$

$\{K,E\} \rightarrow \{O\}$

$\{K\} \rightarrow \{O,E\}$

$\{E\} \rightarrow \{O,K\}$

$\{O\} \rightarrow \{K,E\}$

$\{O\} \rightarrow \{K\}$

$\{O\} \rightarrow \{E\}$

$\{K\} \rightarrow \{O\}$

$\{K\} \rightarrow \{E\}$

$\{E\} \rightarrow \{O\}$

$\{E\} \rightarrow \{K\}$

Body ==> Consequent [*Support* , *Confidence*]

Confidence = Body and Consequent / Body

Transaction ID	Items Bought
T1	{M, O, N, K, E, Y }
T2	{D, O, N, K, E, Y }
T3	{M, A, K, E}
T4	{M, U, C, K, Y }
T5	{C, O, O, K, I, E}

Rule Generation

- How to efficiently generate rules from frequent itemsets?
  - In general, confidence does not have an anti-monotone property

    $c(ABC \rightarrow D)$ can be larger or smaller than $c(AB \rightarrow D)$

  - But confidence of rules generated from the same itemset has an anti-monotone property
  - e.g., L = {A,B,C,D}:

$$c(ABC \rightarrow D) \geq c(AB \rightarrow CD) \geq c(A \rightarrow BCD)$$

    - Confidence is anti-monotone w.r.t. number of items on the RHS of the rule

Rule Generation

Lattice of rules