TEXAS A&M UNIVERSITY
Engineering

ECEN 758 Data Mining and Analysis: Lecture 5, Frequent Itemset Mining and Association Rules

Joshua Peeples, Ph.D.
Assistant Professor
Department of Electrical and Computer Engineering

Announcements

TEXAS A&M UNIVERSITY
Engineering

- Assignment #1 due this Friday (09/06)
  - Please upload submission as single PDF

Last Lecture

TEXAS A&M UNIVERSITY
Engineering

- Dimensionality reduction

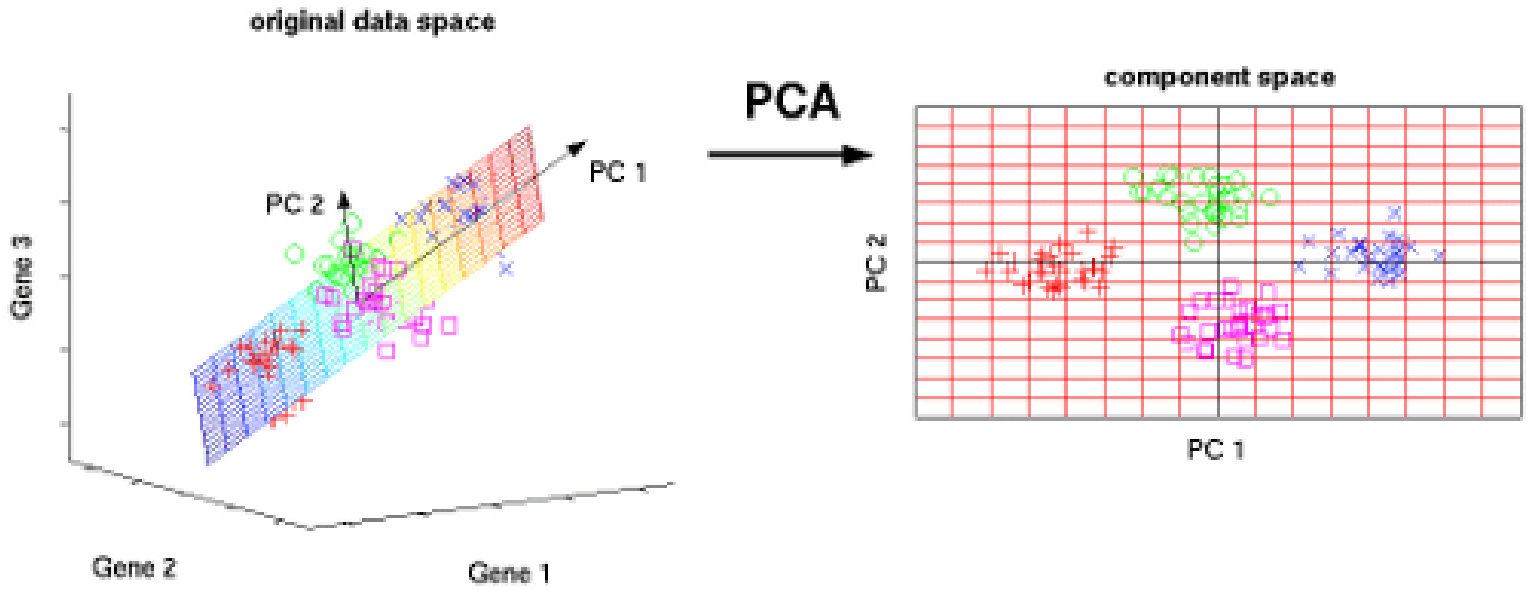


Image from: M. Scholz, Approaches to analyse and interpret biological profile data.

Today

- Frequent Item Set Mining and Association Rules
- Reading: ZM Chapter 8 and MMDS Chapter 6

Market Basket Analysis

Grocery Shopping

TEXAS A&M UNIVERSITY
Engineering

Market-Basket Model

- Large set of items
- Large set of baskets
- Each basket is a small subset of items
- Goal: Discover association rules

Input:

TID	*Items*
1	Bread, Coke, Milk
2	Beer, Bread
3	Beer, Coke, Diaper, Milk
4	Beer, Bread, Diaper, Milk
5	Coke, Diaper, Milk

Output:

Rules Discovered:

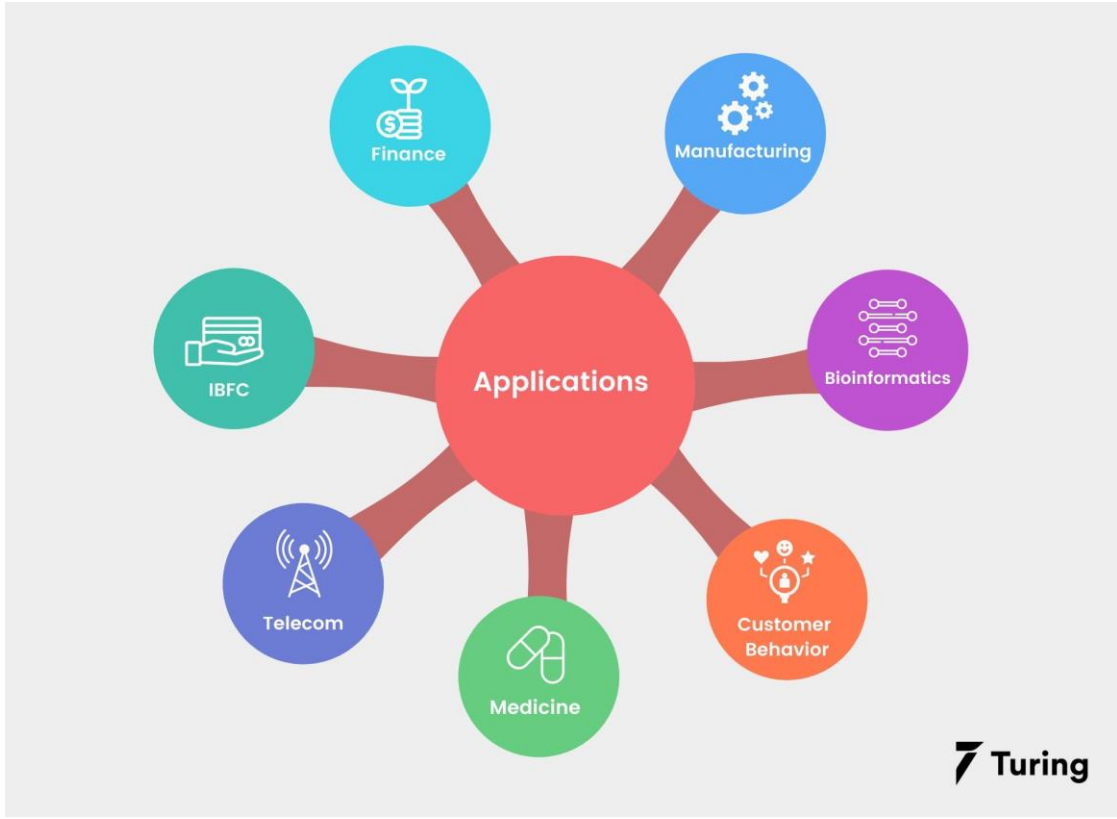
{Milk} --> {Coke}

{Diaper, Milk} --> {Beer}

Material adapted from: Mining of Massive Datasets.

Market-Basket Model Applications

TEXAS A&M UNIVERSITY
Engineering



Frequent Itemsets

Frequent Itemset Terminology

- Find sets of items that occur frequently
- Example:
  - A = {1, 2, 3, 4, 5}
  - B = {1, 2, 3}
  - C = {3, 4, 5}
- Frequent sets: {1, 2, 3}, {3, 4, 5}, etc.

Frequent Itemset Terminology

TEXAS A&M UNIVERSITY Engineering

- Itemsets: set of elements (items) with cardinality *k*
- Tidsets: set of elements (transaction identifiers or tids)
- Transactions: tuple of the form *(t, X)* with unique tids (*t*)
- Database: binary database **D** with binary relation on the set of tids and items

$$\mathcal{I} = \{x_1, x_2, \ldots, x_m\}$$

$$\mathcal{T} = \{t_1, t_2, \ldots, t_n\}$$

$$\langle t, X \rangle$$

Database Representation

TEXAS A&M UNIVERSITY Engineering

- Dataset **D** has 5 items and 6 tids
  - $\mathcal{I} =$ {A, B, C, D, E}
  - $\mathcal{T} =$ {1, 2, 3, 4, 5, 6}

D	A	B	C	D	E
1	1	1	0	1	1
2	0	1	1	0	1
3	1	1	0	1	1
4	1	1	1	0	1
5	1	1	1	1	1
6	0	1	1	1	0

Binary Database

t	$\boldsymbol{i}(t)$
1	ABDE
2	BCE
3	ABDE
4	ABCE
5	ABCDE
6	BCD

Transaction Database

$\boldsymbol{t}(x)$				
A	B	C	D	E
1	1	2	1	1
3	2	4	3	2
4	3	5	5	3
5	4	6	6	4
	5			5
	6			

Vertical Database

One-Hot Encoding

TEXAS A&M UNIVERSITY
Engineering

- Categorical attributes need to be converted to numeric values
- String category attributes such as
  Attribute = “Color” = {“red”, “green”, “blue”} conversion to int (use Pandas factorize()) maps to integers 0-2
  - May enforce relationships that are not present
- Use 1-hot encoding (available in Sklearn)

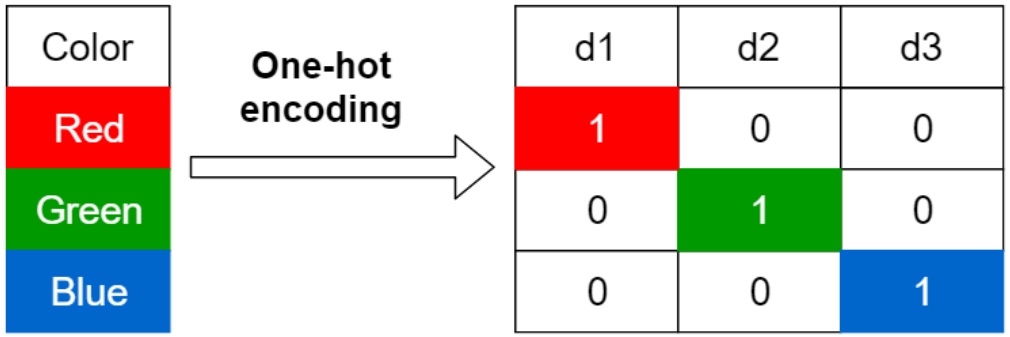


Image from: R. Pramoditha. Encoding Categorical Variables: One-hot vs Dummy Encoding

Measures of Frequent Itemsets and Association Rules

Support

- Support: number of transactions in **D** that contain itemset X
  - Number of baskets containing all items in X
- Support threshold: minimum support number needed to define frequent itemsets
- $\mathcal{F}$ denotes all frequent items and $\mathcal{F}^{(k)}$ denotes the set of frequent k-items

TID	*Items*
1	Bread, Coke, Milk
2	Beer, Bread
3	Beer, Coke, Diaper, Milk
4	Beer, Bread, Diaper, Milk
5	Coke, Diaper, Milk

Support of
{Beer, Bread} = 2

Support Example (Min support = 3)

TEXAS A&M UNIVERSITY Engineering

t	$\boldsymbol{i}(t)$
1	$ABDE$
2	BCE
3	$ABDE$
4	$ABCE$
5	$ABCDE$
6	BCD

Transaction Database

sup	itemsets
6	B
5	E, BE
4	$A, C, D, AB, AE, BC, BD, ABE$
3	$AD, CE, DE, ABD, ADE, BCE, BDE, ABDE$

Frequent Itemsets

$$\mathcal{F}^{(1)} = \{A, B, C, D, E\}$$
$$\mathcal{F}^{(2)} = \{AB, AD, AE, BC, BD, BE, CE, DE\}$$
$$\mathcal{F}^{(3)} = \{ABD, ABE, ADE, BCE, BDE\}$$
$$\mathcal{F}^{(4)} = \{ABDE\}$$

Confidence

- Association rules
  - If-then rules about the contents of “baskets”
  - $\{x_1, x_2, \ldots, x_m\} \longrightarrow j$
    - if a “basket” contains all of $x_1, x_2, \ldots, x_m$ then it is likely to contain j
- Many rules in practice
  - Need to choose interesting/significant rules
- Confidence:

$$\text{conf}(\mathcal{I} \rightarrow j) = \frac{\text{support}(\mathcal{I} \cup j)}{\text{support}(\mathcal{I})} = P(\mathcal{I} \mid j) = \frac{P(j \wedge \mathcal{I})}{P(j)} = \frac{sup(j\,\mathcal{I})}{sup(j)}$$

Other Measures

- Confidence alone may not be sufficient
- Interest
  - Confidence – Probability of *j*
- Lift
- Conviction
- All-confidence
- Etc.

B_1 = {m, c, b} B_2 = {m, p, j}

B_3 = {m, b} B_4= {c, j}

B_5 = {m, p, b} B_6 = {m, c, b, j}

B_7 = {c, b, j} B_8 = {b, c}

$$\text{conf}(\mathcal{I} \rightarrow j) = \frac{\text{support}(\mathcal{I} \cup j)}{\text{support}(\mathcal{I})}$$

Association rule: {m, b} →c

- **Confidence** = 2/4 = 0.5
- **Interest** = |0.5 – 5/8| = 1/8
  - Item ***c*** appears in 5/8 of the baskets
  - Rule is not very interesting!

Mining Association Rules

Mining Association Rules Overview

TEXAS A&M UNIVERSITY Engineering

- Problem: Find all association rules with support $\geq s$ and confidence $\geq c$
- Need to find the frequent items
- Step 1: Find all frequent itemsets
  - Leverage minimum support
- Step 2: Rule generation
  - Output rules above confidence threshold

Mining Association Rules Overview

TEXAS A&M UNIVERSITY Engineering

- Problem: Find all association rules with support $\geq s$ and confidence $\geq c$
- Need to find the frequent items
- **Step 1: Find all frequent itemsets**
  - Leverage minimum support
- Step 2: Rule generation
  - Output rules above confidence threshold

Frequent Itemset Algorithms

- Brute force/Naïve Approach
- Apriori
- Frequent Pattern Growth

Frequent Itemset Algorithms

TEXAS A&M UNIVERSITY
Engineering

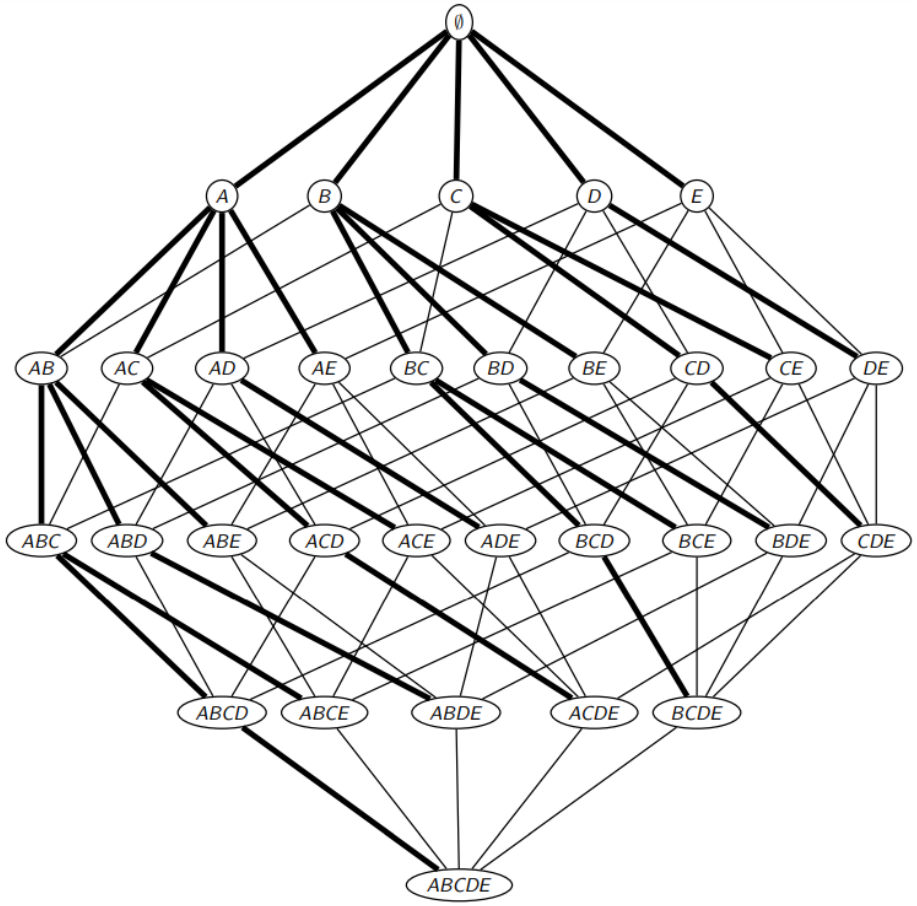
- **Brute force/Naïve Approach**
- Apriori
- Frequent Pattern Growth

Brute Force/Naïve Approach

TEXAS A&M UNIVERSITY Engineering

- Find all possible itemsets and compute support in **D**
- Two steps:
  - Candidate generation
  - Support computation
- Computationally expensive

$$O(|\mathcal{I}| \cdot |\boldsymbol{D}| \cdot 2^{|\mathcal{I}|})$$



Apriori Algorithm

Apriori Algorithm Overview

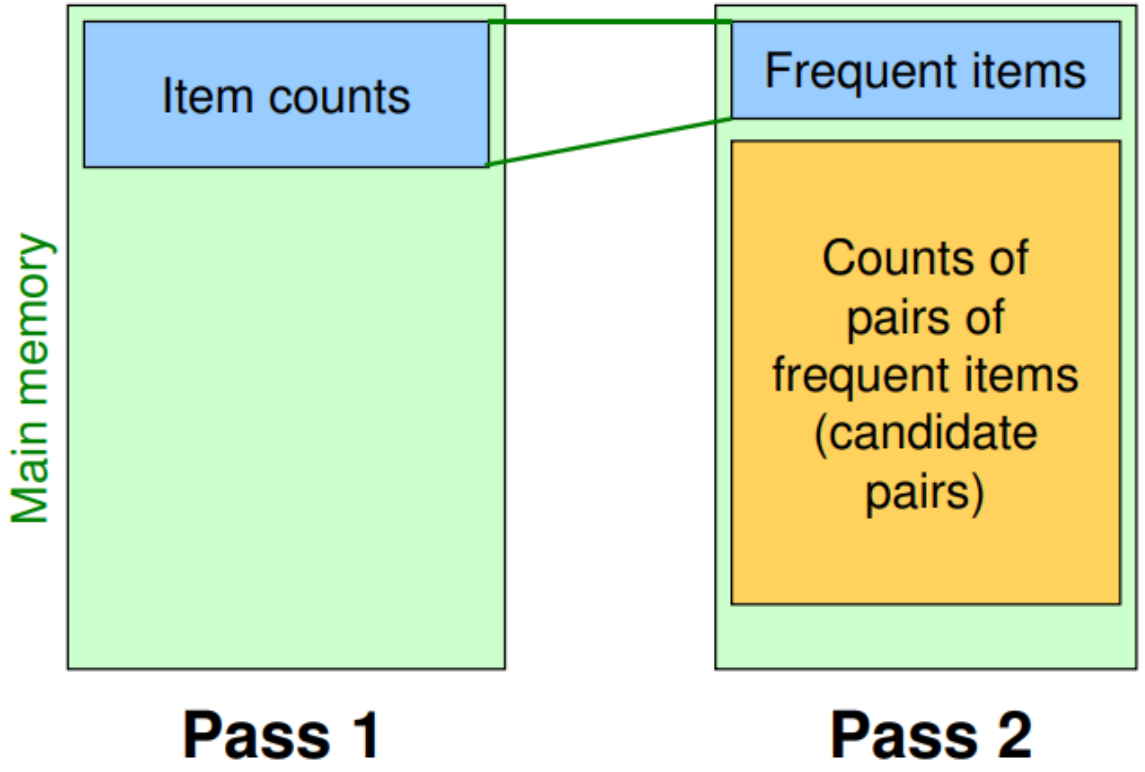
TEXAS A&M UNIVERSITY Engineering

- "Two-pass" method improves over brute force/naïve approach
- Monotonicity
  - If a set of items appears at least ***s*** times, so does every subset ***J*** of ***I***
- Contrapositive for pairs
  - If item ***i*** does not appear in ***s*** baskets, then no pair including ***i*** can appear in **s** baskets

Apriori Algorithm Overview

TEXAS A&M UNIVERSITY Engineering

- Pass 1: Read baskets and count occurrences of each individual item
- Pass 2: Read baskets again and count only pairs where both elements are frequent



Apriori Algorithm Overview

TEXAS A&M UNIVERSITY Engineering

- Can generalize beyond pairs
- For each *k*, construct two sets of *k* tuples ($\mathcal{F}^{(k)}$)
- Candidates, $\mathbf{C}_k$, possible frequent itemsets
- Filter for true frequent *k*-tuples, $\mathbf{L}_k$

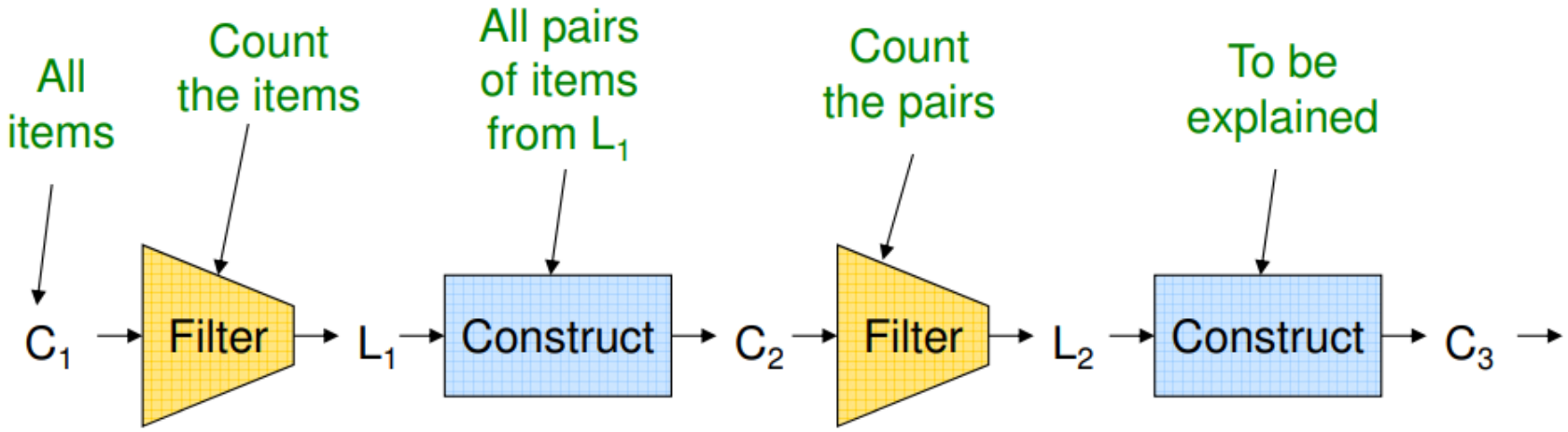


Image from: Mining of Massive Datasets.

Apriori Algorithm Example

Apriori Algorithm Example

TEXAS A&M UNIVERSITY
Engineering

- Three main steps:
  - Count, Pruning, Joining

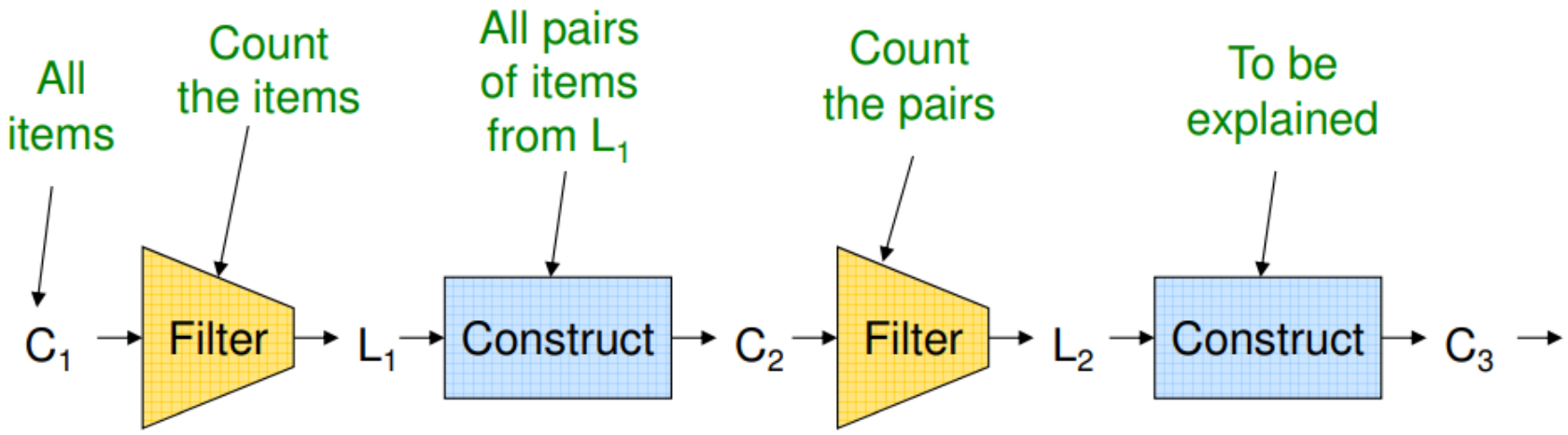


Image from: Mining of Massive Datasets.

Apriori Algorithm Example

TEXAS A&M UNIVERSITY
Engineering

- **D** =

TID	i(t)
1	A, C, D
2	B, C, E
3	A, B, C, E
4	B, E

- Find frequent itemsets, $\mathcal{F}^{(2)}$ and $\mathcal{F}^{(3)}$, with minimum support = 2

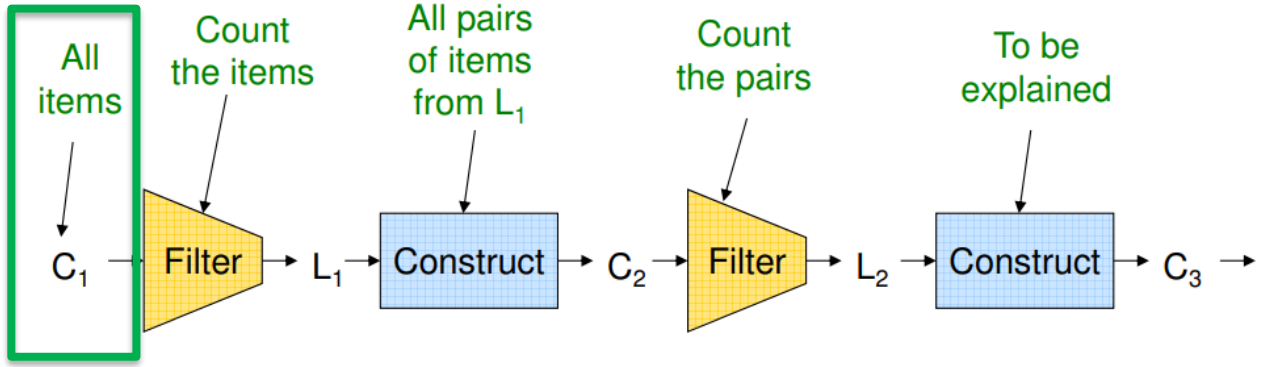
Apriori Algorithm Example

TEXAS A&M UNIVERSITY Engineering

- **D =**

TID	i(t)
1	A, C, D
2	B, C, E
3	A, B, C, E
4	B, E

- Step 1: Count items



Apriori Algorithm Example: Count

TEXAS A&M UNIVERSITY Engineering

- **D** =

TID	i(t)
1	A, C, D
2	B, C, E
3	A, B, C, E
4	B, E

- C_1 =

Itemsets	Count
{A}	2
{B}	3
{C}	3
{D}	1
{E}	3

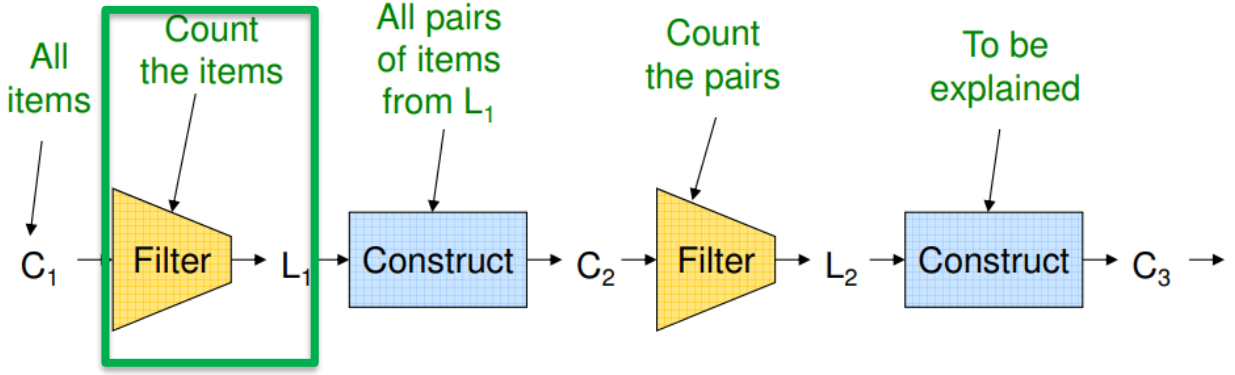
Apriori Algorithm Example

TEXAS A&M UNIVERSITY Engineering

- **D =**

TID	i(t)
1	A, C, D
2	B, C, E
3	A, B, C, E
4	B, E

- Step 2: Filter items



Apriori Algorithm Example: Filter/Pruning

- C_1 =

Itemsets	Count
{A}	2
{B}	3
{C}	3
{D}	1
{E}	3

- L_1 =

Itemsets	Count
{A}	2
{B}	3
{C}	3
{E}	3

- Remove all itemsets that do not meet minimum support (i.e., 2)

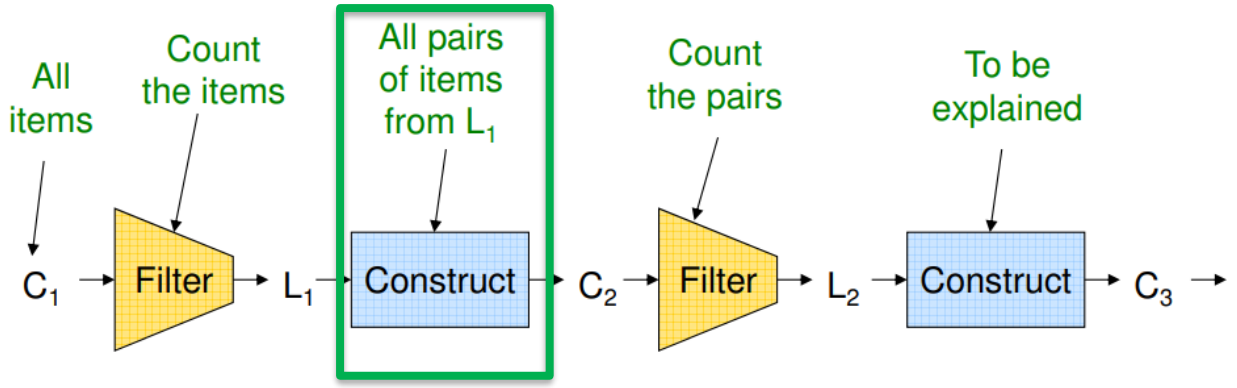
Apriori Algorithm Example

TEXAS A&M UNIVERSITY Engineering

- **D =**

TID	i(t)
1	A, C, D
2	B, C, E
3	A, B, C, E
4	B, E

- Step 3: Join items



Apriori Algorithm Example: Join Items

- L_1 =

Itemsets	Count
{A}	2
{B}	3
{C}	3
{E}	3

- C_2 =

Itemsets	Count
{A, B}	?
{A, C}	?
{A, E}	?
{B, C}	?
{B, E}	?
{C, E}	?

- Use **D** to generate new counts

Apriori Algorithm Example, Iteration 2

TEXAS A&M UNIVERSITY Engineering

- Break into pairs
- Work through second iteration to generate C_2 and L_2
- 5 minutes for activity

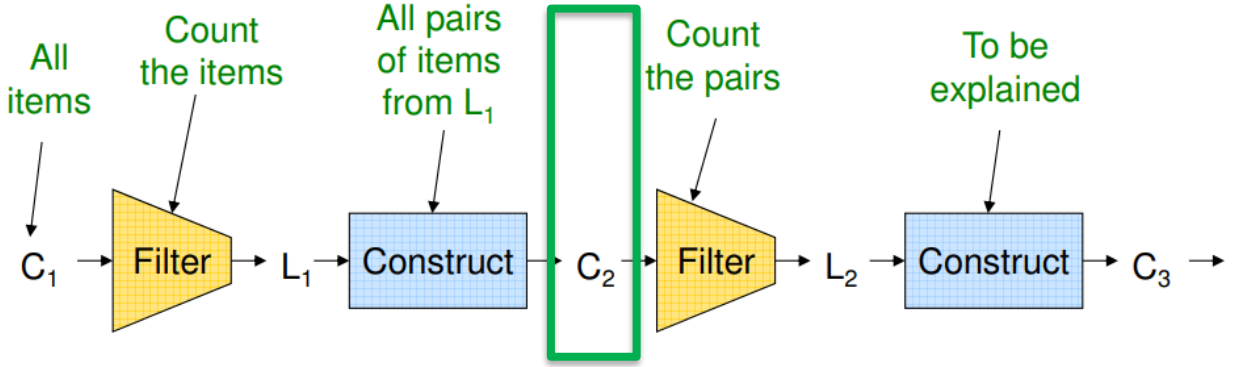
Apriori Algorithm Example, Iteration 2

TEXAS A&M UNIVERSITY Engineering

- **D =**

TID	i(t)
1	A, C, D
2	B, C, E
3	A, B, C, E
4	B, E

- Step 1: Count items



Apriori Algorithm Example: Count

TEXAS A&M UNIVERSITY Engineering

- **D** =

TID	i(t)
1	A, C, D
2	B, C, E
3	A, B, C, E
4	B, E

- C_2 =

Itemsets	Count
{A, B}	1
{A, C}	2
{A, E}	1
{B, C}	2
{B, E}	3
{C, E}	2

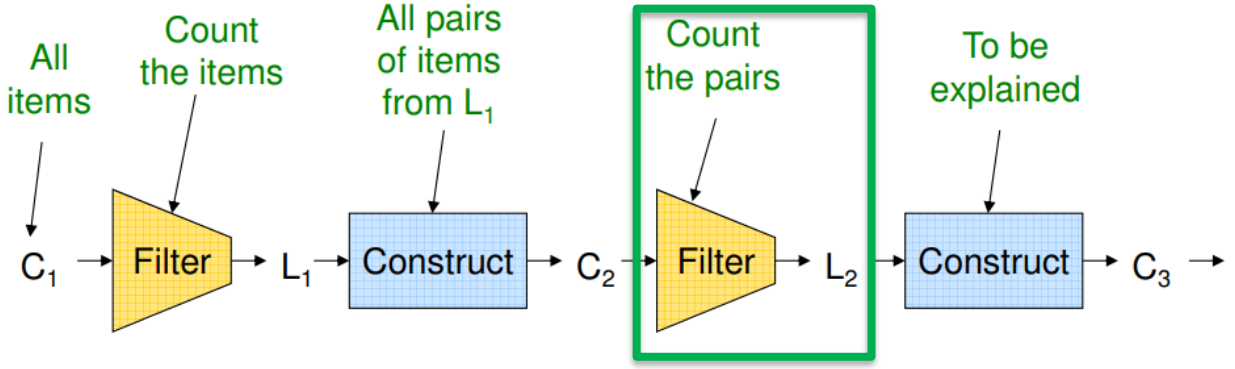
Apriori Algorithm Example, Iteration 2

TEXAS A&M UNIVERSITY Engineering

- **D =**

TID	i(t)
1	A, C, D
2	B, C, E
3	A, B, C, E
4	B, E

- Step 2: Filter items



Apriori Algorithm Example: Filter/Pruning

TEXAS A&M UNIVERSITY Engineering

- C_2 =

Itemsets	Count
{A, B}	1
{A, C}	2
{A, E}	1
{B, C}	2
{B, E}	3
{C, E}	2

- L_2 =

Itemsets	Count
{A, C}	2
{B, C}	2
{B, E}	3
{C, E}	2

- Remove all itemsets that do not meet minimum support (i.e., 2)

Apriori Algorithm Example: Filter/Pruning

TEXAS A&M UNIVERSITY Engineering

- L_2 =

Itemsets	Count
{A, C}	2
{B, C}	2
{B, E}	3
{C, E}	2

- This would be your answer for $\mathcal{F}^{(2)}$
- Continue process for $\mathcal{F}^{(3)}$

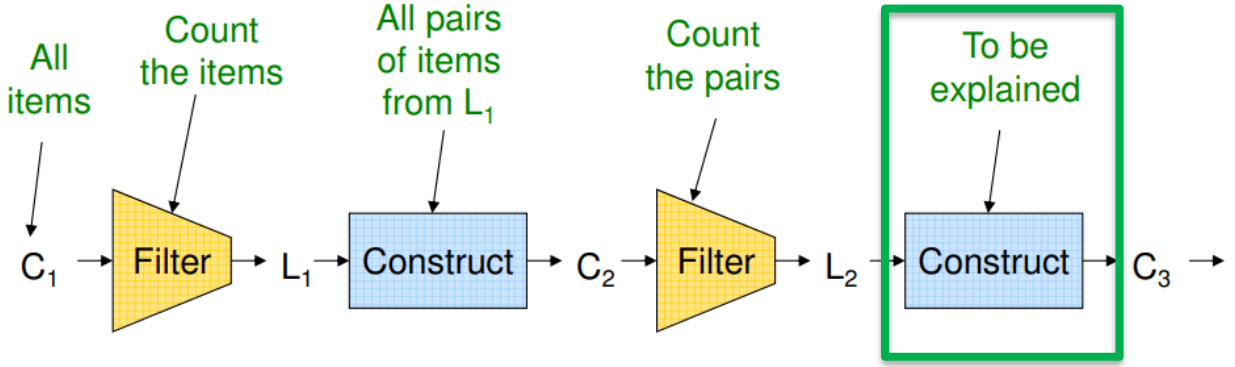
Apriori Algorithm Example, Iteration 2

TEXAS A&M UNIVERSITY Engineering

- **D =**

TID	i(t)
1	A, C, D
2	B, C, E
3	A, B, C, E
4	B, E

- Step 3: Join items



Apriori Algorithm Example: Join Items

- L_2 =

Itemsets	Count
{A, C}	2
{B, C}	2
{B, E}	3
{C, E}	2

- C_3 =

Itemsets	Count
{A, B, C}	?
{B, C, E}	?
{A,C, E}	?
{A, B, E}	?

- Use **D** to generate new counts

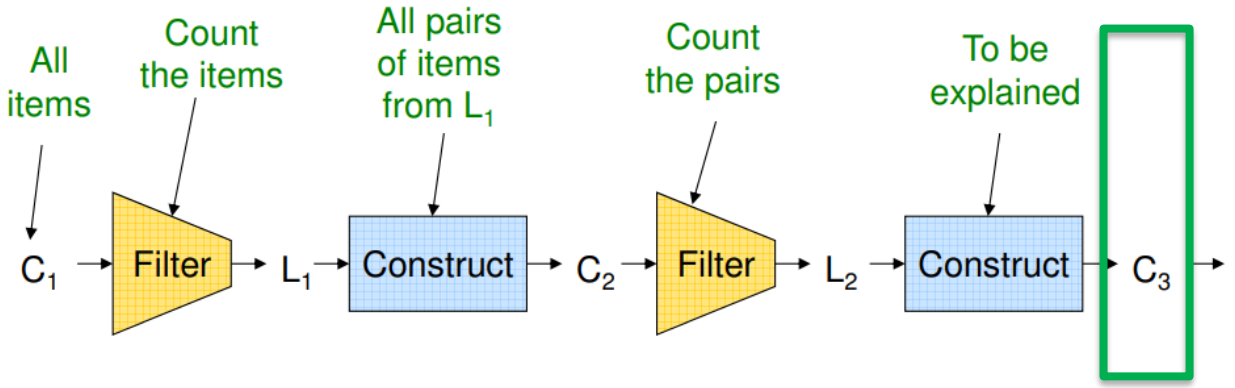
Apriori Algorithm Example, Iteration 3

TEXAS A&M UNIVERSITY Engineering

- **D =**

TID	i(t)
1	A, C, D
2	B, C, E
3	A, B, C, E
4	B, E

- Step 1: Count items



Apriori Algorithm Example: Count

TEXAS A&M UNIVERSITY Engineering

- D =

TID	i(t)
1	A, C, D
2	B, C, E
3	A, B, C, E
4	B, E

- C_3 =

Itemsets	Count
{A, B, C}	1
{B, C, E}	2
{A,C, E}	1
{A, B, E}	1

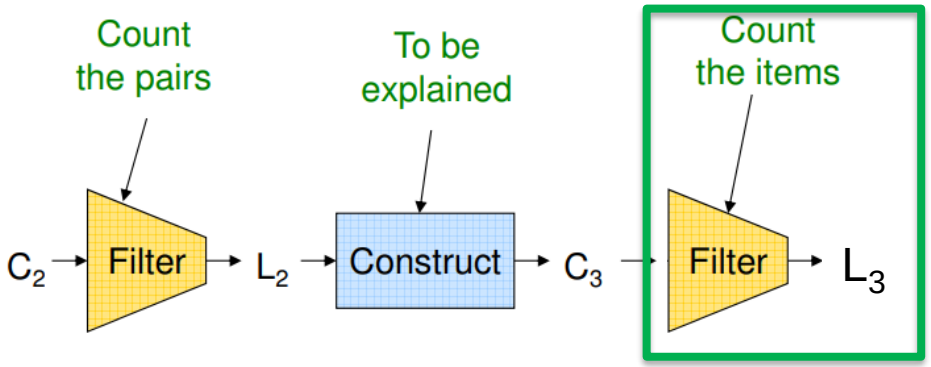
Apriori Algorithm Example, Iteration 3

TEXAS A&M UNIVERSITY Engineering

- **D =**

TID	i(t)
1	A, C, D
2	B, C, E
3	A, B, C, E
4	B, E

- Step 2: Filter items



Apriori Algorithm Example: Filter/Pruning

- C_3 =

Itemsets	Count
{A, B, C}	1
{B, C, E}	2
{A,C, E}	1
{A, B, E}	1

- L_3 =

Itemsets	Count
{B, C, E}	2

- This would be your answer for $\mathcal{F}^{(3)}$

Frequent Pattern Growth

Frequent Pattern Growth Overview

TEXAS A&M UNIVERSITY Engineering

- Frequent Pattern Growth (FPGrowth) indexes database for fast support computation
  - Uses frequent pattern tree
- Node in tree represents single item and child node represents different item

TID	Items bought
100	{a, c, d, f, g, i, m, p}
200	{a, b, c, f, i, m, o}
300	{b, f, h, j, o}
400	{b, c, k, s, p}
500	{a, c, e, f, l, m, n, p}

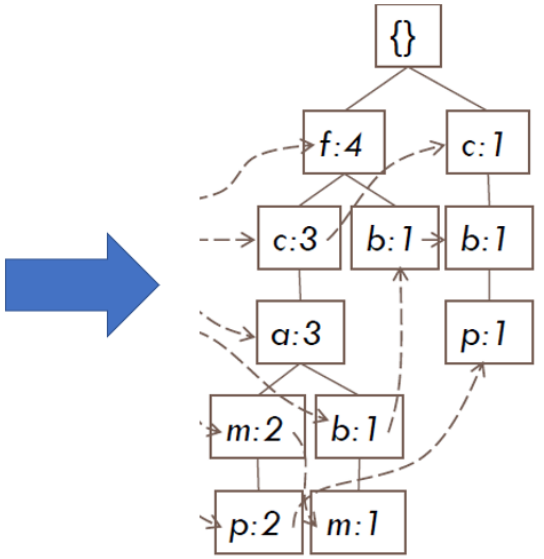


Image from: Chonny, FP Growth — Frequent Pattern Generation in Data Mining with Python Implementation.

FP-tree Generation

TEXAS A&M UNIVERSITY
Engineering

- FP-tree is a prefixed compressed representation of **D**
- Sorted in descending order of support

Transactions
BEAD
BEC
BEAD
BEAC
BEACD
BCD

FPGrowth Example

FP-tree Generation: Frequency Table

TEXAS A&M UNIVERSITY Engineering

- Count items
- Filter based off minimum support
  - Let's use min support = 2

Transactions
BEAD
BEC
BEAD
BEAC
BEACD
BCD

Items	Count
A	4
B	6
C	4
D	4
E	5

FP-tree Generation: Order Itemsets

TEXAS A&M UNIVERSITY Engineering

- Order based off frequency
- Sort transactions based on highest frequency
  - Completed already in example

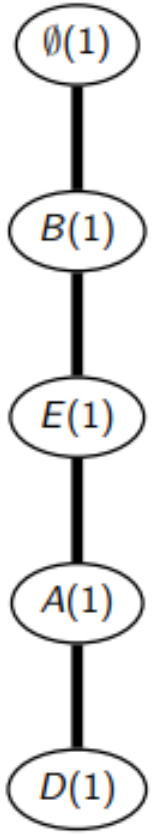
Items	Count
B	6
E	5
A	4
C	4
D	4

Transactions
BEAD
BEC
BEAD
BEAC
BEACD
BCD

FP-tree Generation: Tree Structure

- Take each transaction and insert into tree
- Begin with null node

Transactions
BEAD
BEC
BEAD
BEAC
BEACD
BCD

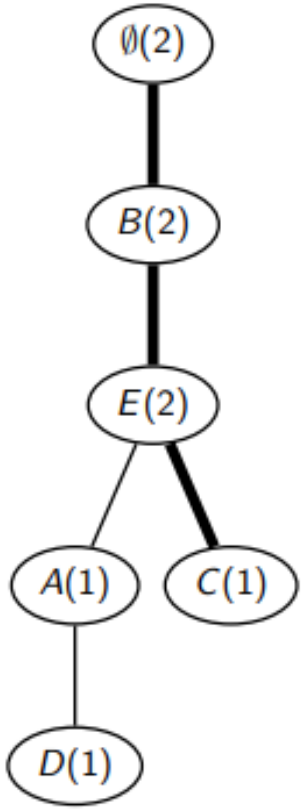


FP-tree Generation: Tree Structure

TEXAS A&M UNIVERSITY Engineering

- Take each transaction and insert into tree
- Begin with null node

Transactions
BEAD
BEC
BEAD
BEAC
BEACD
BCD

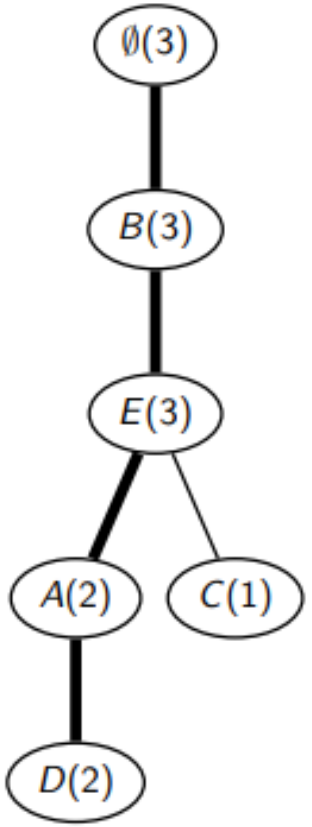


FP-tree Generation: Tree Structure

TEXAS A&M UNIVERSITY Engineering

- Take each transaction and insert into tree
- Begin with null node

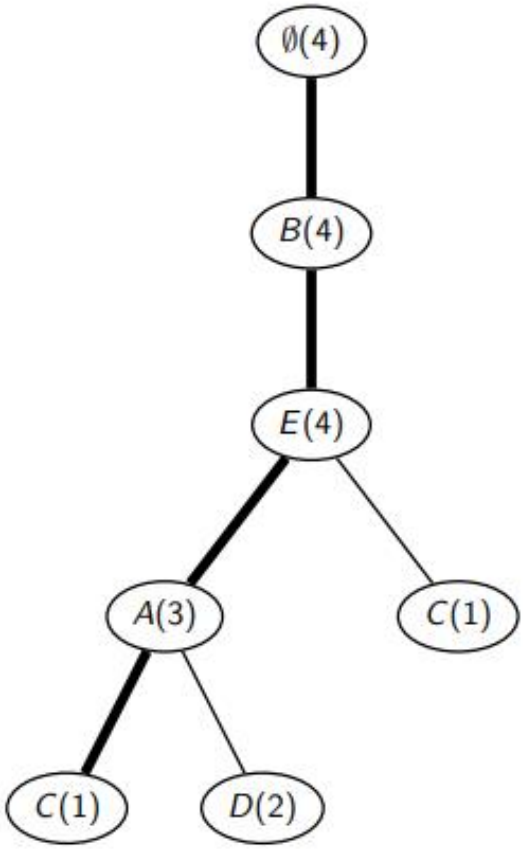
Transactions
BEAD
BEC
BEAD
BEAC
BEACD
BCD



FP-tree Generation: Tree Structure

- Take each transaction and insert into tree
- Begin with null node

Transactions
BEAD
BEC
BEAD
BEAC
BEACD
BCD

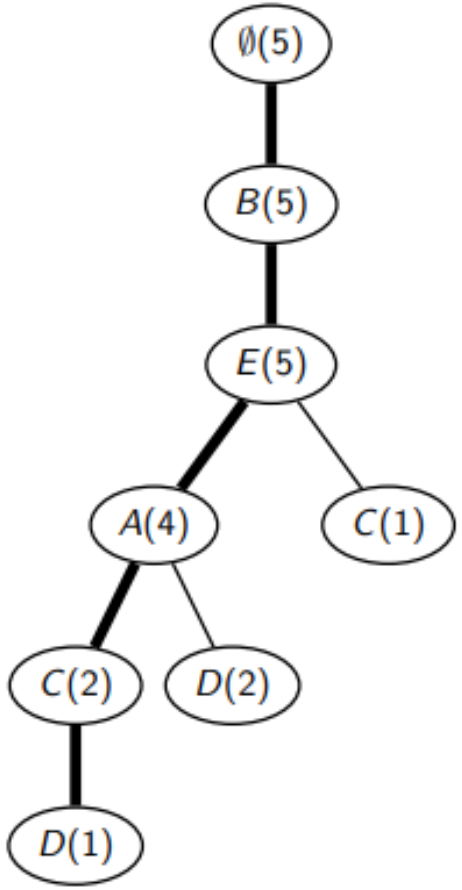


FP-tree Generation: Tree Structure

TEXAS A&M UNIVERSITY Engineering

- Take each transaction and insert into tree
- Begin with null node

Transactions
BEAD
BEC
BEAD
BEAC
BEACD
BCD

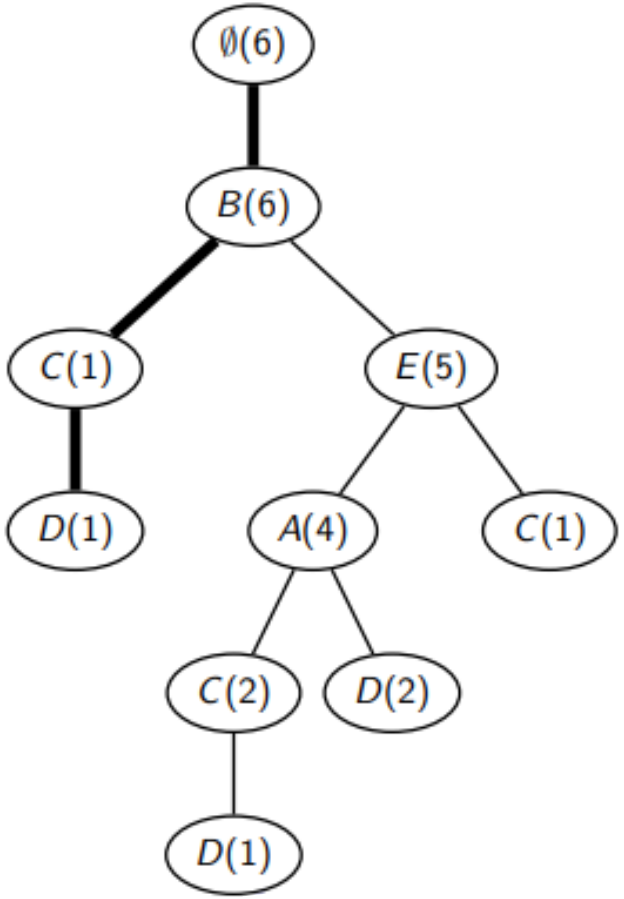


FP-tree Generation: Tree Structure

TEXAS A&M UNIVERSITY Engineering

- Take each transaction and insert into tree
- Begin with null node

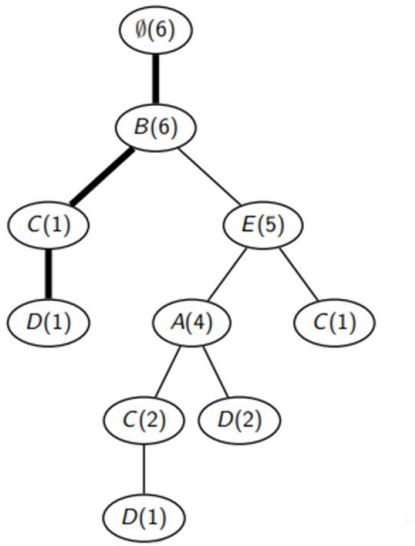
Transactions
BEAD
BEC
BEAD
BEAC
BEACD
BCD



FP-tree Generation

- Once FP-tree is completed, generate conditional pattern base
- Sorted in ascending order of support
- Make sure conditional patterns add up to number of support for individual item

Transactions
BEAD
BEC
BEAD
BEAC
BEACD
BCD



Items	Conditional Pattern Base
D	{BC: 1, BEA: 2, BEAC: 1}
C	{B: 1, BE: 1, BEA: 2}
A	{BE: 4}
E	{B: 5}
B	-

FP Growth Algorithm

TEXAS A&M UNIVERSITY
Engineering

- Create conditional frequent pattern tree
- Generate rules
- FP Algorithm Example

Apriori vs FPGrowth

Apriori vs FPGrowth

TEXAS A&M UNIVERSITY
Engineering

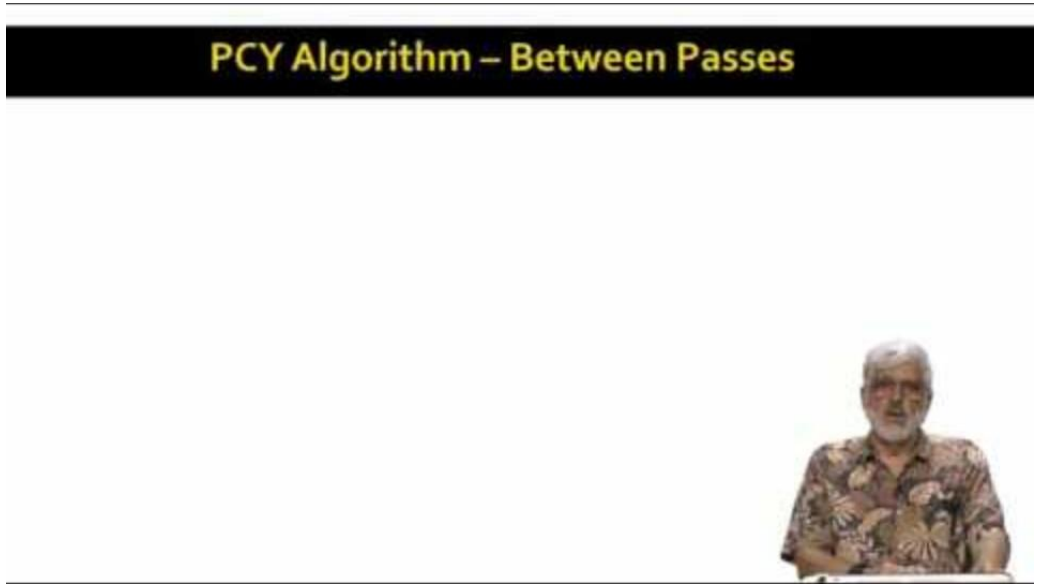
	FP Growth	Apriori
Speed	Faster, runtime increases linearly with increase in number of itemsets	Slower, runtime increases exponentially with increase in number of itemsets
Memory	Small, storing the compact version of database	Large, all the candidates from self-joining are stored in the memory
Candidates	No candidate generation	Use self-joining for candidate generation
Frequent patterns	Pattern growth achieved by mining conditional FP trees.	Patterns selected from the candidates whose support is higher than minSup.
Scans	Only require two scans	Scan the database over and over again.

Image from: Chonny, FP Growth — Frequent Pattern Generation in Data Mining with Python Implementation.

Other Frequent Mining Algorithms

TEXAS A&M UNIVERSITY Engineering

- Park-Chen-Yu (PCY) [MMDS]
- Random sampling [MMDS]
- Savasere, Omiecinski, and Navathe (SON) [MMDS]
- Toivonen [MMDS]
- Equivalent Class (Eclat) [ZM]



Mining Association Rules Overview

TEXAS A&M UNIVERSITY Engineering

- Problem: Find all association rules with support $\geq s$ and confidence $\geq c$
- Need to find the frequent items
- Step 1: Find all frequent itemsets
  - Leverage minimum support
- **Step 2: Rule generation**
  - Output rules above confidence threshold

Association Rule Generation

TEXAS A&M UNIVERSITY Engineering

B_1 = {m, c, b} B_2 = {m, p, j}

B_3 = {m, c, b, n} B_4= {c, j}

B_5 = {m, p, b} B_6 = {m, c, b, j}

B_7 = {c, b, j} B_8 = {b, c}

$$\text{conf}(\mathcal{I} \rightarrow j) = \frac{\text{support}(\mathcal{I} \cup j)}{\text{support}(\mathcal{I})}$$

- **Support threshold $s = 3$, confidence $c = 0.75$**
- **1) Frequent itemsets:**
  - {b,m} {b,c} {c,m} {c,j} {m,c,b}
- **2) Generate rules:**
  - ~~b→m: c=4/6~~ b→c: c=5/6 ~~b,c→m: c=3/5~~
  - m→b: c=4/5 … b,m→c: c=3/4
  - ~~b→c,m: c=3/6~~

Python Package for Frequent Itemset Mining

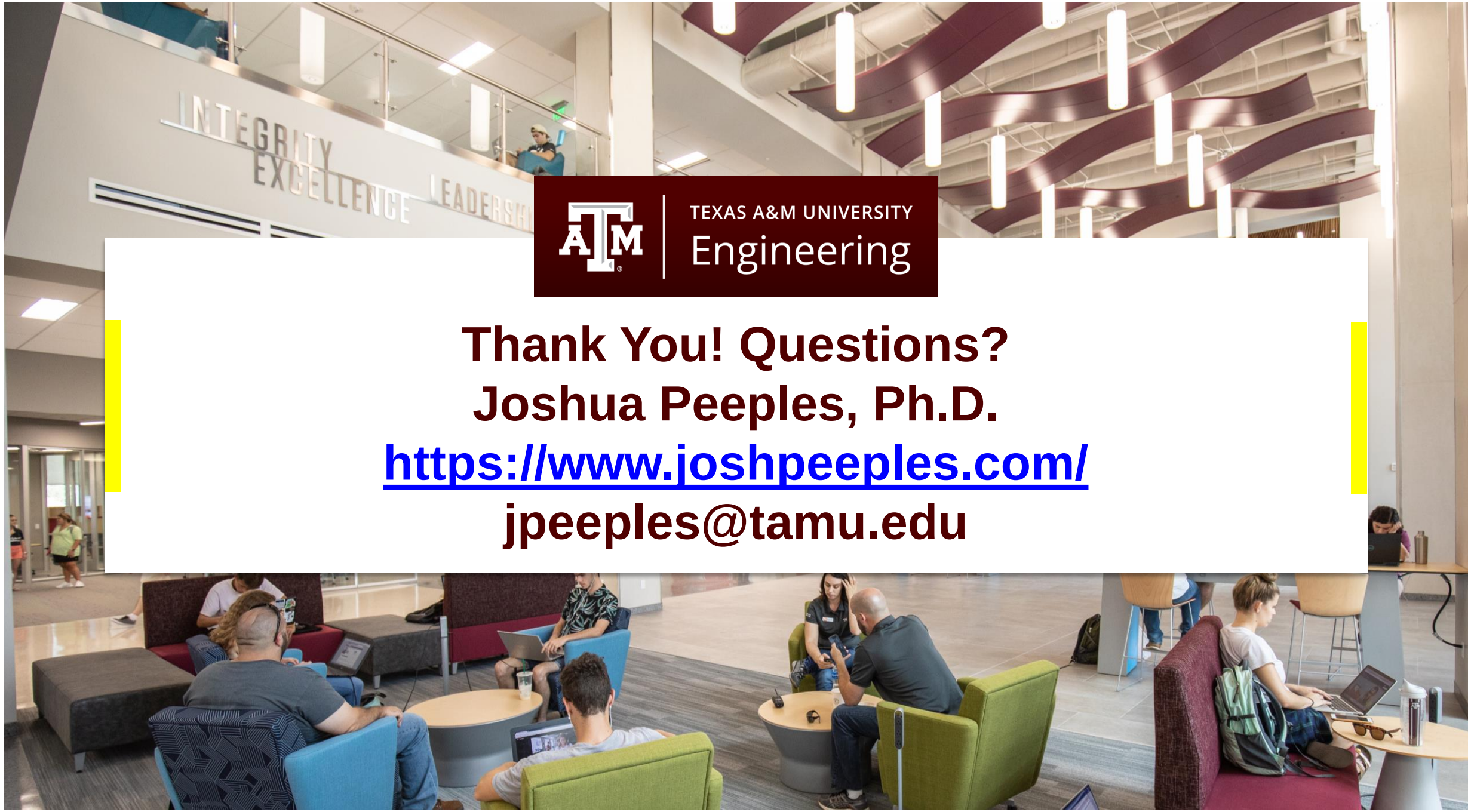
TEXAS A&M UNIVERSITY Engineering

- Machine Learning Extensions (Mlxtend)
- Apriori and FP-Growth implementations available!

Next class

TEXAS A&M UNIVERSITY
Engineering

- Representative Clustering



TEXAS A&M UNIVERSITY
Engineering

Thank You! Questions?

Joshua Peeples, Ph.D.

https://www.joshpeeples.com/

jpeeples@tamu.edu

TEXAS A&M UNIVERSITY
Engineering

Supplemental Slides

Useful Links

- Set notation
- Frequent Itemset Mining- Apriori Algorithm
- Apriori — Association Rule Mining In-depth Explanation and Python Implementation
- FP Growth- Frequent Pattern Generation in Data Mining with Python Implementation