Unit-5
Mining Frequent Patterns

Frequent Patterns

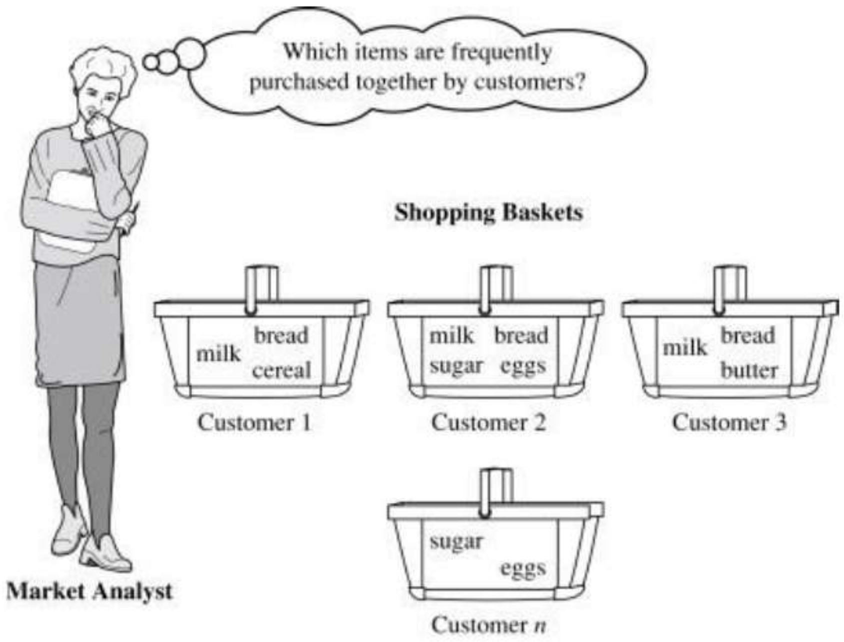
Frequent patterns are patterns (***itemsets, subsequences, or substructures)*** that appear in a data set frequently.

- For example, a set of items, such as milk and bread that appear frequently together in a transaction data set is a ***frequent itemset***.
- A ***subsequence***, such as buying first a PC, then a digital camera, and then a memory card, if it occurs frequently in a shopping history database, is a (frequent) sequential pattern.
- A ***substructure*** can refer to different structural forms, such as *subgraphs* or *subtrees*, which may be combined with itemsets or subsequences. If a substructure occurs frequently in a graph database, it is called a (frequent) structural pattern.

Finding frequent patterns plays an essential role in mining associations, correlations, and many other interesting relationships among data. Moreover, it helps in data indexing, classification, clustering, and other data mining tasks as well.

Market Basket Analysis

Market basket analysis analyzes customer buying habits by finding associations between the different items that customer place in their shopping baskets. The discovery of such associations can help retailers develop marketing strategies by gaining insight into which items are frequently purchased together by customers. Such information can lead to increased sales by helping retailers do selective marketing and design different store layouts.



Items that are frequently purchased together can be placed in proximity in order to further encourage the sale of such items together. Market basket analysis can also help retailers plan which items to put on sale at reduced prices. If customers tend to purchase *milk* and *bread* together, then having a sale on *milk* may encourage the sale of *milk* as well as *bread*.

Frequent Itemsets

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

- ***Itemsets:*** An itemset is a set of items.
  E.g. *X = {milk, bread, coke}* is an itemset.
- An itemset that contains k items is a ***k-itemset***.
  E.g. The set *{milk, bread}* is a 2-itemset.
- ***Support Count (σ):*** Frequency of occurrence of an itemset.
  E.g. $\sigma(bread, milk) = 3$
- ***Frequent itemset:*** Items that frequently appear together. E.g. *{bread, milk}*. An itemset whose support is greater than or equal to a minimum support threshold (*min_sup*) is called frequent itemset.

Closed Itemsets

An itemset X is closed in a data set S if there exists no proper super-itemset Y such that Y has the same support count as X in S. For example,

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

Here, {B} is closed itemset because it all supersets have less support count than the itemset {B}. But other itemsets such as {A}, {C}, and {D} is not closed itemsets. Here, item set {A, B} is also closed, because its superset {A, B, D} and {A, B, C, D} has less support count than {A, B}.

Association Rules

Association rules are if/then statements that help uncover relationships between seemingly unrelated data in a relational database or other information repository. It has two parts, an antecedent (if) and a consequent (then). An antecedent is an item found in the data. A consequent is an item that is found in combination with the antecedent.

Let $I = \{I_1, I_2, \dots \dots, I_n\}$ be a set of *items* and $D = \{T_1, T_2, \dots \dots, T_m\}$ be a set of *transactions* called the *database*. Each transaction in D has a unique transaction ID and contains a subset of the items in I. A *rule* is defined as an implication of the form:

$$\boldsymbol{X \rightarrow Y}$$

Where $X, Y \subseteq I$ and $X \cap Y = \emptyset$

An example rule for the supermarket could be $\{butter, bread\} \Rightarrow \{milk\}$ meaning that if a customer purchases butter and bread then he/she is also likely to buy milk.

Support and Confidence

To measure the interestingness of association rules two measures are used:

- ***Support:*** *Support* of association rule $X \Rightarrow Y$ is the percentage of transactions in *dataset* that contain both items (X & Y). In formula,

$$Support(X \Rightarrow Y) = P(X \cup Y) = \frac{\sigma(X \cup Y)}{No.of\ trans}$$

For example,

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

In above dataset, $Support(bread \Rightarrow milk) = \frac{3}{5}$ since both items occurs in *60%* of all transactions.

- ***Confidence:*** *Confidence* of association rule $X \Rightarrow Y$ is the percentage of transactions containing X that also contain Y. In formula,

$$Confidence(X \Rightarrow Y) = P(Y|X) = \frac{P(X \cup Y)}{P(X)} = \frac{\sigma(X \cup Y)}{\sigma(X)}$$

For example, In above dataset, $fidence(bread \Rightarrow milk) = \frac{3}{4}$, since *75%* of all transactions containing *bread* also contains *milk*.

Association Rule Mining

Association rule mining is a method for discovering frequent patterns in large databases. It is intended to identify strong rules discovered in databases using different measures of interestingness.

Association rule mining consists of two sub-processes: *finding frequent item sets* and *generating association rules* from those item sets.

- ***Frequent itemset:*** Frequent itemset is a set of items whose support is greater than the user specified minimum support.
- ***Association rule:*** An association rule must satisfy user-set minimum support (*min_sup*) and minimum confidence (*min_conf*). The rule $X \Rightarrow Y$ is called a strong association rule if *support* ≥ *min_sup* and *confidence* ≥ *min_conf*.

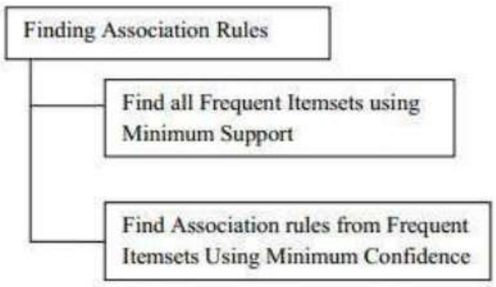

Finding Association Rules
Find all Frequent Itemsets using Minimum Support
Find Association rules from Frequent Itemsets Using Minimum Confidence

Figure 1: Generating Association Rules

Types of Association Rule

1. ***Single Dimensional Association Rule***

   It contains a single predicate (*e.g. purchase*) with its multiple occurrences (i.e., the predicate occurs more than once within the rule). For example:

   $purchase(X, \text{"milk"}) \Rightarrow purchase(X, \text{"bread"})$

2. ***Multi-dimensional Association Rule***

   It contains two or more predicates. Each predicate occurs only once. For example:

   $age(X, \text{"}19-25\text{"}) \wedge occupation(X, \text{"}student\text{"}) \Rightarrow buys(X, \text{"}laptop\text{"})$

3. ***Hybrid dimensional Association Rule***

   It is a multidimensional association rule with repeated predicates, which contain multiple occurrences of some predicates. For example:

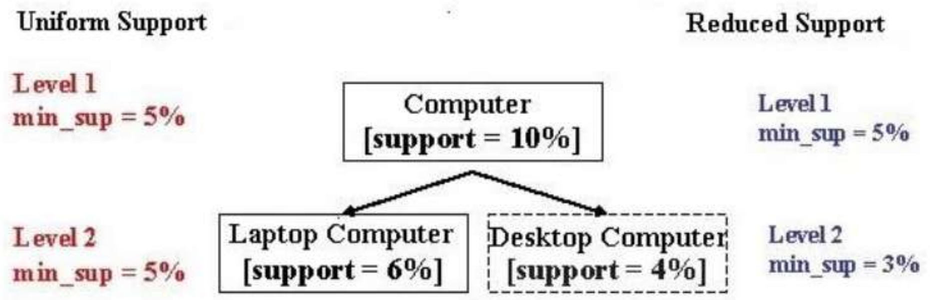
   $age(X, \text{"}19-25\text{"}) \wedge buys(X, \text{"}laptop\text{"}) \Rightarrow buys(X, \text{"}printer\text{"})$

4. ***Multilevel Association Rule***

   Association rules generated from mining data at multiple levels of abstraction are called multiple-level or multilevel association rules. Multilevel association rules can be mined efficiently using concept hierarchies under a support-confidence framework.

   Two Approaches of Multilevel Association Rules:

   - ***Using uniform minimum support for all levels (referred to as uniform support):*** The same minimum support threshold is used when mining at each level of abstraction.
   - ***Using reduced minimum support at lower levels (referred to as reduced support):*** Each level of abstraction has its own minimum support threshold. The deeper the level of abstraction, the smaller the corresponding threshold is.



5. ***Quantitative Association Rule***

   Database attributes can be categorical or quantitative. Categorical attributes have a finite number of possible values, with no ordering among the values (e.g., *occupation, brand, color*). Quantitative attributes are numeric and have an implicit ordering among values (e.g., *age, income, price*). Association rules mined from quantitative attributes are referred as quantitative association rules.

   Different from general association rules where both the left-hand and right-hand sides of the rule should be categorical attributes, at least one attribute of the quantitative association rule (left or right) must involve a numerical attribute. For example:

   $Age(x, \text{"}20-30) \hat{} salary(x, \text{"}50-60k\text{"}) \rightarrow buys(x, \text{"}laptop\text{"})$

Apriori Algorithm

Apriori algorithm is a sequence of steps to be followed to find the most frequent itemset in the given database.

- ***Apriori Property:*** All subsets of a frequent itemset must be frequent.
- ***Apriori pruning principle:*** If there is any itemset which is infrequent, its superset should not be generated/tested.

Apriori employs an iterative approach known as a level-wise search, where frequent k-itemsets are used to explore frequent (k+1)-itemsets. First, the set of frequent 1-itemsets that satisfy minimum support is found by scanning the database. The resulting set is denoted L_1. Next, L_1 is used to find L_2, the set of frequent 2-itemsets, which is used to find L_3, and so on, until no more frequent k-itemsets can be found. The finding of each L_k requires one full scan of the database. To improve the efficiency of the level-wise generation of frequent itemsets, an important property called the Apriori property is used.

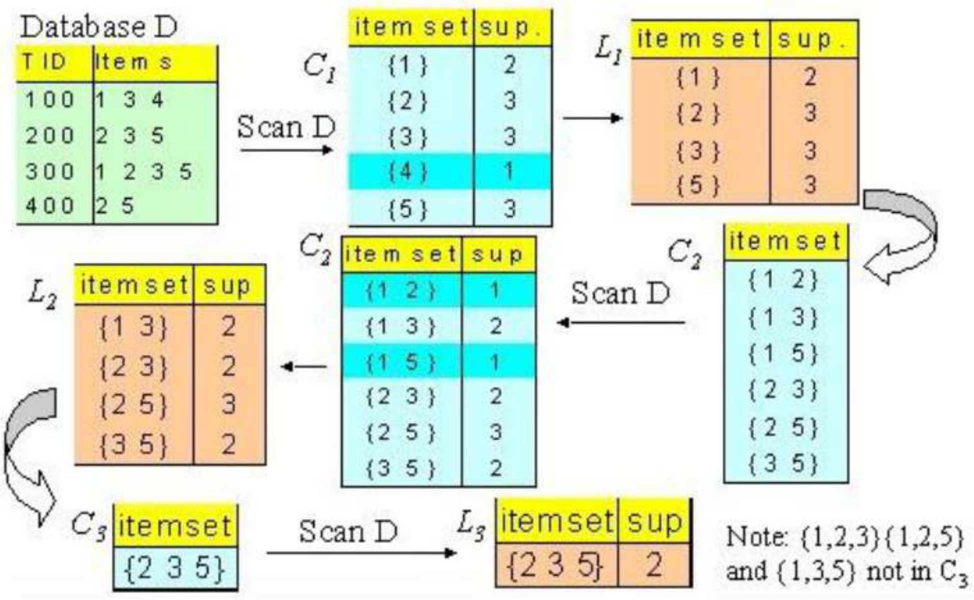
Pseudo-code:

```
$C_k$: 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 $\cup_k L_k$;
```

Example

Min_support = 2



Database D

TID	Items
100	1 3 4
200	2 3 5
300	1 2 3 5
400	2 5

C_1

itemset	sup.
{1}	2
{2}	3
{3}	3
{4}	1
{5}	3

L_1

itemset	sup.
{1}	2
{2}	3
{3}	3
{5}	3

C_2

itemset
{1 2}
{1 3}
{1 5}
{2 3}
{2 5}
{3 5}

C_2

itemset	sup
{1 2}	1
{1 3}	2
{1 5}	1
{2 3}	2
{2 5}	3
{3 5}	2

L_2

itemset	sup
{1 3}	2
{2 3}	2
{2 5}	3
{3 5}	2

C_3

itemset
{2 3 5}

L_3

itemset	sup
{2 3 5}	2

Generating Association Rules From Frequent Itemsets

```
For each frequent itemset, $f$, generate all non-empty subsets of $f$.
For every non-empty subset $s$ of $f$ do
    output rules $s \Rightarrow (f - s)$ if $\frac{support(f)}{support(s)} \geq \min\_confidence$
end
```

From above example:

Frequent itemset = {2, 3, 5}

Now,

Rules	Confidence
$I_2 \Rightarrow I_3 \wedge I_5$	2/3 = 66%
$I_3 \Rightarrow I_2 \wedge I_5$	2/3 = 66%
$I_5 \Rightarrow I_2 \wedge I_3$	2/3 = 66%
$I_2 \wedge I_3 \Rightarrow I_5$	2/2 = 100%
$I_2 \wedge I_5 \Rightarrow I_3$	2/3 = 66%
$I_3 \wedge I_5 \Rightarrow I_2$	2/2 = 100%

As the given threshold or minimum confidence is 75%, so the rules $I_2 \wedge I_3 \Rightarrow I_5$ and $I_3 \wedge I_5 \Rightarrow I_2$ can be considered as the strong association rules for the given problem.

Examples

Q. You are given the transaction data shown below from a fast food restaurant. There are 9 distinct transactions (order 1 to order 9). There are total 5 meal (M_1 to M_5) involved in transactions.

Meal Items	*List of IDs*	*Meal Items*	*List of IDs*
Order 1	M_1, M_2, M_5	***Order 6***	M_2, M_3
Order 2	M_2, M_4	***Order 7***	M_1, M_3
Order 3	M_2, M_3	***Order 8***	M_1, M_2, M_3, M_5
Order 4	M_1, M_2, M_4	***Order 9***	M_1, M_2, M_3
Order 5	M_1, M_3		

Minimum support = 2, Minimum confidence = 0.7

Apply the Apriori algorithm to the database to identify frequent k-itemset and find all strong association rules.

Solution:

Step 1: Calculating C_1 and L_1

Candidate 1-itemsets (C_1)

Itemset	Sup_count
$\{M_1\}$	6
$\{M_2\}$	7
$\{M_3\}$	6
$\{M_4\}$	2
$\{M_5\}$	2

Frequent 1-itemsets (L_1)

Items	Sup_count
$\{M_1\}$	6
$\{M_2\}$	7
$\{M_3\}$	6
$\{M_4\}$	2
$\{M_5\}$	2

Step 2: Calculating C_2 and L_2

Candidate 2-itemsets (C_2)

Itemset	Sup_count
$\{M_1, M_2\}$	4
$\{M_1, M_3\}$	4
$\{M_1, M_4\}$	1
$\{M_1, M_5\}$	2
$\{M_2, M_3\}$	4
$\{M_2, M_4\}$	2
$\{M_2, M_5\}$	2
$\{M_3, M_4\}$	0
$\{M_3, M_5\}$	1
$\{M_4, M_5\}$	0

Frequent 2-itemsets (L_2)

Items	Sup_count
$\{M_1, M_2\}$	4
$\{M_1, M_3\}$	4
$\{M_1, M_5\}$	2
$\{M_2, M_3\}$	4
$\{M_2, M_4\}$	2
$\{M_2, M_5\}$	2

Step 3: Calculating C_3 and L_3

Candidate 3-itemsets (C_3)

Itemset	Sup_count
$\{M_1, M_2, M_3\}$	2
$\{M_1, M_2, M_5\}$	2

Frequent 3-itemsets (L_3)

Items	Sup_count
$\{M_1, M_2, M_3\}$	2
$\{M_1, M_2, M_5\}$	2

Step 4: Calculating C_4 and L_4

A candidate set of 4-itemsets, C_4 is $\{M_1, M_2, M_3, M_5\}$. This item set is pruned since its subset $\{M_2, M_3, M_5\}$ is not frequent. Thus $C_4 = \phi$, and algorithm terminates .

∴ Frequent Itemsets = $\{M_1, M_2, M_3\}$ and $\{M_1, M_2, M_5\}$

Now,

Generating Association Rules from Frequent Itemsets:

Taking $\{M_1, M_2, M_3\}$

Rules	Confidence
$M_1 \Rightarrow M_2 \wedge M_3$	2/6 = 0.333
$M_2 \Rightarrow M_1 \wedge M_3$	2/7 = 0.285
$M_3 \Rightarrow M_1 \wedge M_2$	2/6 = 0.333
$M_2 \wedge M_3 \Rightarrow M_1$	2/4 = 0.5
$M_1 \wedge M_3 \Rightarrow M_2$	2/4 = 0.5
$M_1 \wedge M_2 \Rightarrow M_3$	2/4 = 0.5

Taking $\{M_1, M_2, M_5\}$

Rules	Confidence
$M_1 \Rightarrow M_2 \wedge M_5$	2/6 = 0.333
$M_2 \Rightarrow M_1 \wedge M_5$	2/7 = 0.285
$M_5 \Rightarrow M_1 \wedge M_2$	2/2 = 1
$M_2 \wedge M_5 \Rightarrow M_1$	2/2 = 1
$M_1 \wedge M_5 \Rightarrow M_2$	2/2 = 1
$M_1 \wedge M_2 \Rightarrow M_5$	2/4 = 0.5

As the given threshold or minimum confidence is 0.7, so the rules $M_5 \Rightarrow M_1 \wedge M_2$, $M_2 \wedge M_5 \Rightarrow M_1$ and $M_1 \wedge M_5 \Rightarrow M_2$ can be considered as the strong association rules for the given problem.

Q. A database has five transactions. Let min sup = 60% and min conf = 80%.

TID	*Items_bought*
$T100$	$\{M, O, N, K, E, Y\}$
$T200$	$\{D, O, N, K, E, Y\}$
$T300$	$\{M, A, K, E\}$
$T400$	$\{M, U, C, K, Y\}$
$T500$	$\{C, O, O, K, I, E\}$

- ***Find all frequent itemsets using Apriori algorithm.***
- ***List all the strong association rules.***

Solution:

Given,

$Min\ sup\ =\ 60\% = 5 \times \frac{60}{100} = 3$

$Min\ conf\ =\ 80\%$

Step 1: Calculating C_1 and L_1

Candidate 1-itemsets (C_1)

Itemset	Sup_count
{M}	3
{O}	3
$\{N\}$	2
{K}	5
{E}	4
$\{Y\}$	3
$\{D\}$	1
$\{A\}$	1
$\{U\}$	1
{C}	2
$\{I\}$	1

Frequent 1-itemsets (L_1)

Itemset	Sup_count
{M}	3
{O}	3
{K}	5
{E}	4
$\{Y\}$	3

Step 2: Calculating C_2 and L_2

Candidate 2-itemsets (C_2)

Itemset	Sup_count
{M, O}	1
$\{M, K\}$	3
$\{M, E\}$	2
$\{M, Y\}$	2
$\{O, K\}$	3
$\{O, E\}$	3
$\{O, Y\}$	2
$\{K, E\}$	4
$\{K, Y\}$	3
$\{E, Y\}$	2

Frequent 2-itemsets (L_2)

Itemset	Sup_count
$\{M, K\}$	3
$\{O, K\}$	3
$\{O, E\}$	3
$\{K, E\}$	4
$\{K, Y\}$	3

Step 3: Calculating C_3 and L_3

Candidate 3-itemsets (C_3)

Itemset	Sup_count
$\{O, K, E\}$	3

Frequent 3-itemsets (L_3)

Itemset	Sup_count
$\{O, K, E\}$	3

Now, stop since no more combinations can be made in L_3.

∴ Frequent Itemset = $\{O, K, E\}$

Now,

Generating Association Rules from Frequent Itemsets:

Rules	Confidence
$O \Rightarrow K \wedge E$	3/3 = 100%
$K \Rightarrow O \wedge E$	3/5 = 60%
$E \Rightarrow O \wedge K$	3/4 = 75%
$O \wedge K \Rightarrow E$	3/3 = 100%
$O \wedge E \Rightarrow K$	3/3 = 100%
$K \wedge E \Rightarrow O$	3/4 = 75%

As the given threshold or minimum confidence is 80%, so the rules $O \Rightarrow K \wedge E$, $O \wedge K \Rightarrow E$ and $O \wedge E \Rightarrow K$ can be considered as the strong association rules for the given problem.

Advantages of Apriori Algorithm	*Limitations of Apriori Algorithm*
- This is easy to understand algorithm. - This algorithm has least memory consumption. - Easy implementation. - It uses Apriori property for pruning therefore, itemsets left for further support checking remain less.	- Using Apriori needs a generation of candidate itemsets. These itemsets may be large in number if the itemset in the database is huge. - Apriori needs multiple scans of the database to check the support of each itemset generated and this leads to high costs. These Limitations can be overcome using the ***FP growth algorithm***.

Methods to Improve Apriori's Efficiency

- ***Hash-based itemset counting:*** A k-itemset whose corresponding hashing bucket count is below the threshold cannot be frequent.
- ***Transaction reduction:*** A transaction that does not contain any frequent k-itemset is useless in subsequent scans.
- ***Partitioning:*** An itemset that is potentially frequent in DB must be frequent in at least one of the partitions of DB.
- ***Sampling:*** Mining on a subset of given data, lower support threshold and a method to determine completeness.
- ***Dynamic itemset counting:*** Add new candidate itemsets only when all of their subsets are estimated to be frequent.

Frequent Pattern (FP) Growth Algorithm

This algorithm is an improvement to the Apriori method. A frequent pattern is generated without the need for candidate generation. FP growth algorithm represents the database in the form of a tree called a frequent pattern tree or FP tree. This tree structure will maintain the association between the itemsets.

In FP Growth there are mainly two steps:

Step 1: Construction of FP-tree

It is built using two passes over the data-set.

Pass 1:

- Scan data and find support for each item.
- Discard infrequent items.
- Sort frequent items in decreasing order based on their support

Pass 2:

- Create the root of the tree, labeled with null
- Scan database again. Process items in each transaction in descending order of support count and create branch for each transaction
- If some transaction shares common prefix with already processed transactions, increment counter for each item in common prefix and create branches for items that are not common.
- To facilitate tree traversal, an item header table is also built so that each item points to its occurrences in the tree via a chain of node links.

Step 2: Extract Frequent Item-sets from FP-tree

- Construct its conditional pattern base, which is subset of database that contains the set of prefix paths in the FP-tree.
- Construct its (conditional) FP-tree. Conditional FP-tree of item I is FP-tree constructed only by considering transaction that contains item I and then removing the Item I from all transactions. Then perform mining recursively on such a tree. The pattern growth is achieved by the concatenation of the suffix pattern with the frequent patterns generated from a conditional FP-tree.

Example

Q. Consider the database, consisting of 9 transactions. Suppose min. support count required is 2. Let minimum confidence required is 70%. Find out the frequent item sets using FP-growth algorithm and then generate all strong association rules.

TID	*Items*
T1	*B , A , T*
T2	*A , C*
T3	*A , S*
T4	*B , A , C*
T5	*B , S*
T6	*A , S*
T7	*B , S*
T8	*B , A , S , T*
T9	*B , A , S*

Solution:

First of all, we need to create a table of item counts in the whole transactional database and sort the item list in descending order for their support count as below:

Item	Support Count
B	6
A	7
T	2
C	2
S	6

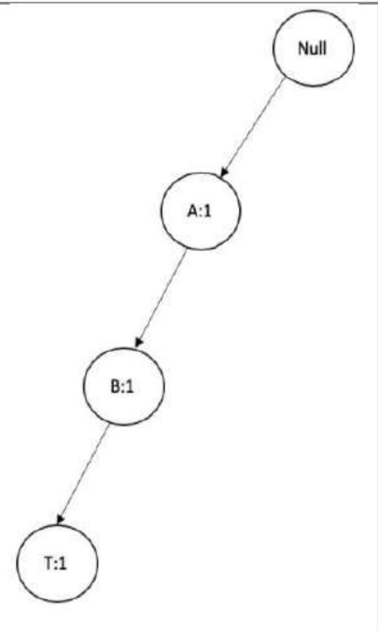
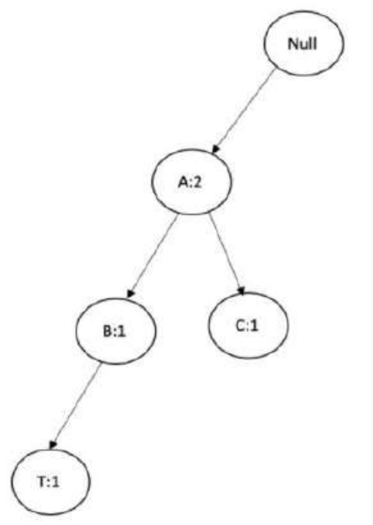
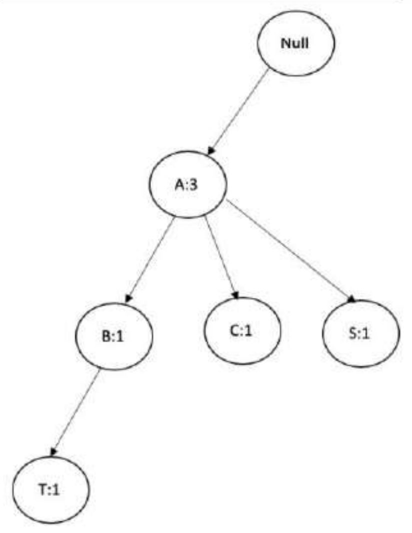
$\xrightarrow{\text{Sort frequent items in frequency descending order}}$

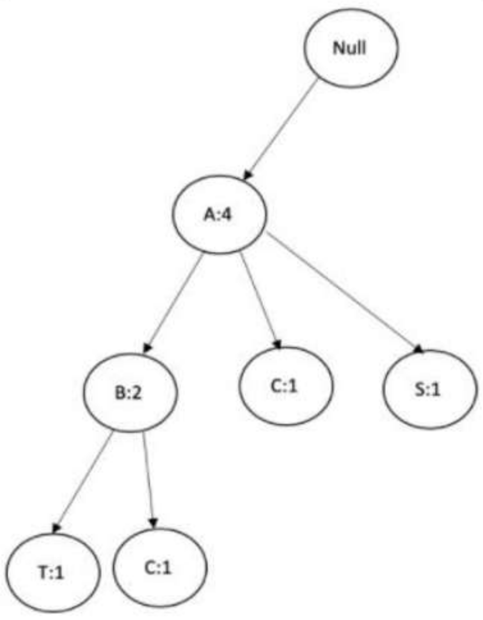
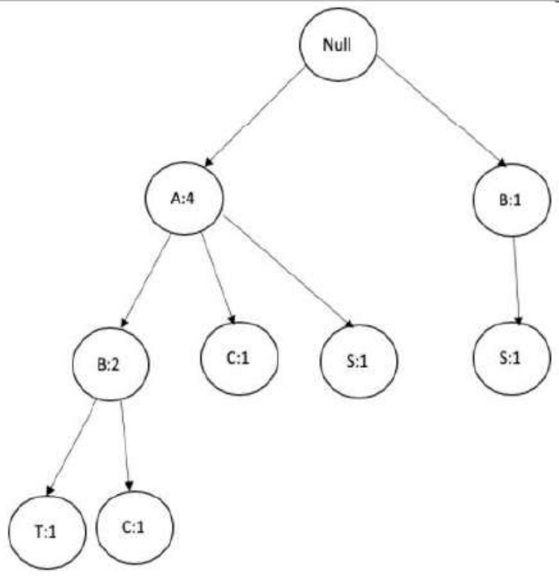
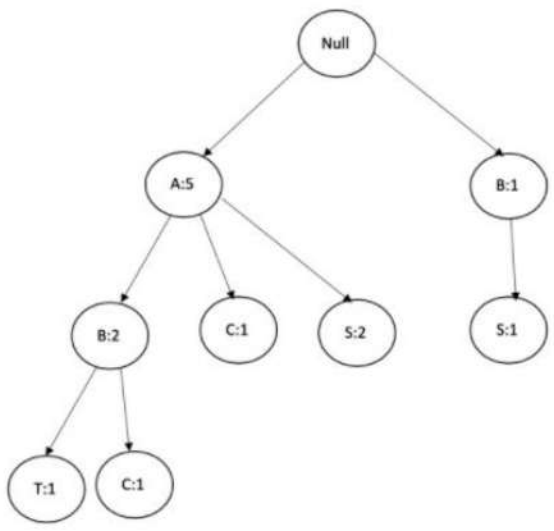
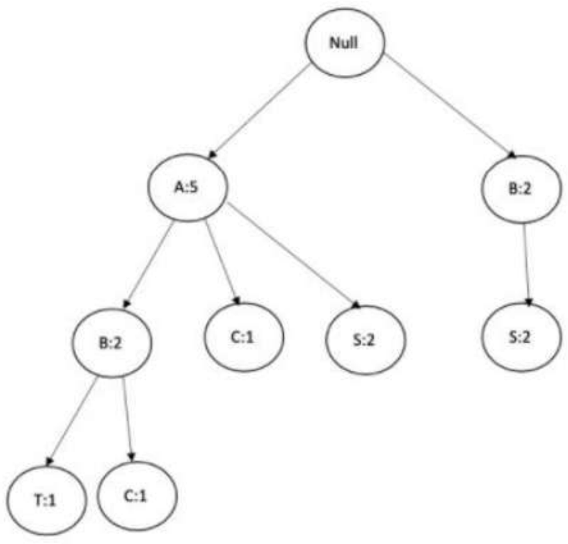
Item	Support Count
A	7
B	6
S	6
C	2
T	2

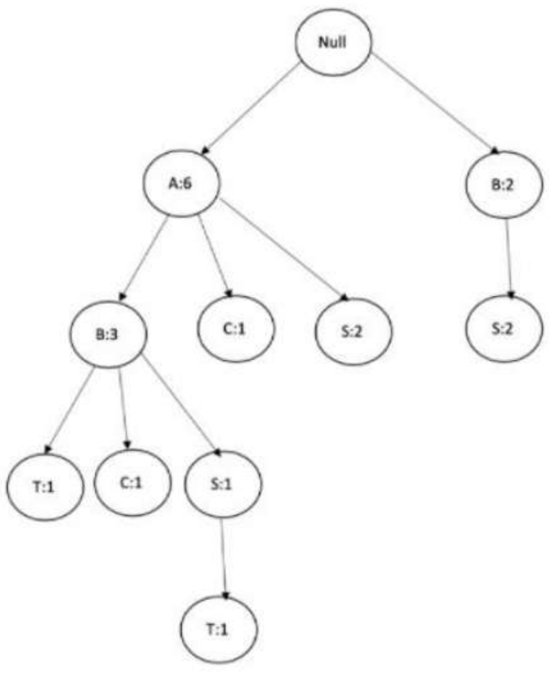
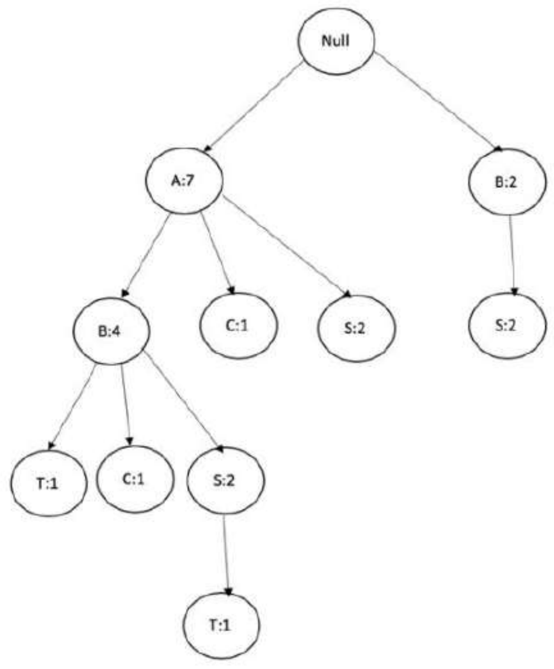
Now, for each transaction, the respective Ordered-Item set is built:

TID	***Items***	***Ordered Itemsets***
T1	B , A , T	A, B, T
T2	A , C	A, C
T3	A , S	A, S
T4	B , A , C	A, B, C
T5	B , S	B, S
T6	A , S	A, S
T7	B , S	B, S
T8	B , A , S , T	A, B, S, T
T9	B , A , S	A, B, S

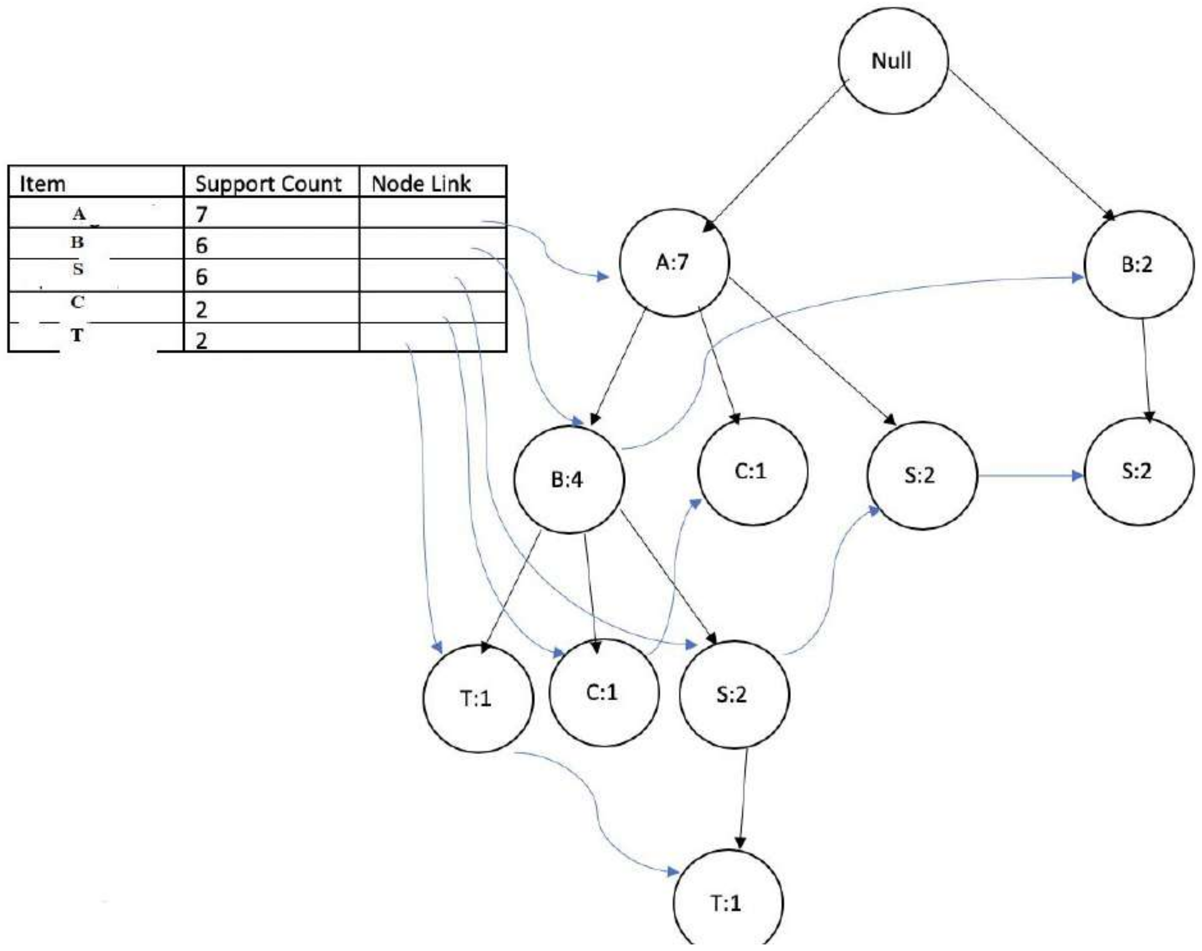
Now building FP-Tree by using ordered itemsets one by one:

1. After reading TID = T1	2. After reading TID = T2	3. After reading TID = T3
		

4. After reading TID = T4	5. After reading TID = T5
	
6. After reading TID = T6	7. After reading TID = T7
	

8. After reading TID = T8	9. After reading TID = T9
	

Now, to facilitate tree traversal, an item header table is built so that each item points to its occurrences in the tree via a chain of node-links.



Item	Support Count	Node Link
A	7	
B	6	
S	6	
C	2	
T	2	

Now we need to construct a table for conditional pattern base and hence, the frequent pattern generation.

Item	***Conditional Pattern Base***	***Conditional FP-tree***	***Frequent Pattern Generation***
T	{{A, B: 1}, {A, B, S: 1}}	<A: 2, B: 2>	{A, T: 2}, {B, T: 2}, {A, B, T: 2}
C	{{A, B: 1}, {A: 1}}	<A: 2>	{A, C: 2}
S	{{A, B: 2}, {A: 2}, {B: 2}}	<A: 4, B: 2>, <B: 2>	{A, S: 4}, {B, S: 4}, {A, B, S: 2}
B	{{A: 4}}	<A: 4>	{A, B: 4}

A is directly from the Null node and since there aren't any in-between nodes to reach *A*, there is no need to go for another row of *A*.

We have generated a 3-item frequent set as {A, B, T: 2} & {A, B, S: 2}. Similarly 2-Item frequent sets are {A, T: 2},{B, T: 2}, {A, C: 2}, {A, S: 4},{B, S: 4} & {A, B:4}.

Generate Association Rules:

Same as in Apriori algorithm.

Advantages of FP growth algorithm

- It allows frequent item set discovery without candidate generation.
- It builds a compact data structure called FP tree.
- Only two passes over dataset.
- Faster than Apriori algorithm.

Disadvantages of FP growth algorithm

- FP tree is difficult to build than Apriori.
- FP tree may not fit in memory.
- FP tree is expensive to build.

Apriori vs FP Growth

Apriori	***FP Growth***
It is slower than FP growth algorithm.	It is faster than Apriori algorithm.
It is an array-based algorithm.	It is a tree-based algorithm.
It uses Apriori, join and prune property.	It constructs conditional frequent pattern tree and conditional pattern base from database which satisfy minimum support.
Apriori utilizes a level-wise approach where it generates patterns containing 1 item, then 2 items, then 3 items, and so on.	FP Growth utilizes a pattern-growth approach means that, it only considers patterns actually existing in the database.
It requires large memory space due to large number of candidate generation.	It requires less memory space due to compact structure and no candidate generation.
It scans the database multiple times for generating candidate sets.	It scans the database only twice for constructing frequent pattern tree.
Candidate generation is very parallelizable.	Data are very interdependent, each node needs the root.

From Association Mining to Correlation Analysis (Lift)

Most association rule mining algorithms employ a support-confidence framework. Often, many interesting rules can be found using low support thresholds. Although minimum support and confidence thresholds *help* weed out or exclude the exploration of a good number of uninteresting rules, many rules so generated are still not interesting to the users.

The support and confidence measures are insufficient at filtering out uninteresting association rules. To tackle this weakness, a correlation measure can be used to augment the support-confidence framework for association rules. This leads to *correlation rules* of the form

$$\boldsymbol{A \Rightarrow B} \, [\boldsymbol{support, confidence, correlation}]$$

That is, a correlation rule is measured not only by its support and confidence but also by the correlation between itemsets A and B. There are many different correlation measures from which to choose. ***Lift*** is simple correlation measure that is given as follows:

Lift: The lift between the occurrence of A and B can be measured as below:

$$lift(A, B) = \frac{P(A \cup B)}{P(A)P(B)}.$$

- If the $lift(A, B)$ is less than 1, then the occurrence of A is negatively correlated with the occurrence of B.
- If the resulting value is greater than 1, then A and B are positively correlated, meaning that the occurrence of one implies the occurrence of the other.
- If the resulting value is equal to 1, then A and B are independent and there is no correlation between them.

Please let me know if I missed anything or anything is incorrect.

poudeljayanta99@gmail.com