DATA MINING

PATTERN AND ASSOCIATION MINING 1

OVERVIEW

Brief introduction to sets

Pattern-mining basics

Apriori algorithm

INTRODUCTION TO SETS

SETS

"A set is a gathering together into a whole of definite, distinct objects of our perception (Anschauung) or of our thought—which are called elements of the set."

—Georg Cantor 1895

SETS

"A set is a well-defined collection of distinct objects, considered as an object in its own right."

—Wikipedia 2017

SETS — EXAMPLES

A = {3, 1, 5}

B = 

CARDINALITY

$A = \{3, 1, 5\}$

$|A| = 3$

$B =$

$|B| = 5$

CARDINALITY

- $\varnothing \equiv \{\}$

  $|\varnothing| = 0$

CARDINALITY

- $\varnothing \equiv \{\}$

  $|\varnothing| = 0$

- $\mathbf{N} \equiv \{0, 1, 2, \ldots\}$

  $|\mathbf{N}| = \infty$

CARDINALITY

- $\varnothing \equiv \{\}$

  $|\varnothing| = 0$

- $\mathbf{N} \equiv \{0, 1, 2, \ldots\}$

  $|\mathbf{N}| = \infty$

- $\mathbf{R} \equiv \{\text{ real numbers}\}$

  $|\mathbf{R}| = \infty$

CARDINALITY

- $\varnothing \equiv \{\}$

  $|\varnothing| = 0$

- $\mathbf{N} \equiv \{0, 1, 2, \ldots\}$

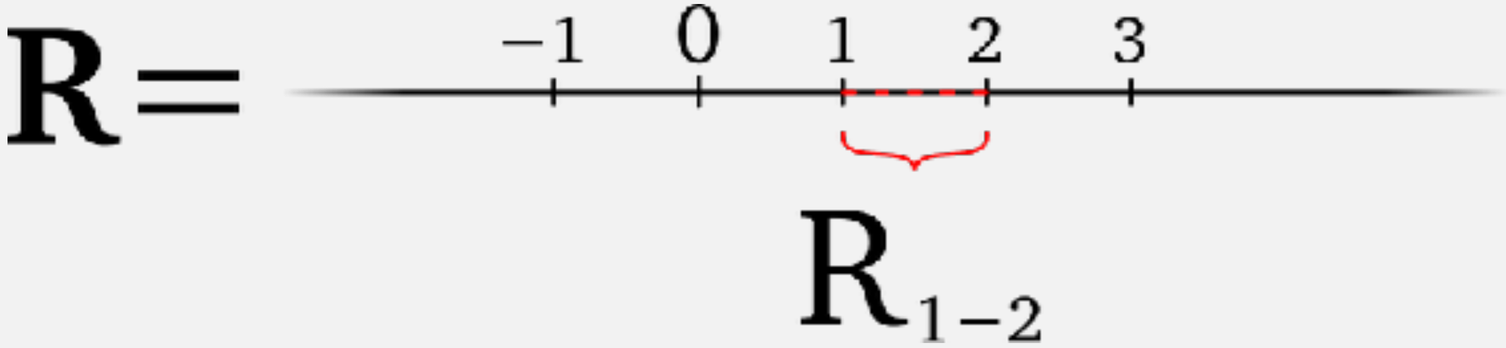
  $|\mathbf{N}| = \infty$

- $\mathbf{R} \equiv \{\text{ real numbers}\}$

  $|\mathbf{R}| = \infty$

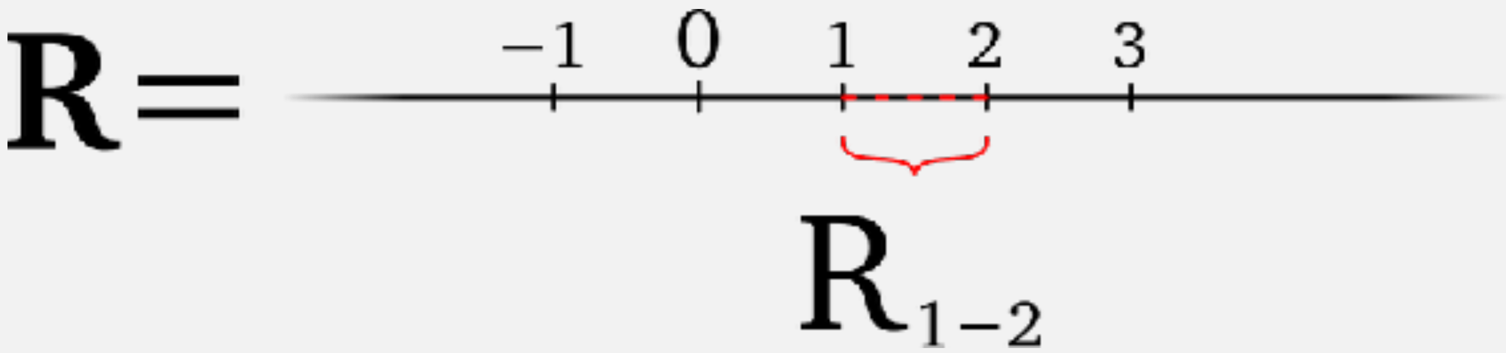
- $|\mathbf{R}| > |\mathbf{N}|$

CARDINALITY



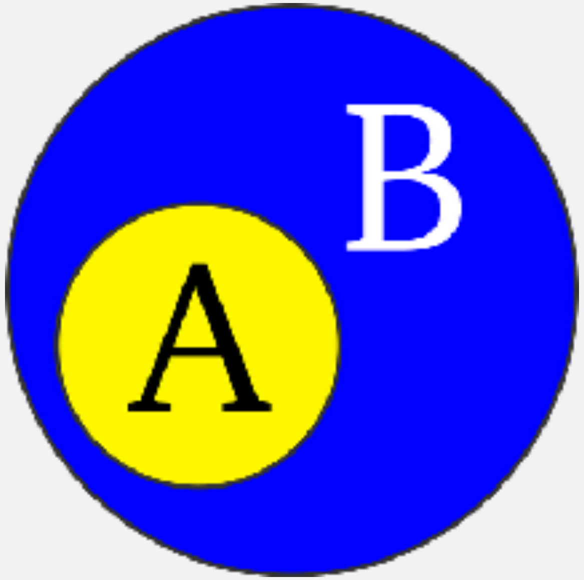
- **R** ≡ { real numbers}
- $R_{1-2} = \{ x \in \mathbf{R} \mid 1 < x < 2\}$
- There are elements in **R** that are **NOT** part of R_{1-2}

CARDINALITY



- **R** ≡ { real numbers}
- $R_{1-2} = \{ x \in \mathbf{R} \mid 1 < x < 2\}$
- There are elements in **R** that are **NOT** part of R_{1-2}
- **But $|R_{1-2}| = |\mathbf{R}|$**

SUBSET-SUPERSET



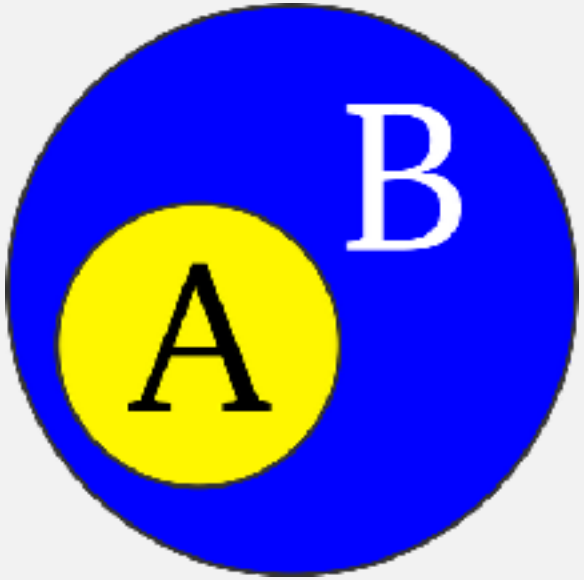
Subset

- A and B are sets
- All elements of A are also elements of B
- $A \subseteq B$

Superset

- If A is a subset of B, B is a superset of A
- $B \supseteq A$

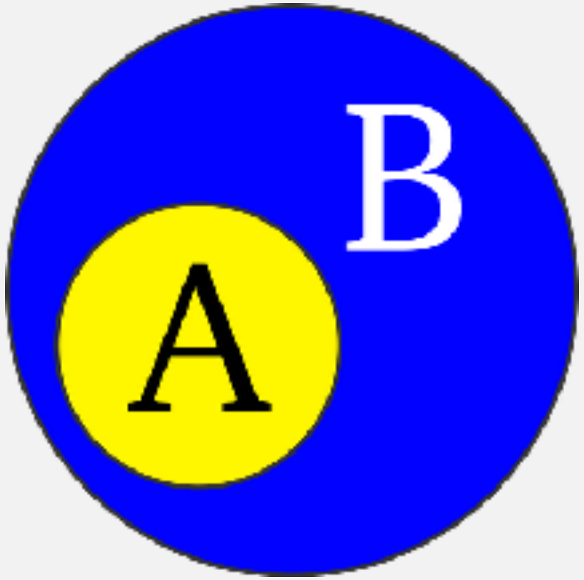
SUBSET-SUPERSET



Proper Subset-superset

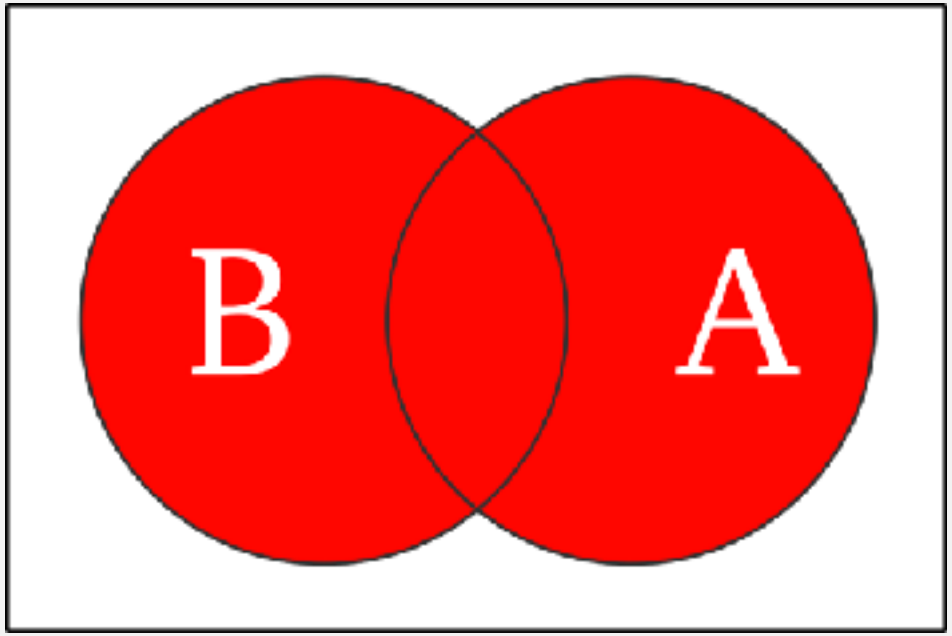
- A and B are sets
- All elements of A are also elements of B
- **There is at least one element in B not in A**
- $A \subsetneq B$
- $B \supsetneq A$

SUBSET-PROPER SUBSET



- If A is a subset of B, A and B **may or may not** be equal
- If A is a proper subset of B, A definitely does **not** equal B
- Sometimes $\subset$ and $\supset$ instead of $\subsetneq$ and $\supsetneq$ (analogy to $\{<, >, \leq, \geq\}$)

UNION

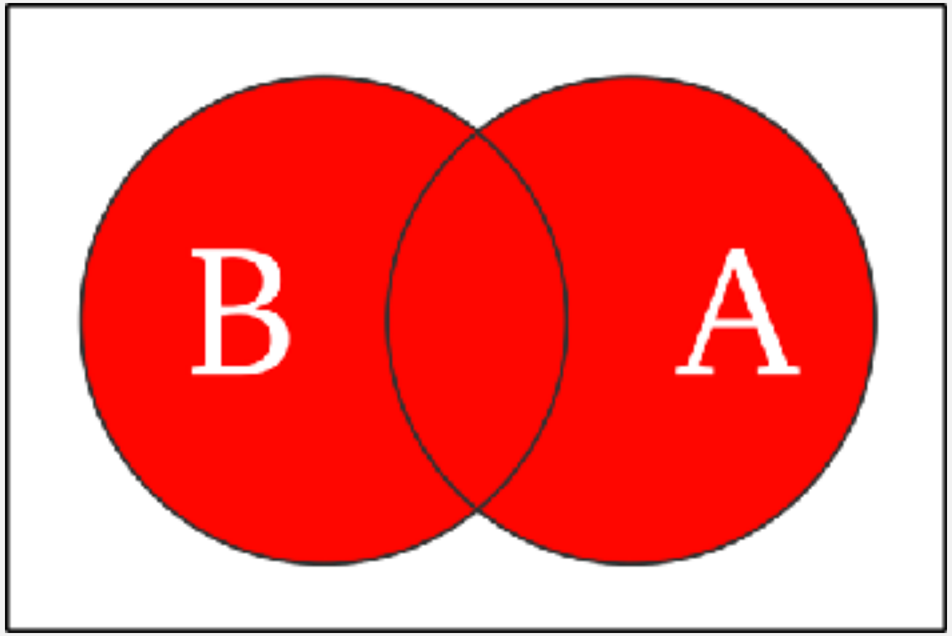


- *"The union of A and B is the set of all things that are members of either A or B"*
- **A** ∪ **B**
- {1, 2} ∪ {5, 2} = {1, 2, 5}

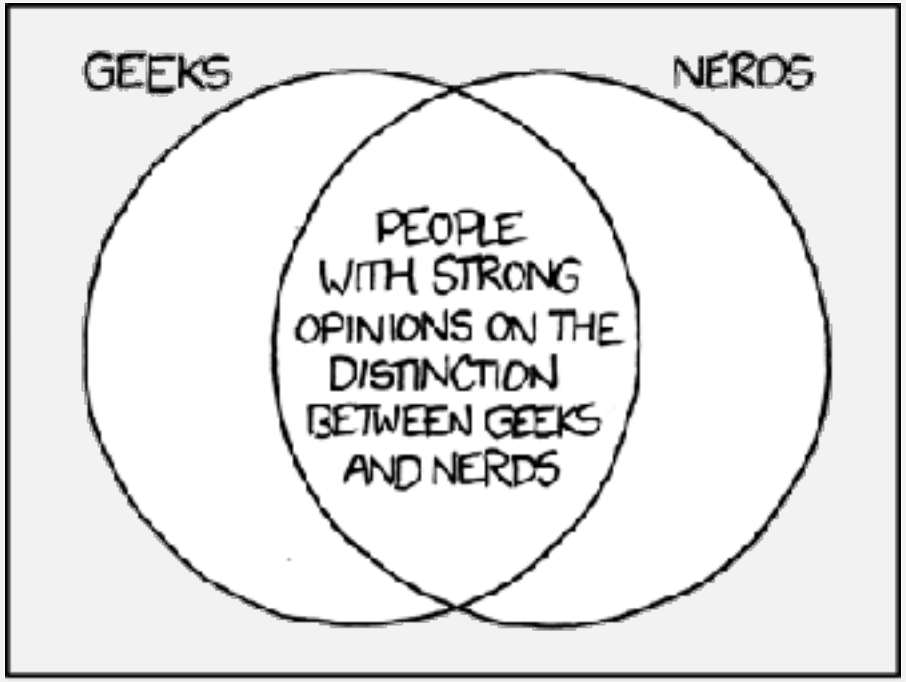
UNION

- $A \cup B = B \cup A$
- $A \cup (B \cup C) = (A \cup B) \cup C$
- $A \subseteq (A \cup B)$
- $A \cup A = A$
- $A \cup \varnothing = A$
- $A \subseteq B \Leftrightarrow A \cup B = B$

(From Wikipedia)



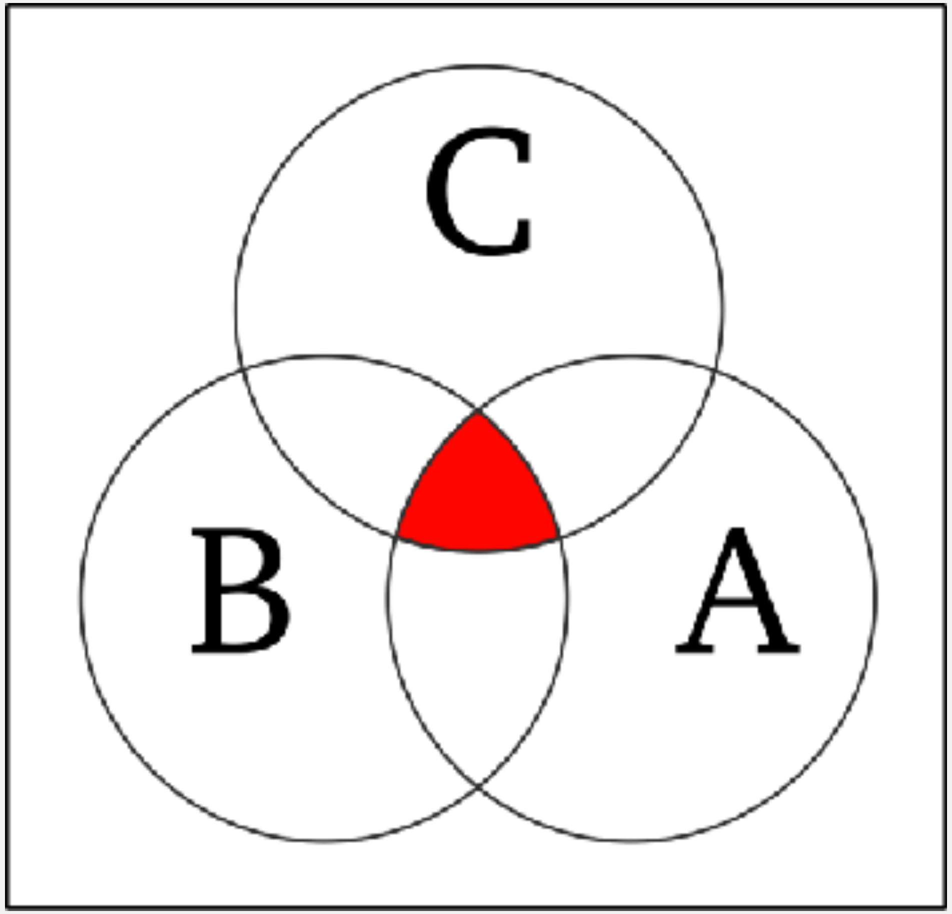
INTERSECTION



https://xkcd.com/747/

- All things that are members of both A and B
- If the intersection is null, the sets are called disjoint
- **A ∩ B**
- {1, 2} ∪ {5, 2} = {2}

INTERSECTION



- $A \cap B = B \cap A$
- $A \cap (B \cap C) = (A \cap B) \cap C$
- $A \cap B \subseteq A$
- $A \cap A = A$
- $A \cap \varnothing = \varnothing$
- $A \subseteq B \Leftrightarrow A \cap B = A$

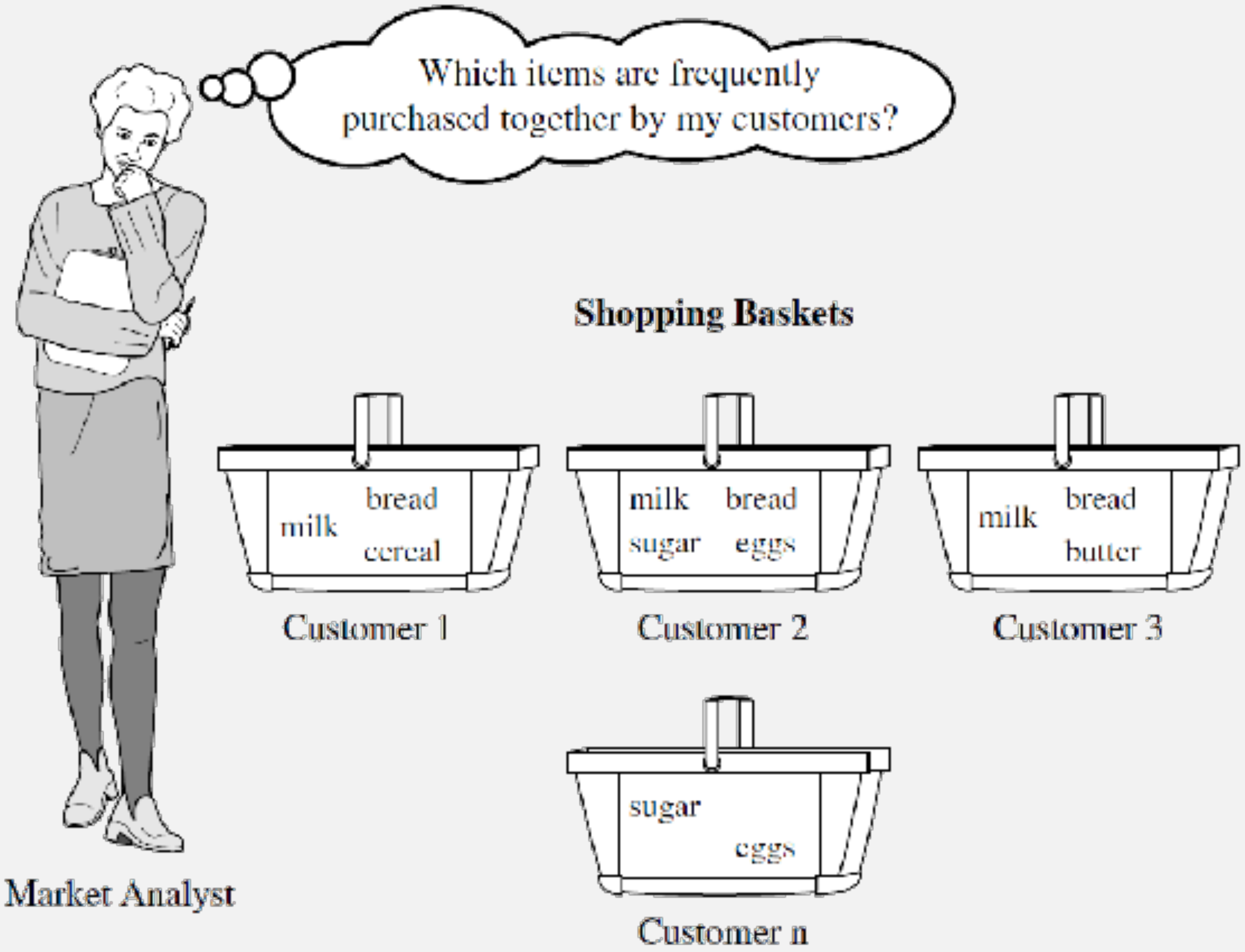
(From Wikipedia)

PATTERN-MINING BASICS

FREQUENT-PATTERN MINING

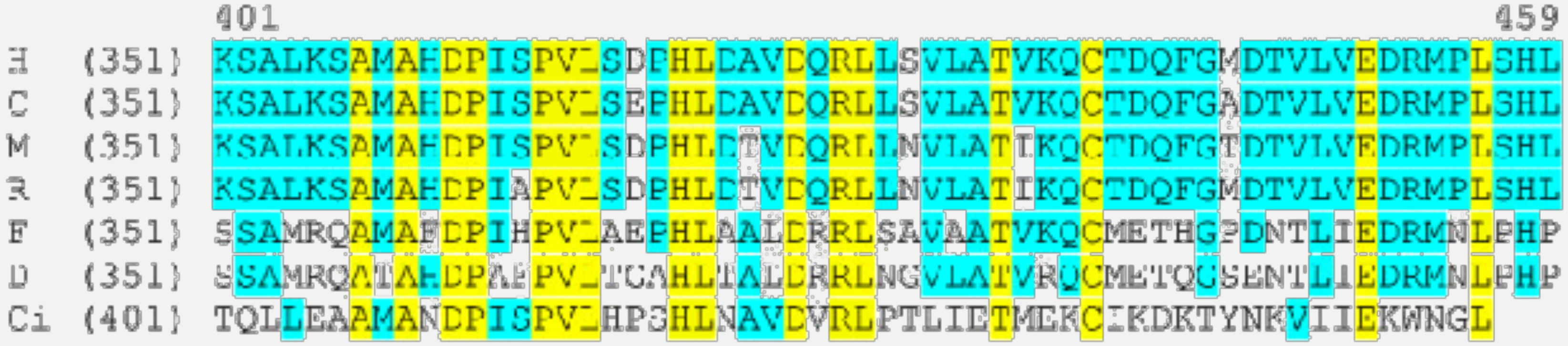
What does it mean?

EXAMPLE: MARKET BASKET ANALYSIS



EXAMPLE: OVERWATCH TEAM COMPOSITION

SCIENCE



- What DNA mutations are associated with a given condition?
- What DNA mutations occur together?
- Preserved DNA sequences through evolution
- Common amino acid sequences in proteins?
- Specific weather conditions previous to certain events?
- Seismic history for earthquake prediction?

FREQUENT-PATTERN MINING

- **Itemset**

  Set of items that occur together (e.g., computer **and** camera)

  Today

- **Subsequece**

  Sequential set (e.g., computer **then** camera)

  Later in the course

ITEMSET

- An item set with k items is a k-itemset

  E.g., the itemset {boat, circle, acid} is a 3-itemset

- **Occurrence frequency**

  Number of times an itemset is in the data set

  Other names: frequency, **support count**, count, **absolute support**

ASSOCIATION RULES

A way to represent frequent patterns (A, B)

- **Support**

  Percentage of tuples in the data set with both A and B, $(A \cup B)$

  Also: relative support

- **Confidence**

  Of the tuples that have A, percentage that also have B

ASSOCIATION RULES

- **Notation**

  A ⇒ B [support = x%, confidence = y%]

  computer ⇒ antivirus software [support = 2%, confidence = 60%]

- **Threshold values** for support and confidence

  Association rules that satisfy both: **strong**

ASSOCIATION RULES—CONFIDENCE

Of the tuples that have A, percentage that also have B, in other words probability of B given A:

$$confidence\left(A \Rightarrow B\right) = P\left(B \mid A\right) = \frac{support\left(A \cup B\right)}{support\left(A\right)} = \frac{support\ count\left(A \cup B\right)}{support\ count\left(A\right)}$$

ASSOCIATION RULES—CONFIDENCE

Of the tuples that have A, percentage that also have B, in other words probability of B given A:

$$confidence\left(A \Rightarrow B\right) = P\left(B \mid A\right) = \frac{support\left(A \cup B\right)}{support\left(A\right)} = \frac{support\ count\left(A \cup B\right)}{support\ count\left(A\right)}$$

Beware of loose notation

ASSOCIATION RULES

Notice!

$A \Rightarrow B \neq B \Rightarrow A$

Client shopping list

Eyes, **wand**, spider leg

Shield, **eyes**, **wand**

Wand, robe, staff

Spider leg, green potion

Wand, truffles

Old wine, **wand**, **eyes**

ASSOCIATION RULES—EXERCISE

Find **support** and **confidence**

Support: % with set ($A \cup B$)

Confidence: % of A with also B

- Bread ⇒ Milk

  [support = ?,
  confidence = ?]

- Newspaper ⇒ Tomato

  [support = ?,
  confidence = ?]

Client shopping list

- Bread, milk, butter, newspaper
- Bread, milk
- Butter, newspaper, asparagus
- Bread, milk, tortellini, batteries
- Tortellini, asparagus, mozzarella
- Bread, milk, butter
- Newspaper, asparagus, tomato
- Newspaper, tomato
- Asparagus, batteries, cigarettes
- Bread, milk

ASSOCIATION RULES—EXERCISE

Find **support** and **confidence**

Support: % with set $(A \cup B)$

Confidence: % of A with B

- Bread ⇒ Milk

  [support = 50%,
  confidence = 100%]

- Newspaper ⇒ Tomato

  [support = 20%,
  confidence = 50%]

Client shopping list

- Bread, milk, butter, newspaper
- Bread, milk
- Butter, newspaper, asparagus
- Bread, milk, tortellini, batteries
- Tortellini, asparagus, mozzarella
- Bread, milk, butter
- Newspaper, asparagus, tomato
- Newspaper, tomato
- Asparagus, batteries, cigarettes
- Bread, milk

TWO-STEP ASSOCIATION-RULE MINING

Because

$$confidence\,(A \Rightarrow B) = P\,(B \mid A) = \frac{support\,(A \cup B)}{support\,(A)} = \frac{support\ count\,(A \cup B)}{support\ count\,(A)}$$

we **only need** to know the **support** of A and A ∪ B

to test if an association rule is strong.

TWO-STEP ASSOCIATION-RULE MINING

1. Find all frequent itemsets
2. Using support values, return strong association rules

TWO-STEP ASSOCIATION-RULE MINING

1. Find all frequent itemsets

Problem:

If an itemset is frequent, all its subsets are frequent!

TWO-STEP ASSOCIATION-RULE MINING

Example:

The 100-itemset {a1, a2, …, a100} contains:

{a1}, …, {a100}, {a1, a2}, {a1, a3}, …, {a1, a100}, {a2, a3}, etc.

Total:

$$\binom{100}{1} + \binom{100}{2} + \cdots + \binom{100}{100} = 2^{100} - 1 \approx 1.27 \times 10^{30}$$

TWO-STEP ASSOCIATION-RULE MINING

TOO MANY

Example:

The 100-itemset {a1, a2, …, a100} contains:

{a1}, …, {a100}, {a1, a2}, {a1, a3}, …, {a1, a100}, {a2, a3}, etc.

Total:

$$\binom{100}{1} + \binom{100}{2} + \cdots + \binom{100}{100} = 2^{100} - 1 \approx 1.27 \times 10^{30}$$

CLOSED/MAXIMAL ITEMSET

- **Closed itemset**
  "A" is closed in data set S if there is no proper super-itemset B such that B has the same support as A
- **Maximal frequent itemset**
  "A" is frequent and there is no proper super-itemset B that is also frequent in S

CLOSED/MAXIMAL ITEMSET—EXERCISE

Itemset and support. Threshold 30%

- {A, B} 50%
- {A, B, C} 40%
- {A, B, C, D} 40%
- {A, B, C, D, E} 35%
- {A, B, C, D, E, F} 29%

*(Consider no other itemsets)

- **Closed itemset**
  No proper super-itemset that has the same support
- **Maximal frequent itemset**
  No proper super-itemset that is also frequent

CLOSED/MAXIMAL ITEMSET—EXERCISE

Itemset and support. Threshold 30%

• {A, B} 50%	Closed, not maximal
• {A, B, C} 40%	Not closed, not maximal
• {A, B, C, D} 40%	Closed, not maximal
• {A, B, C, D, E} 35%	Closed, maximal
• {A, B, C, D, E, F} 29%	Not frequent!

*(Consider no other itemsets)

CLOSED/MAXIMAL ITEMSET

Note there can be more than one maximal itemset!

- {A, B} 50%
- {A, B, C} 12%
- {B, C, D} 40%
- {B, C, D, E} 25%

Maximal sets

CLOSED/MAXIMAL ITEMSET

Are these itemsets closed?

- {Bread, milk}
- {Bread, butter}

Client shopping list

- Bread, milk, butter, newspaper
- Bread, milk
- Butter, newspaper, asparagus
- Bread, milk, tortellini, batteries
- Tortellini, asparagus, mozzarella
- Bread, milk, butter
- Newspaper, asparagus, tomato
- Newspaper, tomato
- Asparagus, batteries, cigarettes
- Bread, milk

CLOSED/MAXIMAL ITEMSET

Are these itemsets closed?

- {Bread, milk}

  Yes

- {Bread, butter}

  No

Client shopping list

- Bread, milk, butter, newspaper
- Bread, milk
- Butter, newspaper, asparagus
- Bread, milk, tortellini, batteries
- Tortellini, asparagus, mozzarella
- Bread, milk, butter
- Newspaper, asparagus, tomato
- Newspaper, tomato
- Asparagus, batteries, cigarettes
- Bread, milk

CLOSED/MAXIMAL ITEMSET

Are these itemsets closed?

- {Bread, milk}

  Yes

- {Bread, butter}

  No {Bread, butter, milk}

Client shopping list

- Bread, milk, butter, newspaper
- Bread, milk
- Butter, newspaper, asparagus
- Bread, milk, tortellini, batteries
- Tortellini, asparagus, mozzarella
- Bread, milk, butter
- Newspaper, asparagus, tomato
- Newspaper, tomato
- Asparagus, batteries, cigarettes
- Bread, milk

APRIORI ALGORITHM

APRIORI

A priori

"Relating to or denoting reasoning or knowledge which proceeds from theoretical deduction rather than from observation or experience."

—Google

APRIORI

- **Goal**

  To obtain all frequent itemsets

- **Apriori property**

  All non-empty subsets of a frequent itemset are also frequent

- **Pruning principle**

  If an itemset contains a subset that is not frequent it can be discarded

APRIORI—ALGORITHM IDEA

- Find frequent 1-itemsets
- Generate k-candidates using (only) frequent (k–1)-itemsets
- Test frequency of candidates
- Terminate if no more candidates or all candidates are infrequent

APRIORI—CANDIDATE GENERATION

Two-step process

- Join
- Prune

APRIORI—JOIN STEP

- Input: frequent (k–1)-itemsets: L_{k-1}
- Elements in each itemset are **ordered**
- Itemsets in L_{k-1} are combined if their first k–2 elements are the same. **Only the last may differ**!
- Example

  $L_3 = \{abc, abd, acd, ace, bcd\}$

  abc ∪ abd ⇒ abcd

  acd ∪ ace ⇒ acde

JOIN STEP—NOTE

- Elements in each itemset are **ordered**
- You can choose how to order your itemset elements
  as long as you are consistent. This won't affect the results!

APRIORI—PRUNE STEP

- Some candidates can be discarded before testing!
- If any subset C_i is not frequent ($C_i \notin L_{k-1}$) the candidate is **not** frequent, and is discarded

APRIORI—PRUNE STEP

Example (continues)

- L_3 = {abc, abd, acd, ace, bcd}
- Join returns two candidates: C_1 = **abcd**, C_2 = **acde**
- $\mathbf{C_1}$, k–1 **subset**:

  abc, **abd**, **acd**, **bcd**
- $\mathbf{C_2}$, k–1 **subset**:

  acd, **ace**, **ade**, **cde**
- $\mathbf{C_2}$ **is pruned!**

APRIORI—STEP-BY-STEP

TID	Items
10	A, C, D
20	B, C
30	A, B, C, E
40	B, E

Let us apply Apriori to this data set, containing a list of items for each transaction ID.

APRIORI—STEP-BY-STEP

1. **Scan** for frequent 1-itemsets

TID	Items
10	A, C, D
20	B, C
30	A, B, C, E
40	B, E

C_1	Support
{A}	2
{B}	3
{C}	3
{D}	1
{E}	2

APRIORI—STEP-BY-STEP

2. **Discard** non frequent candidates: we have L_1

C_1	Support
{A}	2
{B}	3
{C}	3
{D}	1
{E}	2

→

L_1
{A}
{B}
{C}
{E}

APRIORI—STEP-BY-STEP

3. **Join** step to produce k = 2 candidates

L_1
{A}
{B}
{C}
{E}

→

C_2
{A, B}
{A, C}
{A, E}
{B, C}
{B, E}
{C, E}

APRIORI—STEP-BY-STEP

4. **Pruning** (trivial for k = 2)

C_2
{A, B}
{A, C}
{A, E}
{B, C}
{B, E}
{C, E}

C_2	Subset	In L_1?
{A, B}	{A}	yes
	{B}	yes
{A, C}	{A}	yes
	{C}	yes
{A, E}	{A}	yes
	{E}	yes
{B, C}	{B}	yes
	{C}	yes
{B, E}	{B}	yes
	{E}	yes
{C, E}	{C}	yes
	{E}	yes

APRIORI—STEP-BY-STEP

5. **Scan** and **discard** non-frequent candidates: we have L_2

C_2	Support
{A, B}	1
{A, C}	2
{A, E}	1
{B, C}	2
{B, E}	2
{C, E}	1

L_2	Support
{A, C}	2
{B, C}	2
{B, E}	2

APRIORI—STEP-BY-STEP

6. **Join** step to produce k = 3 candidates

L_2	Support
{A, C}	2
{B, C}	2
{B, E}	2

C_3
{B, C, E}

APRIORI—STEP-BY-STEP

7. **Prune**

C_3
{B, C, E}

C_3	Subset	In L_2?
{B, C, E}	{B, C}	yes
	{B, E}	yes
	{C, E}	no

APRIORI—STEP-BY-STEP

8. All C_3 candidates have been rejected. **End** and **return** L_1, L_2

TID	Items
10	A, C, D
20	B, C
30	A, B, C, E
40	B, E

L_2
{A, C}
{B, C}
{B, E}

L_1
{A}
{B}
{C}
{E}

Algorithm: Apriori. Find frequent itemsets using an iterative level-wise approach based on candidate generation.

Input:

- D, a database of transactions;
- *min_sup*, the minimum support count threshold.

Output: L, frequent itemsets in D.

Method:

L_1 = find_frequent_1-itemsets(D);
for ($k = 2; L_{k-1} \neq \phi; k{+}{+}$) {
    C_k = apriori_gen(L_{k-1});
    for each transaction $t \in D$ { // scan D for counts
        C_t = subset(C_k, t); // get the subsets of t that are candidates
        for each candidate $c \in C_t$
            c.count++;
    }
    $L_k = \{c \in C_k | c.count \geq min\_sup\}$
}
return $L = \cup_k L_k$;

```
procedure apriori_gen(L_{k-1}:frequent (k − 1)-itemsets)
    for each itemset l_1 ∈ L_{k-1}
        for each itemset l_2 ∈ L_{k-1}
            if (l_1[1] = l_2[1]) ∧ (l_1[2] = l_2[2]) ∧ ... ∧ (l_1[k − 2] = l_2[k − 2]) ∧ (l_1[k − 1] < l_2[k − 1]) then {
                c = l_1 ⋈ l_2; // join step: generate candidates
                if has_infrequent_subset(c, L_{k-1}) then
                    delete c; // prune step: remove unfruitful candidate
                else add c to C_k;
            }
    return C_k;

procedure has_infrequent_subset(c: candidate k-itemset;
            L_{k-1}: frequent (k − 1)-itemsets); // use prior knowledge
    for each (k − 1)-subset s of c
        if s ∉ L_{k-1} then
            return TRUE;
    return FALSE;
```

ASSOCIATION RULES FROM FREQ. ITEMSETS

Remember: $$confidence\,(A \Rightarrow B) = \frac{support\,(A \cup B)}{support\,(A)}$$

- For each frequent itemset l, generate all non-empty subsets s_i
- Generate rules: $s_i \Rightarrow l - s_i$

  (note $l - s_i = l \cap s_i$, e.g., ABC – A = BC = ABC ∩ A)
- Because $s_i \cup (l \cap s_i) = l$, return the rule if

$$\frac{support\,(l)}{support\,(s_i)} > threshold$$

BEYOND APRIORI

IMPROVING APRIORI

Sampling

Use a subset of the complete data

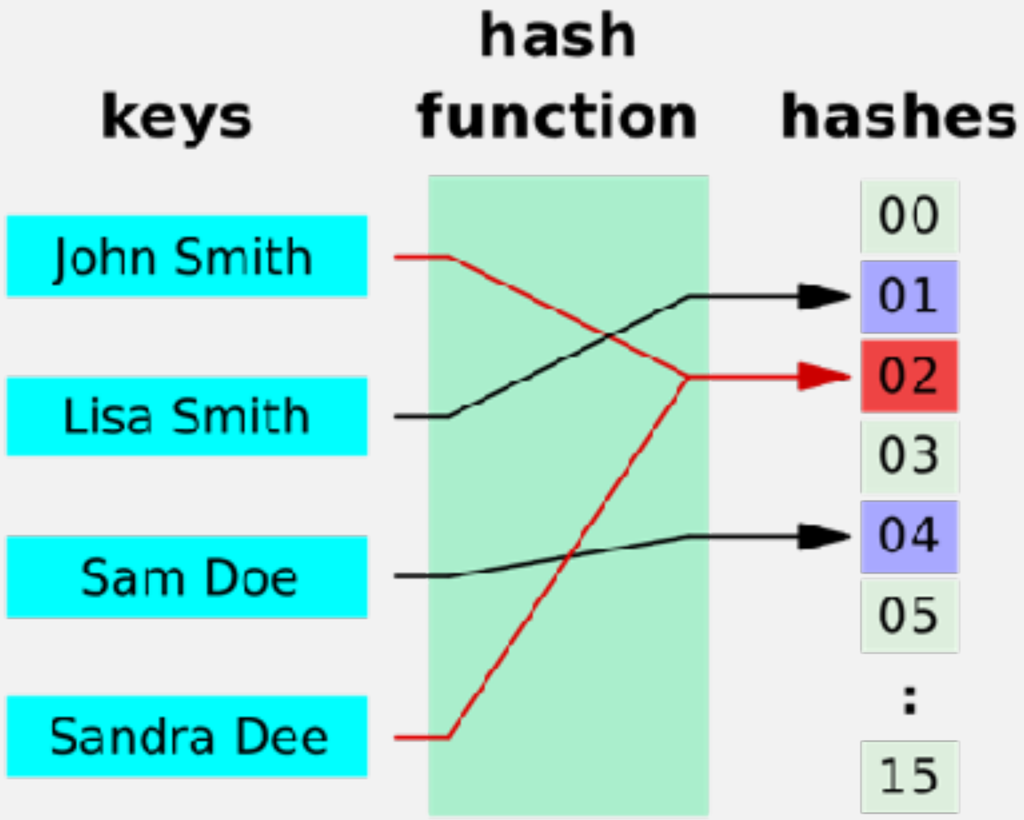
Faster, but may miss some frequent itemsets (tradeoff)

Use lower accuracy threshold to compensate!

REDUCED CANDIDATE SETS WITH HASH

Do you know what a **hash** is?

REDUCED CANDIDATE SETS WITH HASH



REDUCED CANDIDATE SETS WITH HASH

Idea:

- When calculating counts for k-candidates, make all (k+1)-itemsets
  for each itemset
- Classify new (k+1)-itemsets with hash and increase count of each bucket every time one instance is added

REDUCED CANDIDATE SETS WITH HASH

Example

Currently counting support for k = 1

Support threshold: 2

TID	Items
01	{B, D, E}
02	{A, C}
03	{B, D}

→

Hash class	0	1	2
Count	2	2	1
	{B, D}	{B, E}	{A, C}
	{B, D}	{D, E}	

REDUCED CANDIDATE SETS WITH HASH

Hash class	0	1	2
Count	2	2	1
	{B, D}	{B, E}	{A, C}
	{B, D}	{D, E}	

Conservative approach!

REDUCED CANDIDATE SETS WITH HASH

Why not each itemset its own bucket?

Hash class	0	1	2
Count	2	2	1
	{B, D}	{B, E}	{A, C}
	{B, D}	{D, E}	

IMPROVING APRIORI

Reduce data set in future iterations

Any itemset that does not contain any frequent k-itemset will **not** contain any (k+1)-itemsets!

BEYOND APRIORI

Other improvements and alternatives in the book

(Suggested: Mining frequent itemsets without candidate generation)

Read section 6.2.

THANKS FOR LISTENING!