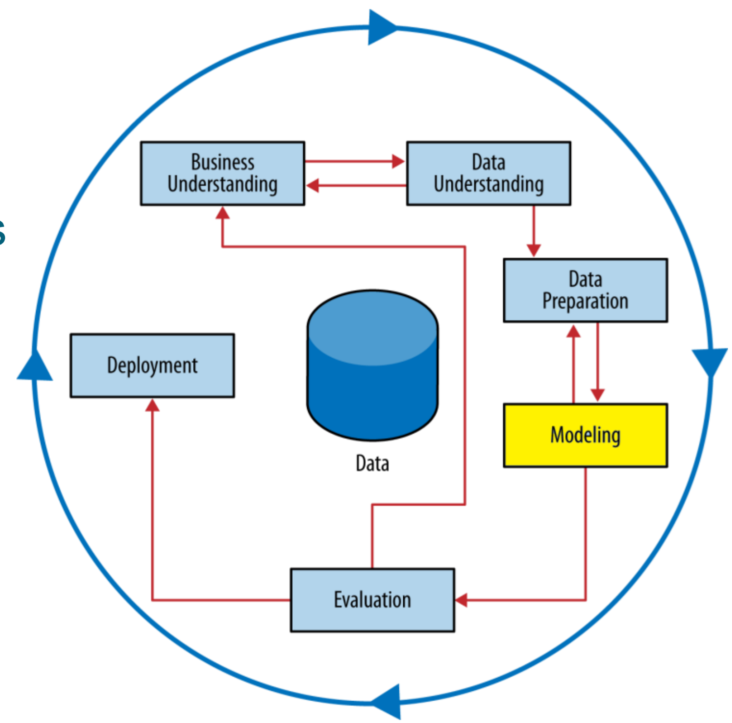
Association Rules and Itemset Mining

PF12 & EMC5

Association Rules

- **Association rules**, also called market basket analysis, discovers interesting **relationships** between **frequent itemsets**.
  - Unsupervised learning method

- Key techniques:
  - Frequent itemsets
    - Apriori algorithm
  - Association rules and evaluation metrics
    - Support
    - Confidence
    - Lift
    - Leverage
  - Similarity of itemset and items
    - Jaccard Similarity
  - Extension: from itemset to sequence



Association Rules

- A **transaction database** stores multiple transaction records.
  - Each transaction record contains one or multiple items.

Transaction ID	A	B	C	D	E	F
T_1	1	0	1	1	0	0
T_2	0	1	0	1	0	0
T_3	1	1	1	0	1	0
T_4	0	1	0	1	0	1

- **Association rules** explore the relationship between itemsets in a large transaction database.

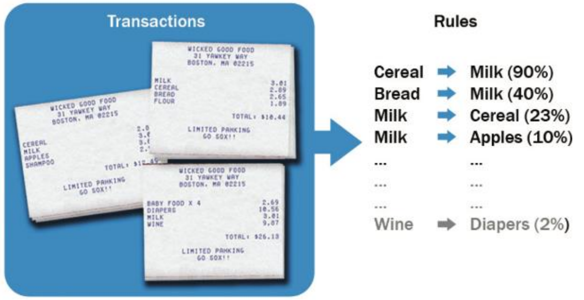


FIGURE 5-1 *The general logic behind association rules*

Itemset and Frequent Itemset

- **Itemset**: a set of items that appear together (in a transaction).
  - No order, no quantity
  - All items are unique

$$X = \{🐇, 🐂, 🍑, 🍉, 🍊\}$$

  - An itemset containing *k* items is a ***k*-itemset**: X = $\{X_1, X_2, \dots, X_k\}$

- **Support**: the frequency of itemset X in a database.
  - **Absolute support**: the number of transactions that contain X.
  - **Relative support**: the percentage of transactions that contain X.
    - i.e., the probability that a transaction contains X.

- X is considered as a **frequent itemset** if its support value $\sup(X)$ meets the threshold (i.e., $\min\_\sup$).

Example: Shopping Baskets

TID	Items Bought
1	🍺 🍼 🍉
2	🍺 🍭 🍼 🍋
3	🍼 🍺 🍋
4	🍼 🍭
5	🍷 🍪 🍺 🍞 🍋

{🍺}: support = 80%

{🍺, 🍼}: support = 60%

{🍺, 🍼, 🍋}: support = 40%

If *min_sup* = 50%, then
{🍺} and {🍺, 🍼} are frequent

The Apriori Principle

- If every item in itemset ***A*** also appears in itemset ***B***, then ***A*** is a **subset** of ***B***, and ***B*** is a **superset** of ***A*** .
  - $A = \{X_1, X_2\}$
  - $B = \{X_1, X_2, \dots, X_5\}$

- The **Apriori Principle** (downward closure property):
  - Any **subset** of a frequent itemset must be frequent.

    If {🍺, 🍼, 🍋} is frequent, so is {🍺 , 🍼}
  - Any **superset** of an infrequent itemset is infrequent.

    If { 🍺 , 🍼 } is NOT frequent, neither {🍺 , 🍼, 🍋}

Frequent Itemset Mining

- Apply **Apriori Algorithm** to find frequent itemsets:
  - Scan the database once to get **frequent 1-itemset**.
  - Generate **$(k+1)$ candidate itemsets** from **k frequent itemsets**.
  - Check the support of all candidate itemsets.
    - If not frequent, drop them.
  - Terminate when no frequent or candidate set can be generated.
    - Any superset of an infrequent itemset is infrequent.

The Apriori Algorithm - Example

TID	Items
1	🍺 🍼 🍉
2	🍺 🍭 🍼 🍋
3	🍼 🍺 🍋
4	🍼 🍭
5	🍷 🍺 🍋

min_sup = 2/5

first scan of DB

1-itemsets

Itemsets	Count
{🍺}	4
{🍼}	4
{🍉}	1
{🍭}	2
{🍋}	3
{🍷}	1

Frequent 1-itemsets

Itemsets	Count
{🍺}	4
{🍼}	4
{🍭}	2
{🍋}	3

The Apriori Algorithm - Example

TID	Items
1	🍺 🍼 🍉
2	🍺 🍭 🍼 🍋
3	🍼 🍺 🍋
4	🍼 🍭
5	🍷 🍺 🍋

min_sup = 2/5

Frequent 1-itemsets

Itemsets	Count
{🍺}	4
{🍼}	4
{🍭}	2
{🍋}	3

Candidate generation (self-join)

Candidate 2-itemsets

Itemsets
{🍺, 🍼}
{🍺, 🍭}
{🍺, 🍋}
{🍼, 🍭}
{🍼, 🍋}
{🍭, 🍋}

2nd scan of DB

Itemsets	Count
{🍺, 🍼}	3
{🍺, 🍭}	1
{🍺, 🍋}	3
{🍼, 🍭}	2
{🍼, 🍋}	2
{🍭, 🍋}	1

The Apriori Algorithm - Example

TID	Items
1	🍺 🍼 🍉
2	🍺 🍭 🍼 🍋
3	🍼 🍺 🍋
4	🍼 🍭
5	🍷 🍺 🍋

min_sup = 2/5

Frequent 2-itemsets

Itemsets	Count
{🍺, 🍼}	3
{🍺, 🍋}	3
{🍼, 🍭}	2
{🍼, 🍋}	2

Candidate generation (self-join)

Candidate 3-itemsets

Itemsets
{🍺, 🍼, 🍋}
{🍺, 🍼, 🍭}
{🍼, 🍭, 🍋}

3rd scan of DB

Itemsets	Count
{🍺, 🍼, 🍋}	2
{🍺, 🍼, 🍭}	1
{🍼, 🍭, 🍋}	1

The Apriori Algorithm - Example

TID	Items
1	🍺 🍼 🍉
2	🍺 🍭 🍼 🍋
3	🍼 🍺 🍋
4	🍼 🍭
5	🍷 🍺 🍋

min_sup = 2/5

Frequent
1-itemsets

Itemsets	Count
{🍺}	4
{🍼}	4
{🍭}	2
{🍋}	3

Frequent
2-itemsets

Itemsets	Count
{🍺, 🍼}	3
{🍺, 🍋}	3
{🍼, 🍭}	2
{🍼, 🍋}	2

Frequent
3-itemsets

Itemsets	Count
{🍺 , 🍼, 🍋}	2

Association Rules: From Support to Confidence

- Find ***X→Y*** that has both high **support** and high **confidence**.
  - X → Y: [support, confidence]

  - **Support**: joint probability that X and Y appear together in a transaction.

$$Support(X \wedge Y) \qquad P(X \cap Y)$$

  - **Confidence**: conditional probability that a transaction which contains X also contains Y.

$$Confidence(X \rightarrow Y) = \frac{Support(X \wedge Y)}{Support(X)} \qquad P(Y \mid X) = \frac{P(X \cap Y)}{P(X)}$$

Association Rules - Example

TID	Items Bought
1	🍺 🍼 🍉
2	🍺 🍭 🍼 🍋
3	🍼 🍺 🍋
4	🍼 🍭
5	🍷 🍪 🍺 🍞 🍋

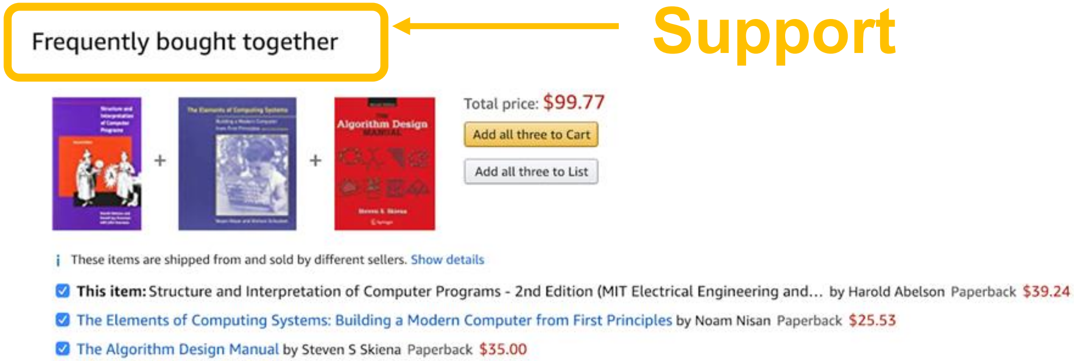
$\boxed{\{🍺\} \rightarrow \{🍼\}:}$ Association Rules

support = 60%
confidence = 75%
[60%, 75%]

$\boxed{\{🍺, 🍼\} \rightarrow \{🍋\}:}$ Association Rules

support = 40%
confidence = 66.7%
[40%, 66.7%]

Association Rules for Recommendation



Support

Confidence

Association Rules for Classification

- Y = {spam} (a 1-itemset)
- X = {a URL, an image} (a 2-itemset)

- X → Y:
  - **Support**: 1% of emails are spams (Y) with this URL & image (X).

$$P(X \cap Y) = 1\%$$

  - **Confidence**: 90% of emails with this URL & image (X) are spams (Y).

$$P(Y \mid X) = 90\%$$

  - **Conclusion**: classify an email as spam (Y) if it contains X.

Question: how is the confidence $P(Y \mid X)$ different from the conditional probability $P(c_i|E)$ estimated by a Naive Bayes Classifier?

Problem with Confidence

	Games	¬ Games	Sum (row)
Videos	4,000	3,500	7,500
¬ Videos	2,000	500	2,500
Sum (col.)	6,000	4,000	10,000

- A customer has bought computer games. Will he buy videos?
  - Customers who buy “computer games” → buy “videos”
    - [40%, 66.7%]

    The prior probability of buying videos is 75%.
  - Customers who buy “computer games” → **NOT** buy “videos”
    - [20%, 33.3%]

    The prior probability of NOT buying videos is 25%.
- **Confidence** cannot tell whether an association is coincidental.
  - What is relationship between computer game and video buying?

Recap: Evidence Lift

- Take purchase of computer games as evidence (e_1) and video buying as a target value (c_1), what is the lift for evidence e_1 ?

$$lift_{c_1}(e_i) = \frac{p(e_i|c_1)}{p(e_i)} \qquad \text{or} \qquad lift_{video}(games) = \frac{p(games|video)}{p(games)}$$

- Replace evidence (e_1) with X and classification result (c_1) with Y:

Observed joint probability of X & Y

$$\boldsymbol{Lift(X \rightarrow Y) = \frac{P(X|Y)}{P(X)} = \frac{P(X \cap Y)}{P(X) * P(Y)} = \frac{support\ (X \cap Y)}{support(X) * support(Y)}}$$

Expected joint probability of X & Y
(if they are independent)

Lift

- **Lift** measures how many times more often X and Y occur together than their expected frequency when they are independent.
- Lift is ranged between [0, ∞]:
  - Lift = 1: X and Y are statistically independent of each other.
  - Lift > 1: positive association between X and Y,
  - Lift < 1: negative association between X and Y.

$$Lift(X \rightarrow Y) = \frac{P(X|Y)}{P(X)} = \frac{P(X \cap Y)}{P(X) * P(Y)} = \frac{support\ (X \cap Y)}{support(X) * support(Y)}$$

	Games	¬ Games	Sum (row)
Videos	4,000	3,500	7,500
¬ Videos	2,000	500	2,500
Sum (col.)	6,000	4,000	10,000

$$\text{lift}(\text{Games}, \text{Videos}) = \frac{4000/10000}{6000/10000 * 7500/10000} = 0.89$$

$$\text{lift}(\text{Games}, \neg\, \text{Videos}) = \frac{2000/10000}{6000/10000 * 2500/10000} = 1.33$$

Leverage

- **Leverage** measures the difference between the observed joint probability of X and Y, and the expected (joint probability) if they are independent.

$$Leverage(X \rightarrow Y) = Support(X \wedge Y) - Support(X) * Support(Y)$$

$$\text{Leverage}(X \rightarrow Y) = P(X \cap Y) - P(X) * P(Y)$$

- Leverage is ranged between [-1,1]:
  - Leverage = 0: X and Y are independent of each other.
  - Leverage > 0: positive association between X and Y.
  - Leverage < 0: negative association between X and Y.
    - The larger (absolute) leverage is, the stronger the association is.

Measures such as ***lift*** and ***leverage*** not only ensure interesting rules are discovered, but also filter out coincidental rules.

Similarity of Itemsets

- Compare T1 or T3, which one is more similar to T2?
  - Consider each transaction as an itemset.
- Intuition:
  - Two sets are similar if they share a lot of items in common.
  - But larger sets are likely to share more items with others.

TID	Items Bought
T1	🍺 🍼 🍉
T2	🍺 🍭 🍼 🍋
T3	🍼 🍺 🍋
T4	🍼 🍭
T5	🍷 🍪 🍺 🍞 🍋

Jaccard Similarity

- **Jaccard Similarity** measures the similarity of two itemsets.
  - Also known as Jaccard coefficient or Jaccard index.

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|}$$

Jaccard Distance: 1 - J(A, B)

  - **A ∩ B** (Intersection): largest common subset of A and B

{🍺, 🍼, 🍉} ∩ {🍼, 🍺, 🍋} = {🍺, 🍼}

  - **A ∪ B** (Union): smallest common superset of A and B

{🍺, 🍼, 🍉} ∪ {🍼, 🍺, 🍋} = {🍺, 🍼, 🍉, 🍋}

- Jaccard similarity is ranged [0,1].
  - J(A, B) = 0 if two sets share no items in common.
  - J(A, B) = 1 if two sets are identical.

Similarity of Itemsets

TID	Items Bought
T1	🍺 🍼 🍉
T2	🍺 🍭 🍼 🍋
T3	🍼 🍺 🍋
T4	🍼 🍭
T5	🍷 🍪 🍺 🍞 🍋

$$J(T2,T1) = \frac{|\{\text{🍺},\text{🍼}\}|}{|\{\text{🍺},\text{🍼},\text{🍉},\text{🍭},\text{🍋}\}|} = \frac{2}{5} = 0.4$$

$$J(T2,T3) = \frac{|\{\text{🍺},\text{🍼},\text{🍋}\}|}{|\{\text{🍺},\text{🍼},\text{🍭},\text{🍋}\}|} = \frac{3}{4} = 0.75$$

Can you calculate J(T2, T4)?

Similarity of Items

TID	Items Bought
T1	🍺 🍼 🍉
T2	🍺 🍭 🍼 🍋
T3	🍼 🍺 🍋
T4	🍼 🍭
T5	🍷 🍺 🍋

transpose

Item	Transactions
🍺	T1, T2, T3, T5
🍼	T1, T2, T3, T4
🍉	T1
🍭	T2, T4
🍋	T2, T3, T5
🍷	T5

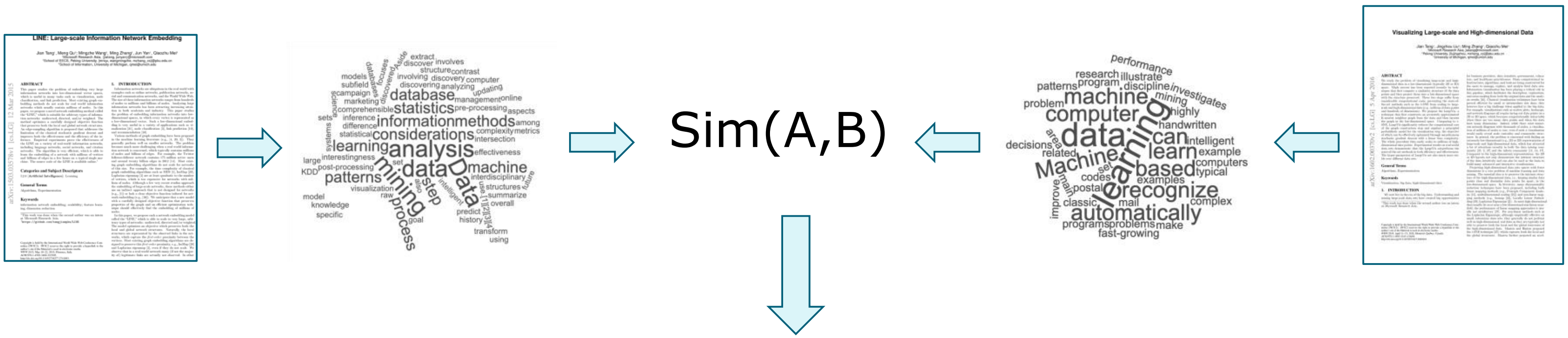
$$J(\text{🍺}, \text{🍼}) = \frac{|\{T1,T2,T3\}|}{|\{T1,T2,T3,T4,T5\}|} = \frac{3}{5} = 0.6$$

$$J(\text{🍺}, \text{🍋}) = \frac{|\{T2,T3,T5\}|}{|\{T1,T2,T3,T5\}|} = \frac{3}{4} = 0.75$$

Question:

How is this measure different from the support of a 2-itemset?

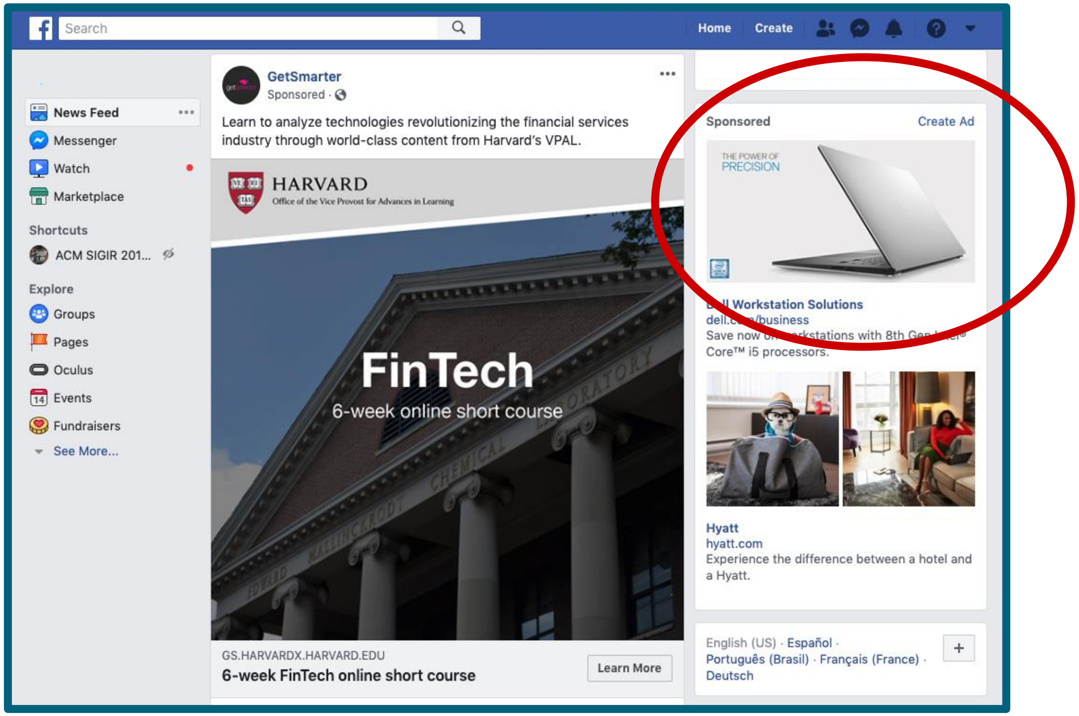
Application of Itemset similarity



- Classification
- Clustering
- Ranking
- Recommendation

Limitation of Itemset

- **Itemset** representation ignores order and quantity.
- **Vector** representation only handles quantity.
- But what about **order**? Order matters in reality...



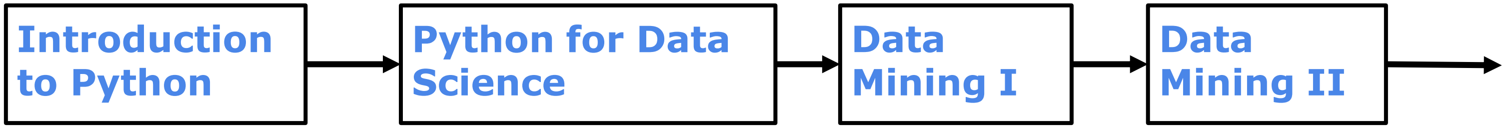
Viewed this model of laptop on *dell.com* as displayed in this Facebook ad.

Ended up purchasing a different model.

Having already purchased another model, is this ad relevant anymore?

From Itemset to Sequence

- **Sequence**: **categorical items** organized in a sequential **order**



- X_k is the categorical item that appears in k^{th} position of the sequence X.
  - Order matters.
  - Repeating items matter.
  - Absolute position does NOT matter.

$$X = \{(x_1, 1), (x_2, 2), ..., (x_k, k)\}$$

$$X : x_1 \to x_2 \to ... \to x_k$$

$$X : x_1 x_2 ... x_k$$

Frequent Sequential Patterns

- **Sequence**: categorical items + order
- **Frequent sequential patterns**: frequent itemsets + order
  - Order matters.
  - Repeating items matter.
  - Absolute position does NOT matter.

From Itemsets to Sequences

TID	Sequences
1	🍺 🍼 🍉
2	🍺 🍭 🍼 🍋
3	🍼 🍺 🍋
4	🍼 🍭
5	🍷 🍪 🍺 🍞 🍋

{🍺}: support = 80% ✓

{🍺, 🍼}: support = 40%

{🍺, 🍼, 🍋}: support = 20%

If *min_sup* = 50%, then only
{🍺} is frequent

From Itemsets to Sequences

- The **apriori algorithm** also works on sequences.
  - If a sequence is frequent, all its **sub-sequences** are frequent.
  - If a sequence is NOT frequent, no need to check its **super-sequence**.

- **Association rules** still apply, but they are order sensitive.
  - Evaluation metrics still apply, but it is order sensitive.

Association Rules Without Order

TID	Items Bought
1	🍺 🍼 🍉
2	🍺 🍭 🍼 🍋
3	🍼 🍺 🍋
4	🍼 🍭
5	🍷 🍪 🍺 🍞 🍋

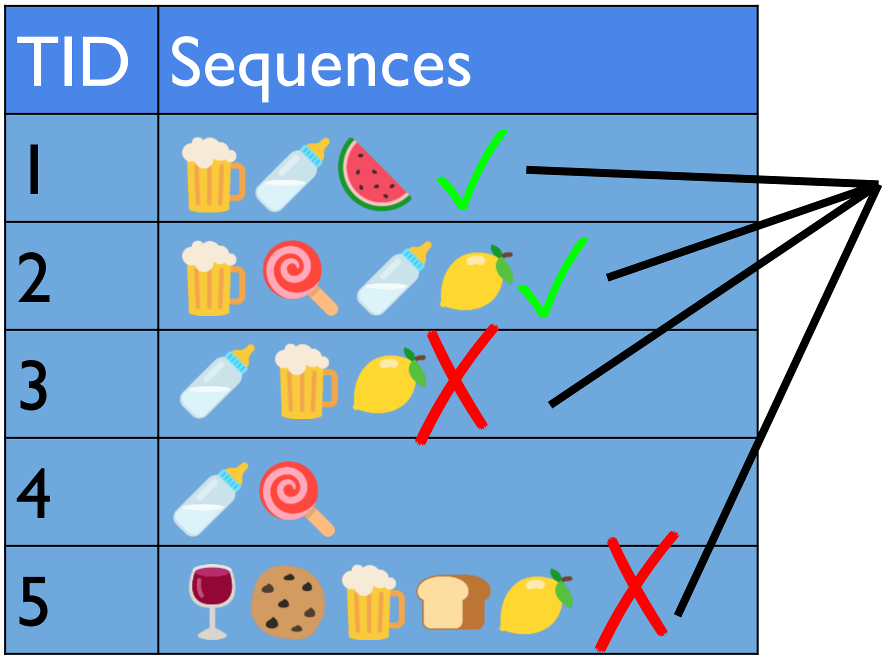
{🍺} → {🍼}:

support = 60%

confidence = 75%

[60%, 75%]

Association Rules With Order



{🍺} → {🍼}:

support = 40%

confidence = 50%

[40%, 50%]

Similarity of Sequences

- First try: apply vector or itemset similarity measures
  - Order matters for sequence data
    - e.g., “live” vs. “evil”
  - Both itemset and vectors failed to handle order

- Second try: **Hamming Distance**
  - Equal-length sequences: number of positions at which the corresponding items are different
    - e.g., distance between “Carolyn” and “Karolin” is 2

      (differences in the same positions: the 1st and 6th)

What about “awake” and “waked”?

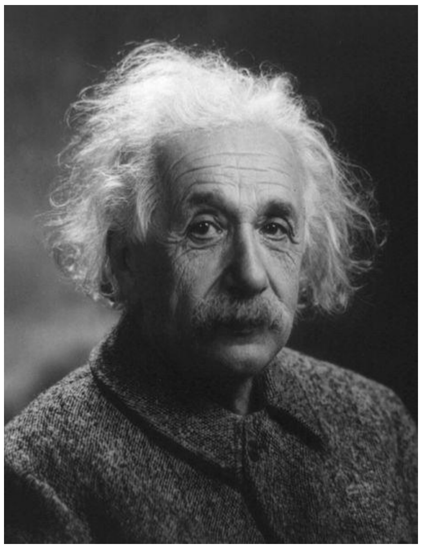
Similarity of Sequences

- Intuition:
  - The more items in common, the more similar;
  - The more aligned the order, the more similar.

- **Edit distance**:
  - the **minimum number of edit operation** (e.g., insertion, deletion, or substitution) required to convert one sequence into the other.

Which one is more like “computer”?
“compute” or “counter” ?

Application: Spelling Correction



albert einstein	4834
albert einstien	525
albert einstine	149
albert einsten	27
albert einsteins	25
albert einstain	11
albert einstin	10
albert eintein	9
albeart einstein	6
aolbert einstein	6
alber einstein	4
albert einseint	3
albert einsteirn	3
albert einsterin	3
albert eintien	3
alberto einstein	3
albrecht einstein	3
alvert einstein	3

Table 1. Counts of different (mis)spellings of Albert Einstein's name in a web query log.

Cucerzan and Brill, EMNLP 2004

Spelling Correction Task

- Input misspelled words "alber einstien"
- Output strings that are:
  - correct (in dictionary or frequently used); and
  - similar enough to the input

"Did you mean: *Albert Einstein*?"