Computational Tools for Data Science

Week 5:

Frequent itemsets

Motivation: Market-Basket Analysis

- **Setting**: Supermarket
- **Task:** Analyse shopping baskets of customers
- **Goal:** Find sets of items that appear often, i.e. in many baskets
- **Why:**
  - See demand, hence we can adjust the price.
  - Find patterns (*association rules*) in the data and make more tricky sales.

Motivation: Market-Basket Analysis

Prime example:

Via data analysis: **{Diapers, Beer}** is a **frequent** pair of items

Not expected / interesting

Explanation: new parents rarely go into bars, they drink at home.

Sales trick: make a sale on diapers, but secretly increase beer price

Motivation: Market-Basket Analysis

Other examples:

- Baskets correspond to patients's infos (**biomarkers, diseases**)

  → Frequent items with a common disease suggest testing.

- Baskets are sets of **sentences**, items are (text) **documents**.

Note: Items need not be \`in´ baskets!

Items and baskets are in **some** many-to-many relation.

Here: item in basket if the document contains the sentence.

Frequent pair → 2 documents that share many sentences (**plagiarism**)

Motivation: Market-Basket Analysis

In general:

- there are many (!) items
- there are also many (!) baskets

- baskets correspond to sets of items
- size of a basket is small compared to the number of all items

- our relevant data: sequence / list of baskets
  - » problem: often too big to fit into main memory

Motivation: Market-Basket Analysis

In general:

Often only interested in frequent pairs, or triples.

Given n items, there are $\binom{n}{2} = \frac{n(n-1)}{2} \approx \frac{n^2}{2}$ pairs of items.

Given n items, there are $\binom{n}{k} = \frac{n!}{(n-k)! \cdot k!} \approx \frac{n^k}{k!}$ k-element itemsets.

Practice:

- Too much data for considering k-element itemsets for big k.
- Sales offers: can only handle few frequent pairs.
  - » There will be even fewer frequent triples, etc.

Measures: (Support)

Let $\boldsymbol{I}$ be a **set of items**.

$$Support(I) = \frac{\#\, baskets\ containing\ I}{total\ number\ of\ baskets}$$

Meaning: Probability of finding all items of I in one basket.

Definition: We call an itemset $\boldsymbol{I}$ **frequent**, if its support is greater than some initially fixed value $\boldsymbol{s} \in \mathbb{R}$.

$$Support(I) \geq s$$

Measures for association rules: (Confidence)

- Let $\boldsymbol{I}$ be a **set of items**
- Let $\boldsymbol{j}$ be an item that is not in $\boldsymbol{I}$.
- Let $\boldsymbol{I} \rightarrow \boldsymbol{j}$ be an **association rule**.

$$Confidence(I \rightarrow j) = \frac{\#\, baskets\ containing\ I \cup \{j\}}{\#\, baskets\ containing\ I}$$

Meaning: Probability of having all of $I \cup \{j\}$ in one basket, given that we already have all of I in the basket.

Motivation: Market-Basket Analysis

Example:

Basket #	Items in the baskets
1	{Butter, Bread, Milk, Cola, Cheese, Toothbrush}
2	{Butter, Bread, Milk, Cola}
3	{Bread, Milk, Toothbrush}
4	{Butter, Bread, Onion}
5	{Flour, Milk, Cola, Cheese}
6	{Flour, Butter, Milk, Cola}
7	{Flour, Milk, Toothbrush}
8	{Flour, Onion, Toothbrush}

Basket #	Items in the baskets
1	{**Butter**, Bread, Milk, Cola, Cheese, Toothbrush}
2	{**Butter**, Bread, Milk, Cola}
3	{Bread, Milk, Toothbrush}
4	{**Butter**, Bread, Onion}
5	{Flour, Milk, Cola, Cheese}
6	{Flour, **Butter**, Milk, Cola}
7	{Flour, Milk, Toothbrush}
8	{Flour, Onion, Toothbrush}

$$Support(\{Butter\}) = 0{,}5$$

Basket #	Items in the baskets
1	{Butter, **Bread**, Milk, Cola, Cheese, Toothbrush}
2	{Butter, **Bread**, Milk, Cola}
3	{**Bread**, Milk, Toothbrush}
4	{Butter, **Bread**, Onion}
5	{Flour, Milk, Cola, Cheese}
6	{Flour, Butter, Milk, Cola}
7	{Flour, Milk, Toothbrush}
8	{Flour, Onion, Toothbrush}

$$Support(\{Butter\}) = 0{,}5$$
$$Support(\{Bread\}) = 0{,}5$$

Basket #	Items in the baskets
1	{Butter, Bread, **Milk**, Cola, Cheese, Toothbrush}
2	{Butter, Bread, **Milk**, Cola}
3	{Bread, **Milk**, Toothbrush}
4	{Butter, Bread, Onion}
5	{Flour, **Milk**, Cola, Cheese}
6	{Flour, Butter, **Milk**, Cola}
7	{Flour, **Milk**, Toothbrush}
8	{Flour, Onion, Toothbrush}

$$Support(\{Butter\}) = 0{,}5$$
$$Support(\{Bread\}) \ = 0{,}5$$
$$Support(\{Milk\}) = 0{,}75$$

Basket #	Items in the baskets
1	{Butter, Bread, Milk, Cola, Cheese, **Toothbrush**}
2	{Butter, Bread, Milk, Cola}
3	{Bread, Milk, **Toothbrush**}
4	{Butter, Bread, Onion}
5	{Flour, Milk, Cola, Cheese}
6	{Flour, Butter, Milk, Cola}
7	{Flour, Milk, **Toothbrush**}
8	{Flour, Onion, **Toothbrush**}

$$Support(\{Butter\}) = 0{,}5$$
$$Support(\{Bread\}) \ = 0{,}5$$
$$Support(\{Milk\}) = 0{,}75$$
$$Support(\{Toothbrush\}) = 0{,}5$$

Basket #	Items in the baskets
1	{Butter, Bread, Milk, Cola, **Cheese**, Toothbrush}
2	{Butter, Bread, Milk, Cola}
3	{Bread, Milk, Toothbrush}
4	{Butter, Bread, Onion}
5	{Flour, Milk, Cola, **Cheese**}
6	{Flour, Butter, Milk, Cola}
7	{Flour, Milk, Toothbrush}
8	{Flour, Onion, Toothbrush}

$$Support(\{Butter\}) = 0{,}5$$
$$Support(\{Bread\}) = 0{,}5$$
$$Support(\{Milk\}) = 0{,}75$$
$$Support(\{Toothbrush\}) = 0{,}5$$
$$Support(\{Cheese\}) = 0{,}25$$

Basket #	Items in the baskets
1	{**Butter**, **Bread**, Milk, Cola, Cheese, Toothbrush}
2	{**Butter**, **Bread**, Milk, Cola}
3	{Bread, Milk, Toothbrush}
4	{**Butter**, **Bread**, Onion}
5	{Flour, Milk, Cola, Cheese}
6	{Flour, Butter, Milk, Cola}
7	{Flour, Milk, Toothbrush}
8	{Flour, Onion, Toothbrush}

$$Support(\{Butter\}) = 0{,}5$$
$$Support(\{Bread\}) = 0{,}5$$
$$Support(\{Milk\}) = 0{,}75$$
$$Support(\{Toothbrush\}) = 0{,}5$$
$$Support(\{Cheese\}) = 0{,}25$$

$$Support(\{Butter, Bread\}) = 0{,}375$$

Basket #	Items in the baskets
1	{Butter, Bread, **Milk**, Cola, **Cheese**, Toothbrush}
2	{Butter, Bread, Milk, Cola}
3	{Bread, Milk, Toothbrush}
4	{Butter, Bread, Onion}
5	{Flour, **Milk**, Cola, **Cheese**}
6	{Flour, Butter, Milk, Cola}
7	{Flour, Milk, Toothbrush}
8	{Flour, Onion, Toothbrush}

$$Support(\{Butter\}) = 0{,}5$$
$$Support(\{Bread\}) = 0{,}5$$
$$Support(\{Milk\}) = 0{,}75$$
$$Support(\{Toothbrush\}) = 0{,}5$$
$$Support(\{Cheese\}) = 0{,}25$$

$$Support(\{Butter, Bread\}) = 0{,}375$$
$$Support(\{Milk, Cheese\}) = 0{,}25$$

Basket #	Items in the baskets
1	{Butter, Bread, **Milk**, Cola, Cheese, **Toothbrush**}
2	{Butter, Bread, Milk, Cola}
3	{Bread, **Milk**, **Toothbrush**}
4	{Butter, Bread, Onion}
5	{Flour, Milk, Cola, Cheese}
6	{Flour, Butter, Milk, Cola}
7	{Flour, **Milk**, **Toothbrush**}
8	{Flour, Onion, Toothbrush}

$$Support(\{Butter\}) = 0{,}5$$
$$Support(\{Bread\}) = 0{,}5$$
$$Support(\{Milk\}) = 0{,}75$$
$$Support(\{Toothbrush\}) = 0{,}5$$
$$Support(\{Cheese\}) = 0{,}25$$

$$Support(\{Butter, Bread\}) = 0{,}375$$
$$Support(\{Milk, Cheese\}) = 0{,}25$$
$$Support(\{Milk, Toothbrush\}) = 0{,}375$$

Basket #	Items in the baskets
1	{**Butter**, **Bread**, Milk, Cola, Cheese, Toothbrush}
2	{**Butter**, **Bread**, Milk, Cola}
3	{Bread, Milk, Toothbrush}
4	{**Butter**, **Bread**, Onion}
5	{Flour, Milk, Cola, Cheese}
6	{Flour, **Butter**, Milk, Cola}
7	{Flour, Milk, Toothbrush}
8	{Flour, Onion, Toothbrush}

$$Confidence(\{Butter\} \rightarrow Bread\}) = 0{,}75$$

Basket #	Items in the baskets
1	{Butter, Bread, **Milk**, Cola, **Cheese**, Toothbrush}
2	{Butter, Bread, Milk, Cola}
3	{Bread, Milk, Toothbrush}
4	{Butter, Bread, Onion}
5	{Flour, **Milk**, Cola, **Cheese**}
6	{Flour, Butter, Milk, Cola}
7	{Flour, Milk, Toothbrush}
8	{Flour, Onion, Toothbrush}

$$Confidence(\{Butter\} \rightarrow Bread\}) = 0{,}75$$
$$Confidence(\{Cheese\} \rightarrow Milk\}) = 1$$

Basket #	Items in the baskets
1	{Butter, Bread, **Milk**, Cola, Cheese, **Toothbrush**}
2	{Butter, Bread, Milk, Cola}
3	{Bread, **Milk**, **Toothbrush**}
4	{Butter, Bread, Onion}
5	{Flour, Milk, Cola, Cheese}
6	{Flour, Butter, Milk, Cola}
7	{Flour, **Milk**, **Toothbrush**}
8	{Flour, Onion, **Toothbrush**}

$$Confidence(\{Butter\} \rightarrow Bread\}) = 0{,}75$$
$$Confidence(\{Cheese\} \rightarrow Milk\}) = 1$$
$$Confidence(\{Toothbrush\} \rightarrow Milk\}) = 0{,}75$$

Measures for association rules: (Lift)

- Let $\boldsymbol{I}$ be a **set of items**
- Let $\boldsymbol{j}$ be an item that is not in $\boldsymbol{I}$.
- Let $\boldsymbol{I} \rightarrow \boldsymbol{j}$ be an **association rule**.

$$Lift(I \rightarrow j) = \frac{Confidence(I \rightarrow j)}{Support(\{j\})}$$

Meaning: Probability of having j in a basket given that we already have all of I in the basket divided by the probability of having j in a basket.

Measures for association rules: (Lift)

- Let $\boldsymbol{I}$ be a **set of items**
- Let $\boldsymbol{j}$ be an item that is not in $\boldsymbol{I}$.
- Let $\boldsymbol{I} \rightarrow \boldsymbol{j}$ be an **association rule**.

$$Lift(I \rightarrow j) = \frac{Confidence(I \rightarrow j)}{Support(\{j\})}$$

Note:

- If $Lift(I \rightarrow j)$ is close to 1, then “not really interesting”.
- If $Lift(I \rightarrow j) \gg 1$, then I encourages to buy j.
- If $Lift(I \rightarrow j) \ll 1$, then I discourages to buy j.

Measures for association rules: (Interest)

- Let $\boldsymbol{I}$ be a **set of items**
- Let $\boldsymbol{j}$ be an item that is not in $\boldsymbol{I}$.
- Let $\boldsymbol{I} \rightarrow \boldsymbol{j}$ be an **association rule**.

$$Interest(I \rightarrow j) = Confidence(I \rightarrow j) - Support(\{j\})$$

Note:

- If $Interest(I \rightarrow j)$ is close to 0, then “not really interesting”.
- If $Interest(I \rightarrow j)$ is highly positive, then I encourages to buy j.
- If $Interest(I \rightarrow j)$ is highly negative, then I discourages to buy j.

Basket #	Items in the baskets
1	{**Butter**, **Bread**, Milk, Cola, Cheese, Toothbrush}
2	{**Butter**, **Bread**, Milk, Cola}
3	{**Bread**, Milk, Toothbrush}
4	{**Butter**, **Bread**, Onion}
5	{Flour, Milk, Cola, Cheese}
6	{Flour, Butter, Milk, Cola}
7	{Flour, Milk, Toothbrush}
8	{Flour, Onion, Toothbrush}

$$Confidence(\{Butter\} \rightarrow Bread\}) = 0{,}75$$
$$Lift(\{Butter\} \rightarrow Bread\}) = 1{,}5$$

Basket #	Items in the baskets
1	{Butter, Bread, **Milk**, Cola, Cheese, **Toothbrush**}
2	{Butter, Bread, **Milk**, Cola}
3	{Bread, **Milk**, **Toothbrush**}
4	{Butter, Bread, Onion}
5	{Flour, **Milk**, Cola, Cheese}
6	{Flour, Butter, **Milk**, Cola}
7	{Flour, **Milk**, **Toothbrush**}
8	{Flour, Onion, Toothbrush}

$$Confidence(\{Toothbrush\} \rightarrow Milk\}) = 0{,}75$$
$$Lift(\{Toothbrush\} \rightarrow Milk\}) = 1$$

Finding frequent itemsets

Finding / determining interesting association rules is easy compared to finding frequent itemsets (what we have to do first anyway).

Challenges:

- Huge amount of data. → Not enough main memory.

Focus:

- How to find frequent itemsets (algorithms).
- How handle (main) memory.

Store itemset count in main memory

For each frequent itemset algorithm we have to store the count for the itemsets.

(!) Use integers (4 byte) instead of item names / long IDs.

→ via a **hash table**.

Store itemset count in main memory

For each frequent itemset algorithm we have to store the count for the itemsets.

Limit on how many items we can deal with without **thrashing**.

Example: Say we have n items and need to count all pairs of them.

We must save about $\frac{n^2}{2}$ integers, each 4 bytes. So $2n^2$ bytes.

If we have 2 GB main memory, say 2^{31} bytes, then $n \leq 2^{15} \approx 33000$.

Triangular-Matrix Method

Let n be the number of all items of which we want to count pairs.

- Use one-dimensional array a.
- $a[k]$ contains count of the k -th pair of items.
- The ordering of the item pairs is as follows:

$$\{1,2\},\{1,3\},\dots,\{1,n\},\{2,3\},\{2,4\},\dots,\{2,n\},\dots,\{n-1,n\}$$

Formally: $a[k]$ contains the count of the pair $\{i,j\}$, with $1 \leq i < j \leq n$,

where $k = (i-1)\left(n - \frac{i}{2}\right) + j - i$.

Triples Method

Let n be the number of all items of which we want to count pairs.
Let $1 \leq i < j \leq n$.

- Store counts as triples $[i, j, c]$ where c equals the count of $\{i, j\}$.

Advantage: No need to store pairs with count c equal to 0.

Disadvantage: For each pair we use 3 integers instead of just 1.

Hence: If fewer than $\frac{1}{3}$ of all possible pairs appear in some basket, then use the Triples Method.

Naïve algorithm for counting frequent itemsets

Say our main memory is gigantic and will not limit us.

1. Read the whole data (list of baskets) into main memory.

2a. Use 2 nested loops to generate all pairs of items.
(Use k nested loops to generate and count all k-element itemsets.)

2b. Use Triangular-Matrix Method or Triples Method to store counts.

3. Examine which pairs / k-tuples have count bigger than the support threshold s. → yields frequent itemsets

Computational model

Remember: In general our dataset is too big to fit into main memory.

The time for reading data into main memory dominates the time for generating / counting the relevant pairs or small k-element itemsets.

Hence, we measure the **cost of computation** by the **number of passes** an algorithm makes **over the data**.

Monotonicity of frequent itemsets

Observation: If an itemset is frequent, then so is every subset of it.

The A-Priori Algorithm utilises this simple observation.

Definition: We call a frequent itemset ***maximal*** if no proper superset of it is frequent.

Saves space later: Only store maximal frequent itemsets.

A-Priori Algorithm (for item pairs)

Uses 2 passes through the data, but requires **less main memory**.

Pass 1: Read data (list of baskets) and **only store** count of each individual item, i.e. of **singleton itemsets** $\{i\}$.

In main memory:

- Hash table (item names to integers in range 1 to n)
- Count for each item (stored as an array).

→ Required space proportional to the number of items.

A-Priori Algorithm (for item pairs)

Before pass 2:

Say we had found m frequent singleton itemsets after pass 1.
($m \approx \frac{n}{100}$ could be reasonable.)

Create new array (***frequent singletons table***), indexed 1 to n, where

- non-frequent items are mapped to 0, and
- each frequent item is mapped to a unique integer between 1 and m.

In main memory:
- Hash table (item names to integers in range 1 to n)
- Frequent singletons table

A-Priori Algorithm (for item pairs)

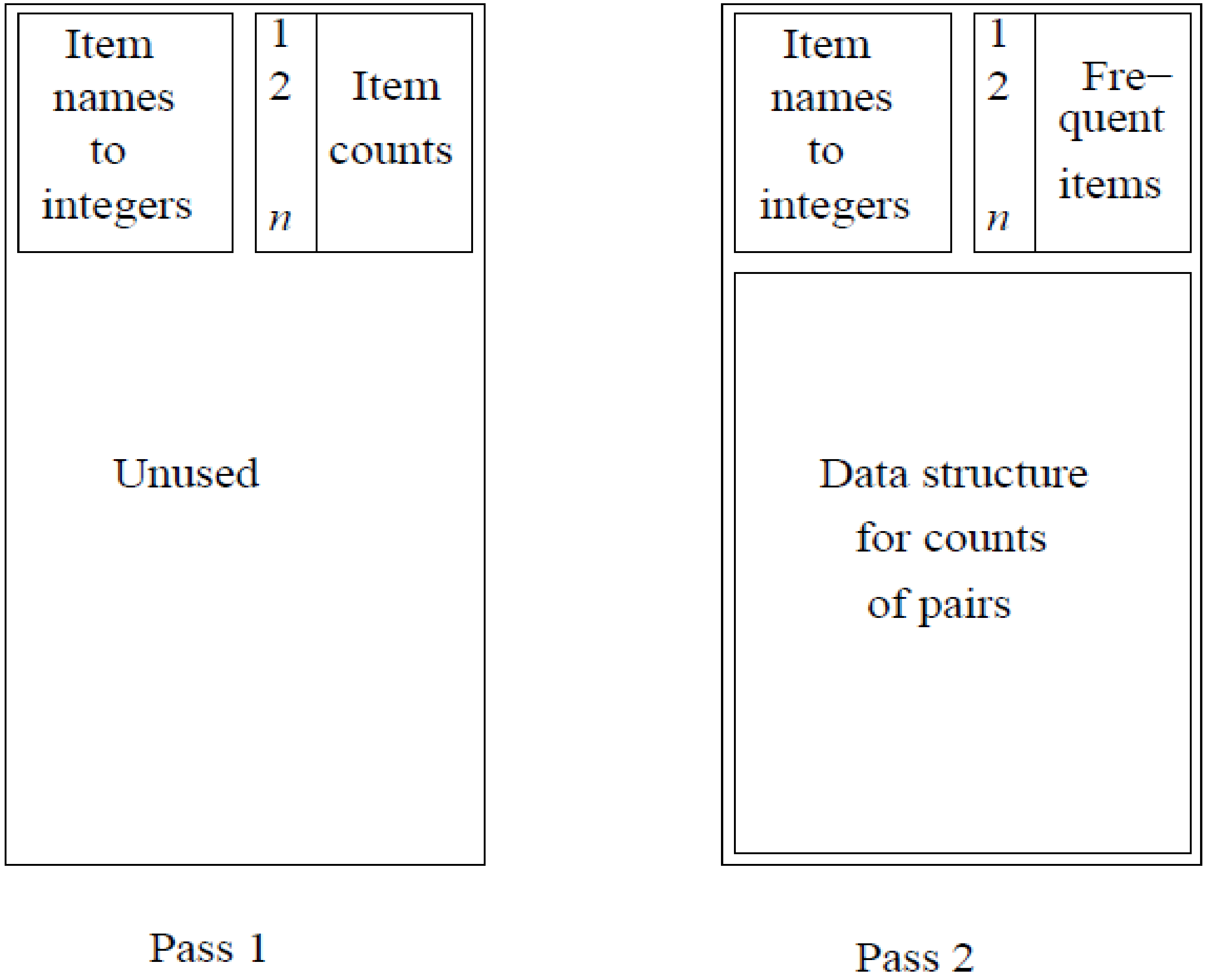
Pass 2: Read data again and **only store** pairs of items both of which are frequent (utilise frequent singletons table).

By monotonicity: we do not miss any pair of frequent items!

In main memory:

- Hash table (item names to integers)
- Frequent singletons table
- Count for pairs of items (via Triangular-Matrix or Triples Method)

A-Priori Algorithm (for item pairs)



picture: from MMDS, p. 219

A-Priori Algorithm (for k-element itemsets)

- Uses k passes through data.

- In pass i:

  generate set of candidates C_i for frequent i-element itemsets and store counts.

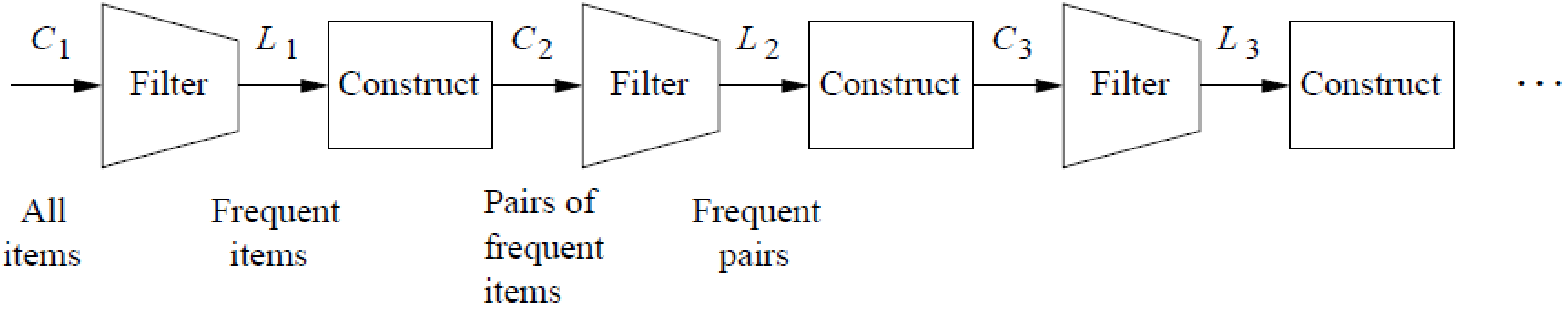
To do this, use:

- frequent $(i-1)$-element itemset table that lists L_{i-1}, the frequent $(i-1)$-element itemsets, and
- frequent singletons table L_1.

- Between pass i and $i+1$:

  create frequent i-element itemset table.

A-Priori Algorithm (for k-element itemsets)



picture: from MMDS, p. 220

PCY-Algorithm (named after Park, Chen and Yu)

- A-Priori works fine unless generating C_2 causes thrashing.

- Rough idea:
  - Utilise unused main memory in pass 1 for additional hash table.
    - » Further reduce number of item pairs that are created.

- Example / Note:

Say we have 1.000.000 items and several GB of main memory.
Then we need not even 10% of main memory for:

  - hash table (item names to integers)
  - array for counting (singleton) item frequency

PCY-Algorithm (for pairs)

- Fill left over main memory with a hash table (integer array).
- The buckets correspond to integers (counts).
- Use as many buckets as possible for the table.
- Hash pairs of items to buckets.

During pass 1 of A-Priori:

- Also initialise hash table / array with all zeros.
- Beside counting all singleton itemsets, **also** generate all pairs, **BUT** instead of storing them: hash them and increase count of bucket to which they hash by 1.

PCY-Algorithm (for pairs)

Idea to improve the second pass:

If a bucket is not frequent, then clearly no pair in it can be frequent.

Hence, define candidate set C_2 of pairs $\{i,\ j\}$ via:

(a) $\{i\}$ and $\{j\}$ are both frequent singleton itemsets

(b) $\{i,\ j\}$ hashes to a frequent bucket, i.e. whose count is bigger than our threshold s.

PCY-Algorithm (for pairs)

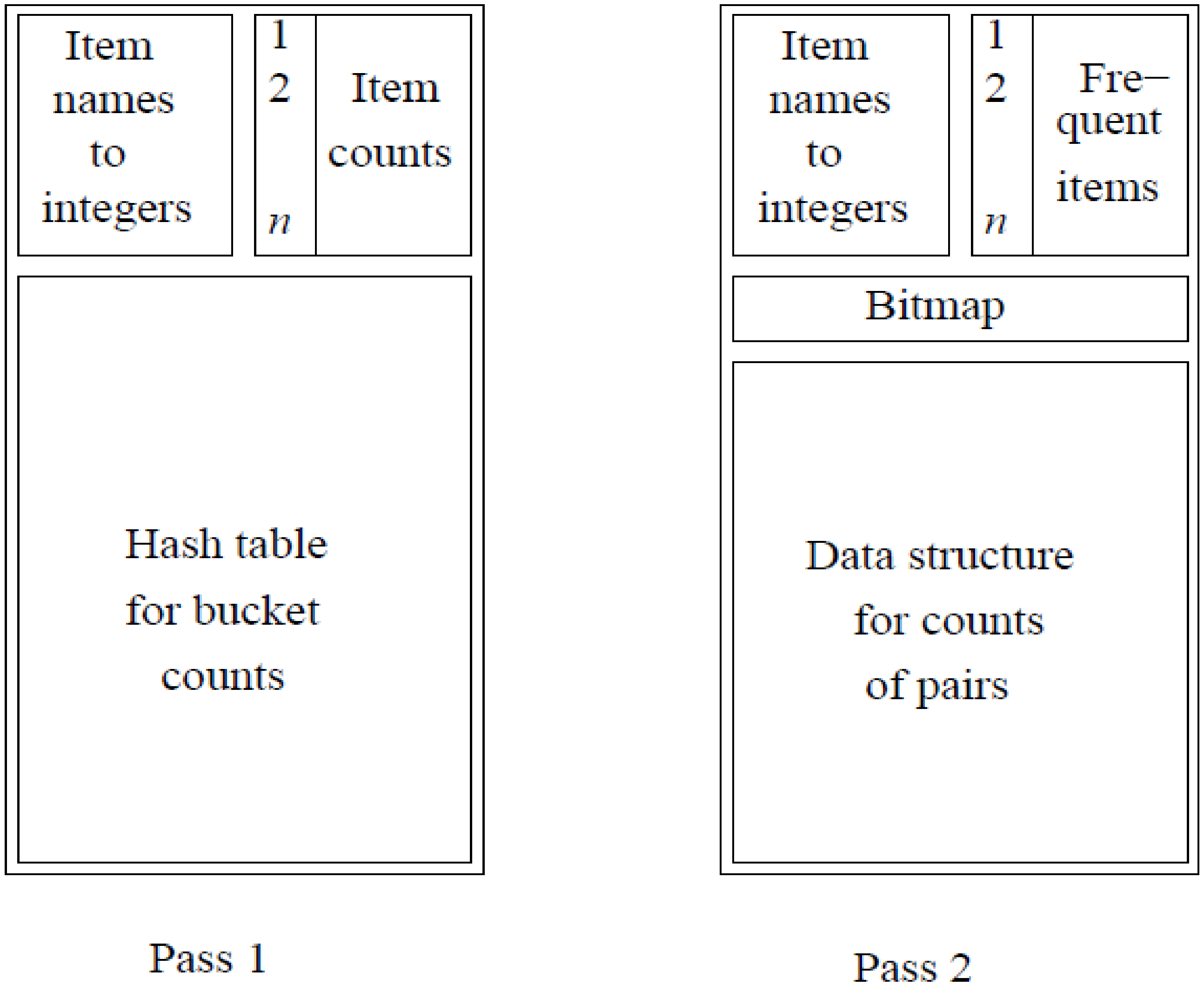
Between pass 1 and pass 2:

Summarize hash table (integer counts) via bitmap:

- » 1 if bucket is frequent
- » 0 if bucket is not frequent.

Hence 4 byte = 32 bits integer counts are replaced by 1 bit each.

PCY-Algorithm (for pairs)



picture: from MMDS, p. 223

PCY-Algorithm (for pairs)

Advantages and disadvantages:

(+) We save memory if we have (many) buckets that are infrequent.

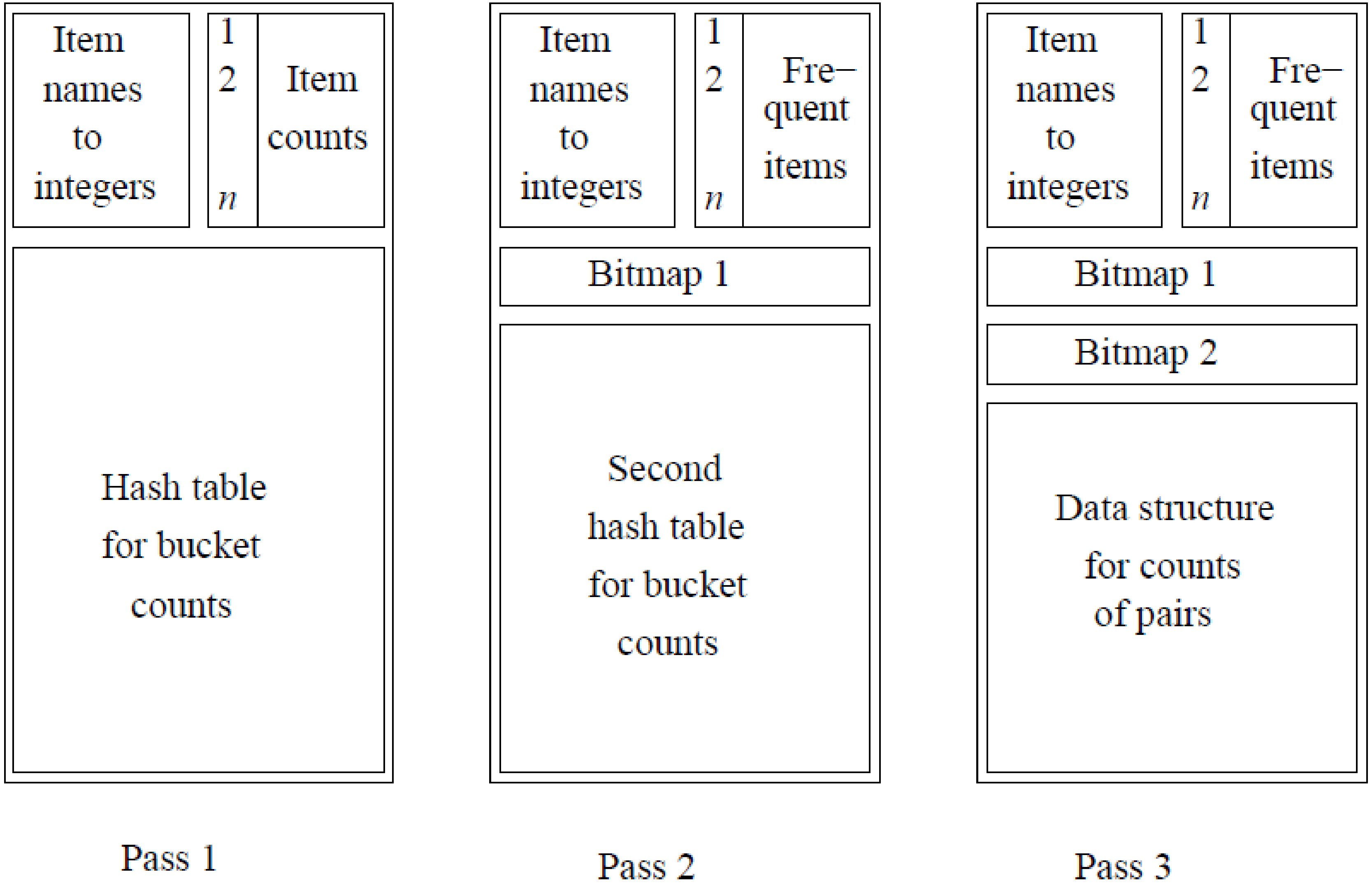
(-) We are forced to store counts for pairs via Triples-Method.

→ In A-Priori: renumbered frequent singleton itemsets (form 1 to m instead to n) to save space.

→ Now pairs in infrequent buckets could be rather randomly distributed within the lexicographic order.

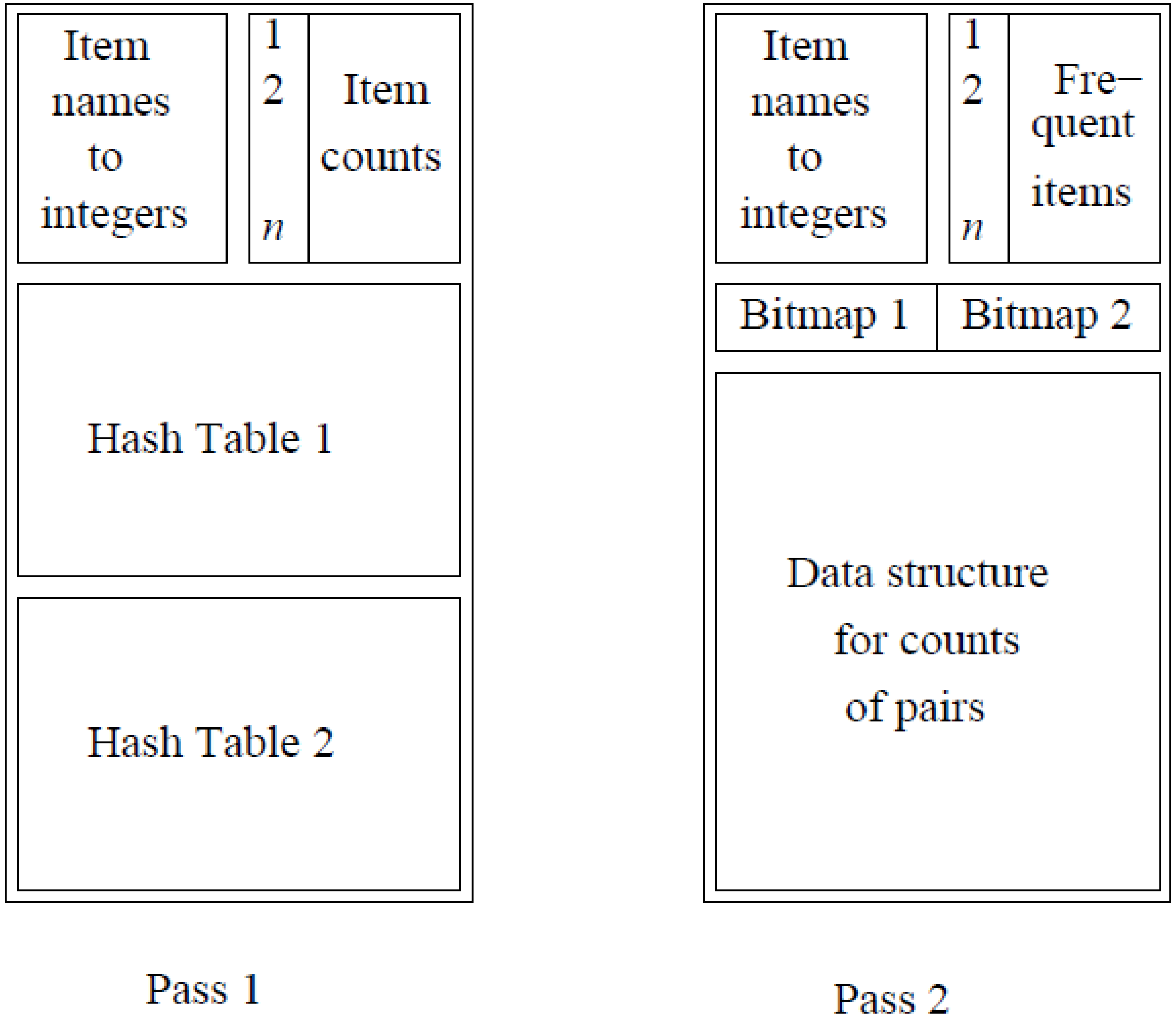
Hence: PCY only saves memory compared to A-Priori if $2/3$ of all candidate pairs can be discarded.

Refinement of PCY: The Multistage Algorithm



picture: from MMDS, p. 225

Refinement of PCY: The Multihash Algorithm



picture: from MMDS, p. 227