



Big Data Analytics

– Week 5: Frequent Pattern Tree –

Knowledge Discovery and Data Mining
Prof. Dr. Emmanuel Müller

Mining Frequent Patterns Without Candidate Generation

Compress a large database into a compact,
Frequent-Pattern tree (FP-tree) structure

- highly condensed, but complete for frequent pattern mining
- avoid costly database scans

Develop an efficient, FP-tree-based frequent pattern mining method

- A divide-and-conquer methodology: decompose mining tasks into smaller ones
- Avoid candidate generation: sub-database test only!

Idea:

- Compress database into FP-tree, retaining the itemset association information
- Divide the compressed database into conditional databases, each associated with one frequent item and mine each such database separately.

Construct FP-tree from a Transaction DB (1)

Steps for compressing the database into a FP-tree:

1. Scan DB once, find frequent 1-itemsets (single items)
2. Order frequent items in frequency descending order

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

header table:

item	frequency
f	4
c	4
a	3
b	3
m	3
p	3

*sort items in the order of
descending support*

minSup=0.5

Construct FP-tree (2)

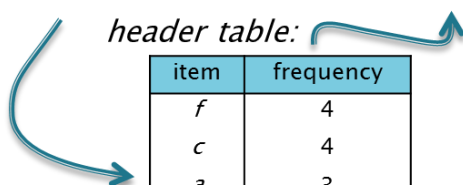
Steps for compressing the database into a FP-tree:

1. Scan DB once, find frequent 1-itemsets (single items)
2. Order frequent items in frequency descending order
3. Scan DB again, construct FP-tree **starting with most frequent item per transaction**

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

for each transaction only keep its frequent items sorted in descending order of their frequencies

header table:



item	frequency
f	4
c	4
a	3
b	3
m	3
p	3

- for each transaction built a path in the FP-tree:
- If a path with common prefix exists:
increment frequency of nodes on this path and append suffix
 - Otherwise: create a new branch

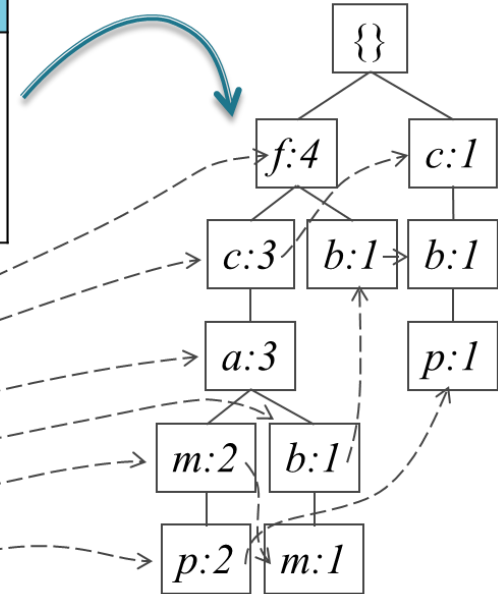
Construct FP-tree (3)

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

header table:

item	frequency	head
f	4	•
c	4	•
a	3	•
b	3	•
m	3	•
p	3	•

header table
references the
occurrences of the
frequent items in the
FP-tree



Benefits of the FP-tree Structure

- Completeness:
 - never breaks a long pattern of any transaction
 - preserves complete information for frequent pattern mining
- Compactness
 - reduce irrelevant information—infrequent items are gone
 - frequency descending ordering: more frequent items are more likely to be shared
 - never be larger than the original database (if not count node-links and counts)
 - Experiments demonstrate compression ratios over 100

Mining Frequent Patterns Using FP-tree

- General idea (divide-and-conquer)
 - Recursively grow frequent pattern path using the FP-tree

