CMPUT 695 Term Project Demo
Mining Frequent Item-sets with Recurrent Items in Multimedia Database

Based on "*Mining Recurrent Items in Multimedia with Progressive Resolution Refinement*"

Osmar R. Zaiane Jaiwei Han, Hua Zhu

Professor: Osmar R. Zaiane

Ying Yuan Bin Cheng
{cheng, yyuan}@cs.ualberta.ca

Outline

- Multimedia associations with recurrent items
- Frequent item-sets with recurrent items
- MaxOccur Algorithm
- Implementation
- Experimentation Results

Outline

- Multimedia associations with recurrent items
- Frequent item-sets with recurrent items
- MaxOccur Algorithm
- Implementation
- Experimentation Results

Multimedia associations with recurrent items

- Visual data has some peculiarities
  - e.g., Some visual features can occur multiple times in an image
  - repetition may carry more information
- An image can be modeled by a transaction
  - items: visual features in the image
  - transaction ID: image ID

Multimedia associations with recurrent items (cont.)

- Apriori: Duplicates are never considered when k-item candidate sets C_k are formed
- In multimedia mining
  - $2\ blue\ circles \Rightarrow high\ texture\ density$
  - Two occurrences have to co-exist in the image for the rule to be valid
- Two notions of *support*
  - Transaction-based support
  - Objected-based support

Multimedia associations with recurrent items (cont.)

- Multimedia associations rule with recurrent items

$$\alpha P_1 \wedge \beta P_2 \wedge .... \wedge \gamma Pn \rightarrow \delta Q_1 \wedge \lambda Q_2 ... uQm(c\%)$$

- A pattern p is sufficiently frequent in a set D at level l if the support of p is no less than its corresponding minimum support threshold and no more than its corresponding maximum support threshold
- A multimedia association rule P→Q in a set of images D is sufficiently strong in D if P and Q are sufficiently frequent and the confidence ofP→Q is greater than the threshold φ`

Outline

- Multimedia associations with recurrent items
- Frequent item-sets with recurrent items
- MaxOccur Algorithm
- Implementation
- Experimentation Results

Frequent item-sets with recurrent items

- A naïve approach

The image set with 6 images.
Maximum object types: 5
Maximum objects in an image: 4

Image ID	Num of Obj.	Content of Images
1	3	{3,4,5}
2	3	{1,5,3}
3	3	{4,5,3}
4	3	{3,3,2}
5	3	{5,1,2}
6	4	{5,1,3,5}

Frequent item-sets with recurrent items (cont.)

- A naïve approach
  - Find all frequent 1-item-sets
  - check how often they might re-occur in an image (maximum occurrence)
  - For each k-item-set, combine these frequent 1-item-sets in the sets of k elements
  - The calculation of the support would filter out the infrequent ones

This naïve algorithm guarantees to find all frequent item-sets with recurrent items.

Outline

- Multimedia associations with recurrent items
- Frequent item-sets with recurrent items
- MaxOccur Algorithm
- Implementation
- Experimentation Results

MaxOccur algorithm

Image ID	Content of Images
1	{2,4,2,2,5,}
2	{2,4,2,4}
3	{4,3,2}
4	{6,7}
5	{4,2,4,2,3,1}

Table 2. Image Tranction Table

Object	Support	Max. Occurrence
1	1	1
2	8	3
3	2	1
4	6	2
5	1	1
6	1	1
7	1	1

Table 3: C1 and M table.

MaxOccur algorithm (cont.)

Object	Support	Max. Occurrence
1	8	3
2	6	1
3	2	2

Table 4. Frequent 1 item set

Image ID	Content of Images
1	{2,4,2,2}
2	{2,4,2,4}
3	{4,3,2}
4	{4,2,4,2,3}

Table 5: Filtered image transaction talbe D2

MaxOccur algorithm (cont.)

2 item-sets	Support
{3,2}	2
{4,2}	6
{4,3}	2
{2,2}	3
{4,4}	2

Talbe 6: Generation of candidate 2 itemsets

2 item-sets	Support
{3,2}	2
{4,2}	6
{4,3}	2
{2,2}	3
{4,4}	2

Table 7: Frequent 2 itemsets

MaxOccur algorithm (cont.)

3 item-sets	Support
{4,3,2}	2
{3,2,2}	1
{4,2,2}	3
{4,4,2}	2
{4,4,3}	1
{2,2,2}	1

Table 8: Candidate 3 itemsets

3 item-sets	Support
{4,3,2}	2
{4,2,2}	3
{4,4,2}	2

Table 9: Frequent 3 itemsets

MaxOccur algorithm (cont.)

4 item-sets	Support
{4,4,2,2}	2

Table 9: Candidate 4 itemsets

4 item-sets	Support
{4,4,2,2}	2

Table 10: Frequent 4 itemsets

MaxOccur algorithm (cont.)

Rules

(1) $\{O_4,O_4\} \rightarrow \{O_2,O_2\}$ [100%] (2) $\{O_2,O_4,O_4\} \rightarrow \{O_2\}$ [100%]

(3) $\{O_3,O_4\} \rightarrow \{O_2\}$ [100%] (4) $\{O_3\} \rightarrow \{O_2,O_4\}$ [100%]

(5) $\{O_2,O_2\} \rightarrow \{O_4\}$ [100%] (6) $\{O_4,O_4\} \rightarrow \{O_2\}$ [100%]

(7) $\{O_3\} \rightarrow \{O_2\}$ [100%] (8) $\{O_3\} \rightarrow \{O_4\}$ [100%]

count replicated objects

$2O_4 \rightarrow 2O_2$ [100%], $O_2 \wedge 2O_4 \rightarrow O_2$ [100%], $O_3 \wedge O_4 \rightarrow O_2$ [100%],

$O_3 \rightarrow O_2 \wedge O_4$ [100%], $2O_2 \rightarrow O_4$[100%], $2O_4 \rightarrow O_2$ [100%],

$O_3 \rightarrow O_2$ [100%], $O_3 \rightarrow O_4$ [100%],

**** $O_4 \rightarrow O_2$ is not confident enough, while "$2O_4 \rightarrow 2O_2$" or "$2O_4 \rightarrow O_2$" are 100% reliable.*

MaxOccur algorithm (cont.)

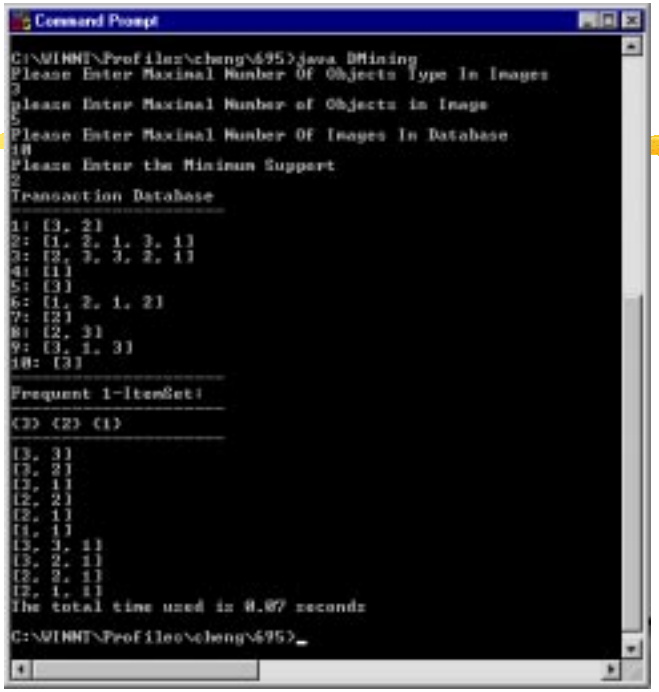
begin
(1) $C_1 \leftarrow$ {Candidate 1 item-sets and their $support$}
(2) $F_1 \leftarrow \{Sufficiently\ frequent$ 1 item-sets and their $support\}$
(3) $M \leftarrow \{Maximum\ occurrence$ in an image of $frequent$ 1 item-sets$\}$
(4) Count # of k-item-sets (total[$1..k$])
(5) for ($i \leftarrow 2; F_{i-1} \neq \emptyset; i \leftarrow i+1$) $do\{$
(6) $C_i \leftarrow (F_{i-1} \bowtie F_{i-1}) \cup$
$\{y \oplus X \mid X \in F_{i-1} \wedge y \in F_1 \wedge Count(y, X) < (M[y]-1)\}$
(7) $C_i \leftarrow C_i - \{c \mid (i-1)$ item-set of $c \notin F_{i-1}\}$
(8) $\mathcal{D}_i \leftarrow FilterTable(\mathcal{D}_{i-1}, F_{i-1})$
(9) foreach image I in $\mathcal{D}_i$ do {
(10) foreach c in C_i do {
(11) $c.support \leftarrow c.support + Count(c, I)$
(12) }
(13) }
(14) $F_i \leftarrow \{c \in C_i \mid \frac{c.support}{total\ i\ itemset} > \sigma t\}$
(15) }
(16) Result $\leftarrow \bigcup_i \{c \in F_i \mid i > 1 \wedge c.support < \Sigma t\}$
end

Outline

- Multimedia associations with recurrent items
- Frequent item-sets with recurrent items
- MaxOccur Algorithm
- Implementation
- Experimentation Results

Implementation of the MaxOccur Algorithm

- We use Java implemented the MaxOccur Algorithm
- The structure of the system



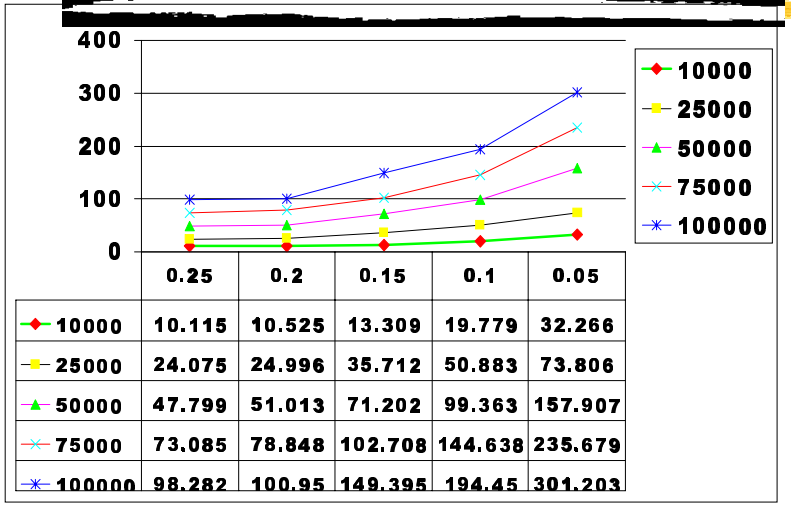
Outline

- Multimedia associations with recurrent items
- Frequent item-sets with recurrent items
- MaxOccur Algorithm
- Implementation
- Experimentation Results

Experimental Results

- We use automatically generated synthetic images to test the system
- Different sized images sets are produced to demonstrate the scalability of the algorithm
- We do the performance evaluation on a Pentium II PC with 128Mb of main memory.

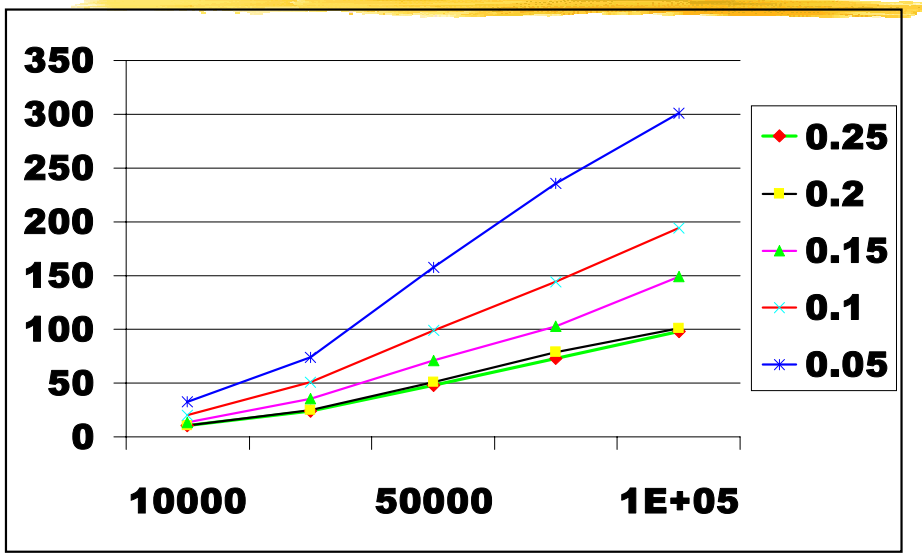
Experimental Results (cont.)



	0.25	0.2	0.15	0.1	0.05
10000	10.115	10.525	13.309	19.779	32.266
25000	24.075	24.996	35.712	50.883	73.806
50000	47.799	51.013	71.202	99.363	157.907
75000	73.085	78.848	102.708	144.638	235.679
100000	98.282	100.95	149.395	194.45	301.203

Experimental Results (cont.)

---Scale up of the system



FP-Tree for Transactions with Recurrent Items

- FP-tree Construction
- A Few Observations
- Example

FP-Tree Construction

Table 1. Frequent 1-Item Sets with Support and Max. Occurrence

Object	Support	Max. Occurrence
O2	5	3
O4	6	2
O5	2	1

Table 2. Filtered Transaction Table

TID	(Ordered) Frequent Items
I1	O2, O2, O2, O4
I2	O2, O2, O4, O4
I3	O2, O4, O3
I4	O2, O2, O4, O4, O5

FP-Tree Construction (Cont.)

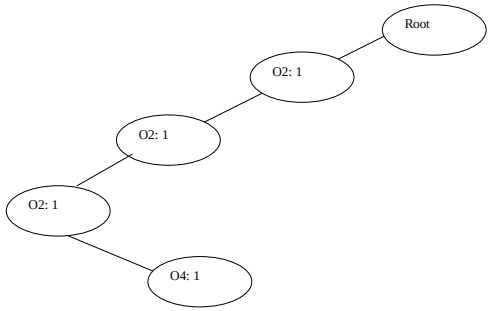


Figure 1

FP-Tree Construction (Cont.)

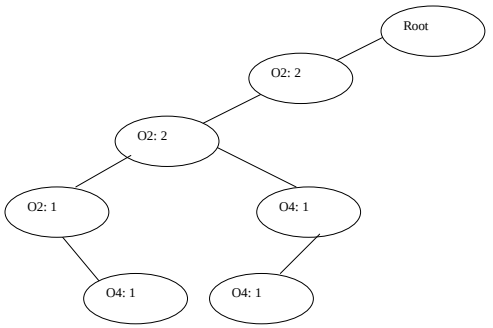


Figure 2

FP-Tree Construction (Cont.)

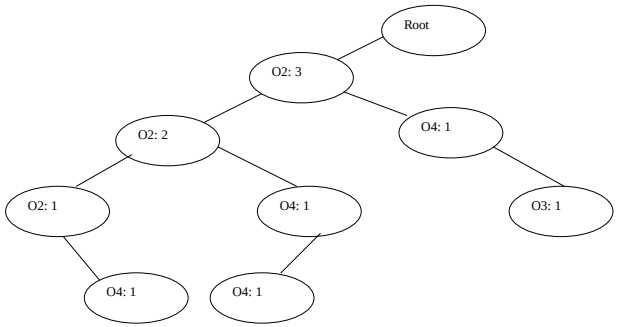


Figure 3

FP-Tree Construction (Cont.)

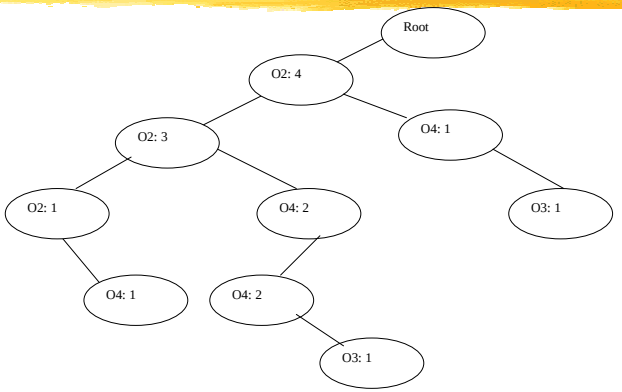


Figure 4

A Few Observations

- Really two database scans?
- How header table is different from the case without item recurrent?
- Is the method good for the domain?

Example

Transaction Database

1: [4]
2: [9, 1, 5, 2]
3: [14, 6, 3, 7, 12, 5, 11, 14]
4: [1, 9, 11, 14, 12, 12, 9, 12]

Frequent 1-ItemSet:

{12} Support = 4
{14} Support = 3
{9} Support = 3
{11} Support = 2
{5} Support = 2
{1} Support = 2

Sorted Transaction Database

1: []
2: [9, 5, 1]
3: [12, 14, 14, 11, 5]
4: [12, 12, 12, 14, 9, 9, 11, 1]

D:\695newproject\695asof120100\Project4>