DSAA 5002 - HW1

50015976 Ruiming ZHANG

Q1 [15 Marks]

Given the transaction database below,set the minimum support count to 2 and the minimum confidence level to 60% to find the strong association rule. Generate the set C_3 of the candidate 3-itemset, using prunning on Apriori principle.

Solution:

TID	Item
T1	A, C, D
T2	B, C, E
T3	A, B, C, E
T4	B, E
T5	A, C, E

1^{st} scan
calculate sup

C_1

Itemset	sup
{A}	3
{B}	3
{C}	4
{D}	1
{E}	4

1^{st} prunning
discard sup<2

L_1

Itemset	sup
{A}	3
{B}	3
{C}	4
{E}	4

generate 2-itemsets

Itemset
{A, B}
{A, C}
{A, E}
{B, C}
{B, E}
{C, E}

C_2

2^{nd} scan
calculate sup

Itemset	sup
{A, B}	1
{A, C}	3
{A, E}	2
{B, C}	2
{B, E}	3
{C, E}	3

C_2

2^{nd} prunning
disgard sup<2

L_2

Itemset	sup
{A, C}	3
{A, E}	2
{B, C}	2
{B, E}	3
{C, E}	3

generate the candidate 3-itemset C_3

Itemset
{A, C, E}
{B, C, E}

C_3

3^{rd} scan

Itemset	sup
{A, C, E}	2
{B, C, E}	2

L_3

Thus we get the candidate 3-itemset C_3.

Q2 [15 Marks]

Reducing the transactions using dynamic hashing and prunning(DHP) algorithm. Set the minimum support count to 2.

Hash function bucket #= $h(\{xy\}) = ((\text{order of } x) * 10 + (\text{order of } y))\%7$

Solution

TID	Item
T1	A, B, C
T2	B, D, E
T3	A, B, D, E
T4	B, E

1^{st} scan $\rightarrow$

C_1

Itemset	sup
{A}	2
{B}	4
{C}	1
{D}	2
{E}	3

1^{st} prunning $\rightarrow$

L_1

Itemset	sup
{A}	2
{B}	4
{D}	2
{E}	3

Find all 2-itemset of each transaction

TID	2-itemset
T1	{A B} {A C} {B C}
T2	{B D} {B E} {D E}
T3	{A B} {A D} {A E} {B D} {B E} {D E}
T4	{B E}

Because we have:

- Items = A, B, C, D, E,
- Order = 1, 2, 3, 4, 5
- Hash function : $h(\{xy\}) = ((\text{order of } x) * 10 + (\text{order of } y))\%7$

Thus we have the hash table below:

bucket	0	1	2	3	4	5	6
count	1	1	1	4	3	2	1
2-itemset	{A D}	{A E}	{B C}	{B D} {D E} {B D} {D E}	{B E} {B E} {B E}	{A B} {A B}	{A C}

Let L_1*L_1 to generate a 2-itemset table, and choose the itemsets where the number of content in its bucket is above the minimum support.

L_1*L_1	# in the bucket
{A B}	2
{A D}	1
{A E}	1
{B D}	4
{B E}	3
{D E}	4

C_2

Resulted C_2
{A B}
{B D}
{B E}
{D E}

L_2

C_2	sup
{A B}	2
{B D}	2
{B E}	3
{D E}	2

Because if an item occurs in a frequent (k+1)-itemset, it must occur in at least k candidate k-itemsets.

TID	Item	2-itemset occurs	
T1	A, B, C	{A B}	Disgard
T2	B, D, E	{B D} {B E} {D E}	Keep {B D E}
T3	A, B, D, E	{A B} {B D} {B E} {D E}	Keep {B D E}
T4	B, E	{E E}	Disacrd

TID	Item
T2	B, D, E
T3	B, D, E

Thus we have reduced the transactions.

Q3 [35 Marks]

An itemset X is said to be a frequent itemset if the frequency count of X is at least a given support threshold.

An itemset Y is a proper super-itemset of X if $X \subset Y$ and $X \neq Y$.

An itemset X is said to be a closed frequent itemset if (1) X is frequent and (2) there exists no proper super-itemset Y of X such that Y is frequent and Y has the same frequency count as X.

An itemset X is said to be a maximal frequent itemset if (1) X is frequent and (2) there exists no proper super-itemset Y of X such that Y is frequent.

Let F_c be the set of (traditional) frequent itemsets each of which is associated with a frequency in the dataset.

For example, if there are three frequent itemsets, $\{I_1\}$ with frequency 4, $\{I_2\}$ with frequency 5, and $\{I_1, I_2\}$ with frequency 3, F={$\{I_1\}$, $\{I_2\}$, $\{I_2, I_2\}$} and $F_c = \{< \{I_1\}, 4 >, < \{I_2\}, 5 >, < \{I_1, I_2\}, 3 >\}$.

Similarly, let C be the set of closed frequent itemsets without specifying the frequency of itemsets.

Let C_c be the set of closed frequent itemsets each of which is associated with a frequency of itemsets.

Let M be the set of maximal frequent itemsets without specifying the frequency of itemsets.

Ler M_c be the set of maximal frequent itemsets each of which is associated with a frequency in the dataset.

The following shows six transactions with four items. Each row corresponds to a transaction where 1 corresponds to a presence of an item and 0 corresponds to an absence.

A	B	C	D
0	0	1	1
1	1	0	0
0	0	1	1
1	0	1	1
1	0	0	0
0	0	0	1

Suppose that the support threshold is 2.

(a) (i) What is F_C? (ii) What is C_c? (iii)What is M_c? **(5 Marks)**

(b) (i) What is the advantages and the disadvantages of using closed frequent itemsets compared with traditional frequent itemsets? **(5 Marks)**

(ii) What are the advantages and the disadvantages of using closed frequent itemsets compared with maximal frequent itemsets? **(5 Marks)**

(c) Please adapt algorithm FP-growth with the use of the FP-tree to find all closed frequent item set. Please write down how to adapt algorithm FP-growth and illustrate the adapted algorithm with the above example. **(20 Marks)**

Solution

(a) According to the topic, we have the following transaction database. And we generate all the k-itemsets which might be frequent itemsets.

TID	transactions
T100	{C, D}
T200	{A, B}
T300	{C, D}
T400	{A, C, D}
T500	{A}
T600	{D}

transaction database

k=1

Itemset	sup
{A}	3
{B}	1
{C}	3
{D}	4

L_1

disgard sup<2

Itemset	sup
{A}	3
{C}	3
{D}	4

C_1

k=2

Itemset	sup
{A, C}	1
{A, D}	1
{C, D}	3

L_2

disgard sup<2

Itemset	sup
{C, D}	3

C_2

i: We have $F_c = \{< \{A\}, 3 >, < \{C\}, 3 >, < \{D\}, 4 >, < \{C, D\}, 3 >\}$

ii: We have $C_c = \{< \{A\}, 3 >, < \{D\}, 4 >, < \{C, D\}, 3 >\}$

iii: We have $M_c = \{< \{A\}, 3 >, < \{C, D\}, 3 >\}$

(b) i:

- Advantages: Compared with traditional frequent itemsets, inclosed itemsets are eliminated, which reduces the number of itemsets. And because a lot of frequent itemsets can be considered as subsets of a closed super-itemset, it does not lose much information.
- Disadvantages: Finding frequent closed itemsets need more computation, increasing the time complexity.

ii:

- Advantages: Closed itemsets do not lose much information because the closed super-itemsets have same frequency as the sub-itemsets. But maximal frequent itemsets may lose some information.
- Disadvantages: Maximal frequent itemsets are more efficient than closed itemsets, because it does not need to check the frequency of the super-itemsets.

(c) Generate a FP-tree. Firstly, deduce the ordered frequent items.

transaction database

TID	transactions
T100	{C, D}
T200	{A, B}
T300	{C, D}
T400	{A, C, D}
T500	{A}
T600	{D}

item&frequency

Item	frequency
A	3
B	1
C	3
D	4

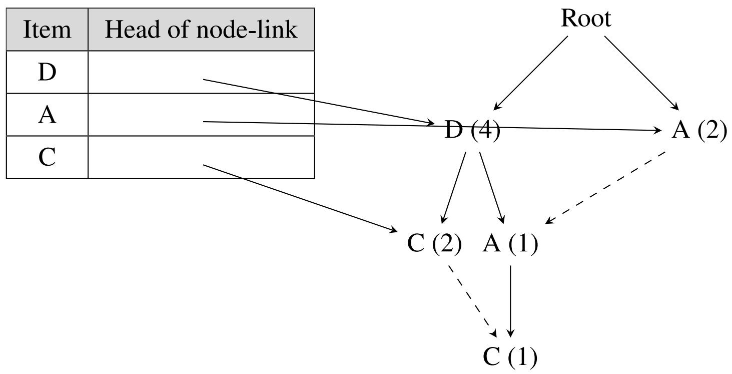
ordered frequency

Item	frequency
D	4
A	3
C	3

TID	transactions	ordered transactions
T100	{C, D}	{D, C}
T200	{A, B}	{A}
T300	{C, D}	{D, C}
T400	{A, C, D}	{D, A, C}
T500	{A}	{A}
T600	{D}	{D}

ordered transaction database

Then we construct the FP-tree from the above data.



Notice that if X is a frequent itemset but not closed, then there exists a super-itemset Y of X such that Y is frequent and Y has the same frequency count as X. Let $Z = Y - X$, (Z is not empty), the for every path lying X, Z must lie in the path, too.

So when we construct the FP-conditional tree for a item, we could check if there exists a common item in all the paths, if not, the item is a closed frequent itemset.

Construct the FP-conditional tree for C, A, D.

for C

Cond. FP-tree on ”C”
{(D:2, C:2), (D:1, A:1, C:1)}

$\longrightarrow$

Cond. FP-tree on ”C”
{(D:2, C:2), (D:1, C:1)}

Because for every path lying C, D also lies in the path, so C is not a closed frequent itemset. But $\{D, C\}$ is a closed frequent itemset.

for A

Cond. FP-tree on ”A”
{(D:1, A:1), (A:2)}

$\longrightarrow$

Cond. FP-tree on ”A”
{(A:3),}

Because there is no other itemsets lies in the right path lying A, so A is a closed frequent itemset.

for D

Cond. FP-tree on ”D”
{(D:4)}

$\longrightarrow$

Cond. FP-tree on ”D”
{(D:4),}

Because there is a path only lying D, so D is a closed frequent itemset.

Thus we have $C_c = \{< \{D, C\}, 3 >, < \{A\}, 3 >, < \{D\}, 4 >\}$.

Q4 [35 Marks]

A GSP example: Suppose now we have 5 events: ’Upload Songs’, ’Add Tags’, ’Share’, ’Listen’ and ’Comment’. Let min-support be 40%. The sequence database of a Music Platform is shown in following table:

Object	Sequence
A	<{’Upload Songs’, ’Add Tags’}>
B	<{’Upload Songs’, ’Share’}>
C	<{’Upload Songs’}, {’Share’, ’Listen’}>
D	<{’Upload Songs’}, {’Upload Songs’, ’Add Tags’}, {’Listen’}>
E	<{’Listen’}, {’Add Tags’, ’Comment’}, {’Share’, ’Listen’}>

Please answer the following questions:

(a) Make the first pass over the sequence database to yield all the 1-element **frequent** sequences and what is the corresponding support? **5 Marks**

(b) Based on (a), do the 2-sequences Candidate Generation and Candidate Pruning. **10 Marks**

(c) What is the **frequent** 2-sequences based on the result of (b)? **5 Marks**

(d) Based on (c), do the 3-sequences Candidate Generation and Candidate Pruning. When a sequence should be pruned, you need to explain why. **10 Marks**

(e) What is the frequent 3-sequences based on the result of (d)? Please calculate the support. **5 Marks**

Remember: For frequent k-sequences, the support >= min-support

Solution

For easier reading, we denote ’Upload Songs’, ’Add Tags’, ’Share’, ’Listen’ and ’Comment’ by U, A, S, L and C respectively. And if there are 2 items in 1 set, we have arranged it in alphabetical order by the first letter. Thus we have:

Object	Sequence
A	<{’Upload Songs’, ’Add Tags’}>
B	<{’Upload Songs’, ’Share’}>
C	<{’Upload Songs’}, {’Share’, ’Listen’}>
D	<{’Upload Songs’}, {’Upload Songs’, ’Add Tags’}, {’Listen’}>
E	<{’Listen’}, {’Add Tags’, ’Comment’}, {’Share’, ’Listen’}>

Object	Sequence
A	<{A, U}>
B	<{S, U}>
C	<{U}, {L, S}>
D	<{U}, {A, U}, {L}>
E	<{L}, {A, C}, {L, S}>

(a) Candidate 1-sequences are:

<{A}>, <{C}>, <{L}>, <{S}>, <{U}>

According to the database above, we have:

Sequence	Sup
<{A}>	60%
<{C}>	20%
<{L}>	60%
<{S}>	60%
<{U}>	80%

disgard sup<40% →

Sequence	Sup
<{A}>	60%
<{L}>	60%
<{S}>	60%
<{U}>	80%

The 1-element frequent sequences and the corresponding support are:

<{’Add Tags’}> (support=60%)

<{’Listen’}> (support=60%)

<{’Share’}> (support=60%)

<{’Upload Songs’}> (support=80%)

(b) Base case ($k = 2$): Merging two frequent 1-sequences <{i_1}> and <{i_2}> will produce two candidate 2-sequences: <{i_1} {i_2}> and <{i_1, i_2}>

Candidate 2-sequences are:

<{A, L}>, <{A, S}>, <{A, U}>, <{L, S}>, <{L, U}>, <{S, U}>,

<{A}, {A}>, <{A}, {L}>, <{A}, {S}>, <{A}, {U}>,

<{L}, {A}>, <{L}, {L}>, <{L}, {S}>, <{L}, {U}>,

<{S}, {A}>, <{S}, {L}>, <{S}, {S}>, <{S}, {U}>,

<{U}, {A}>, <{U}, {L}>, <{U}, {S}>, <{U}, {U}>

All the 1-sequences we generate 2-sequences from are frequent. So after candidate prunning, the 2-sequences should remain the same.

(c) After candidate elimination, frequent 2-sequences are:

<{A, U}> (support=40%),

<{L, S}> (support=40%),

<{A}, {L}> (support=40%),

<{U}, {L}> (support=40%)

(d) Generate 3-sequences from the remaining 2-sequences, 3-sequences are:

<{A, U}, {L}> (generate from <{A, U}> and <{U}, {L}> or from <{A, U}> and <{A}, {L}>),

<{A}, {L, S}> (generate from <{A}, {L}> and <{L, S}),

<{U}, {L, S}> (generate from <{U}, {L}> and <{L, S})

Pruning:

<{A}, {L, S}> should be pruned because one 2-subsequence <{A}, {S}> is not frequent.

<{U}, {L, S}> should be pruned because one 2-subsequence <{U}, {S}> is not frequent.

(e) <{A, U}, {L}> (support=20% < 40%, should be eliminated)

Thus, there is no frequent 3-sequence.