DSAA 5002 - Data Mining and Knowledge Discovery in Data Science

(Fall Semester 2023)

Homework 1 Solution

Q1:

Step 1: Scan the transaction database, generate the candidate 1-itemset C_1, and calculate the support of each C_1 itemset.

C_1

Itemset	Support Count
{A}	3
{B}	3
{C}	4
{D}	1
{E}	4

Step 2: Determine the frequent 1-itemset L_1.

L_1

Itemset	Support Count
{A}	3
{B}	3
{C}	4
{E}	4

Step 3: Generate the candidate 2-itemset C_2, $C_2 = L_1 * L_1$.

C_2

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

Step 4: Scan the transaction database and calculate the support of each C_2 itemset.

C_2

Itemset	Support Count
{A,B}	1
{A,C}	3
{A,E}	2
{B,C}	2

{B,E}	3
{C,E}	3

Step 5: Determine the frequent 2-itemset L_2.

L_2

Itemset	Support Count
{A,C}	3
{A,E}	2
{B,C}	2
{B,E}	3
{C,E}	3

Step 6: Generate the candidate 3-itemset C_3, $C_3 = L_2 * L_2$. Prun on Apriori principle.

Apriori principle: If an itemset is frequent, then all of its subsets must also be frequent.

Apriori pruning principle: If there is any itemset which is infrequent, its superset should not be generated/tested.

C_3

Itemset	2-item subset	2-item subset	2-item subset	Prunned
{A,C,E}	{A,C}	{A,E}	{C,E}	√
{A,B,C}	~~{A,B}~~	{A,C}	{B,C}	×
{A,B,E}	~~{A,B}~~	{A,E}	{B,E}	×
{B,C,E}	{B,C}	{B,E}	{C,E}	√

{A,B} item subset is not belong to L_2, so {A,B,C} and {A,B,E} are prunned.

Prunned C_3

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

Step 7: Scan the transaction database and calculate the support of each C_3 itemset.

C_3

Itemset	Support Count
{A,C,E}	2
{B,C,E}	2

Step 8: Determine the frequent 3-itemset L_3.

L_3

Itemset	Support Count
{A,C,E}	2
{B,C,E}	2

Step 9: Generate the candidate 4-itemset C_4, $C_4 = L_3 * L_3$.

C_4

Itemset	3-item subset	3-item subset	3-item subset	3-item subset	Prunned

{A,B,C,E}	~~{A,B,C}~~	~~{A,B,E}~~	{B,C,E}	{A,C,E}	×

{A,B,C} and {A,B,E} item subsets are not belong to L_3, so {A,B,C,E} is prunned.
Prunned $C_4 = null$

Step 10: Generate the association rules from frequent itemsets and calculate confidence levels.
confidence(S->(I-S)) = support(I)/support(S)

For the 2-Itemset

Frequent Itemset	Subset	Association rules	Support(I)	Support(S)	Confidence	Confidence(%)
{A,C}	{A}	{A}->{C}	3	3	3/3	100%
	{C}	{C}->{A}	3	4	3/4	75%
{A,E}	{A}	{A}->{E}	2	3	2/3	66.66%
	{E}	~~{E}->{A}~~	2	4	2/4	50%
{B,C}	{B}	{B}->{C}	2	3	2/3	66.66%
	{C}	~~{C}->{B}~~	2	4	2/4	50%
{B,E}	{B}	{B}->{E}	3	3	3/3	100%
	{E}	{E}->{B}	3	4	3/4	75%
{C,E}	{C}	{C}->{E}	3	4	3/4	75%
	{E}	{E}->{C}	3	4	3/4	75%

Given the minimum confidence level to 60%, so the final association rules:
Rule1: {A}->{C}
Rule2: {C}->{A}
Rule3: {A}->{E}
Rule4: {B}->{C}
Rule5: {B}->{E}
Rule6: {E}->{B}
Rule7: {C}->{E}
Rule8: {E}->{C}

For the 3-Itemset

Frequent Itemset	Subset	Association rules	Support(I)	Support(S)	Confidence	Confidence(%)
{A,C,E}	{A}	{A}->{C,E}	2	3	2/3	66.66%
	{C}	~~{C}->{A,E}~~	2	4	2/4	50%
	{E}	~~{E}->{A,C}~~	2	4	2/4	50%
	{A,C}	{A,C}->{E}	2	3	2/3	66.66%
	{A,E}	{A,E}->{C}	2	2	2/2	100%
	{C,E}	{C,E}->{A}	2	3	2/3	66.66%
{B,C,E}	{B}	{B}->{C,E}	2	3	2/3	66.66%
	{C}	~~{C}->{B,E}~~	2	4	2/4	50%
	{E}	~~{E}->{B,C}~~	2	4	2/4	50%

	{B,C}	{B,C}->{E}	2	3	2/3	66.66%
	{B,E}	{B,E}->{C}	2	2	2/2	100%
	{C,E}	{C,E}->{B}	2	3	2/3	66.66%

Given the minimum confidence level to 60%, so the final association rules:

Rule1: {A}->{C,E}

Rule2: {A,C}->{E}

Rule3: {A,E}->{C}

Rule4: {C,E}->{A}

Rule5: {B}->{C,E}

Rule6: {B,C}->{E}

Rule7: {B,E}->{C}

Rule8: {C,E}->{B}

Q2:

Step 1: Scan the transaction database, generate the candidate 1-itemset C_1, and calculate the support of each C_1 itemset.

C_1

Itemset	**Support Count**
{A}	2
{B}	4
{C}	1
{D}	2
{E}	3

Step 2: Determine the frequent 1-itemset L_1.

L_1

Itemset	**Support Count**
{A}	2
{B}	4
{D}	2
{E}	3

Step 3: Find all 2-itemset of each transaction.

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

Step 4: Map the 2-itemset of transactions into the hash table

Items = A, B, C, D, E

Order = 1, 2, 3, 4, 5

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

				{D,E}			
				{B,D}	{B,E}		

				{D,E}	{B,E}	{A,B}	
	{A,D}	{A,E}	{B,C}	{B,D}	{B,E}	{A,B}	{A,C}
Number	1	1	1	4	3	2	1
Bucket	0	1	2	3	4	5	6

Step 5: L_1*L_1

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

Step 6: Choose the itemsets where the number of content in its bucket is above the minimum support.

After DHP C_2

2-itemset
{A,B}
{B,D}
{B,E}
{D,E}

Step 7: Determine the frequent 2-itemset L_2.

L_2

2-itemset	Count
{A,B}	2
{B,D}	2
{B,E}	3
{D,E}	2

Step 8: Discard transactions using DHP

DHP:

If an item occurs in a frequent (k+1)-itemset, it must occur in at least k candidate k-itemsets (necessary but not sufficient)

Discard an item if it does not occur in at least k candidate k-itemsets during support counting

TID	Item	2-itemset	DHP Pruned
~~T1~~	~~A,B,C~~	{A,B}	Discard
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~~	{B,E}	Discard

So, the D_3 = {<T2,B C E>,<T3,B C E>} after discarding transactions using DHP

Q3:

(a)

C_1

Itemset	sup
A	3
B	1
C	3
D	4

→

L_1

Itemset	sup
A	3
C	3
D	4

→

C_2

2temset
AC
AD
CD

→

C_2

2temset	sup
AC	1
AD	1
CD	3

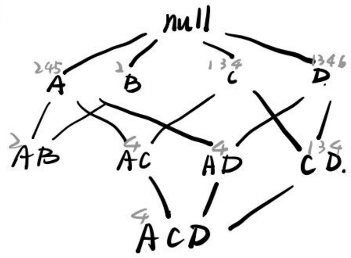
→

L_2

2temset	sup
CD	3

(i) F_c={<{A},3>, <{C},3>, <{D}, 4>, <{C,D}, 3>

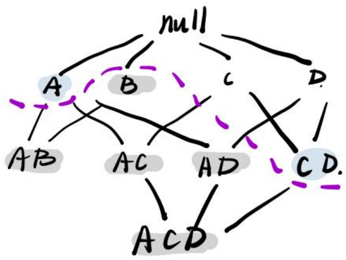
(ii)



C={{A}, {D}, {C,D}}

C_c={<{A},3>, <{D},4>, <{C,D},3>}

(iii)



M={{A}, {C,D}}

M_c={<{A},3>,<{C,D},3>}

(a) i. F_c={<{A},3>,<{C},3>,<{D},4>,<{C,D},3>}
ii. C_c={<{A},3>,<{D},4>,<{C,D},3>}
iii. M_c={<{A},3>,<{C,D},3>}

(b) i. Advantages:

1. The number of possible frequent closed itemsets is much smaller than that of frequent itemsets. Therefore, the storage is smaller.
2. We do not need to generate all possible frequent itemsets. Therefore, execution time is shorter.

Disadvantages:

1. We need to expand the closed frequent itemset in order to check whether an itemset is frequent or not.
2. It is not easy for users to understand the closed frequent itemsets directly.

ii. Advantages:

1. According to closed frequent itesmsets with C_c, we can generate all traditional frequent itemsets associated with frequecies (i.e. F_c). However, according to maximal frequent itemsets with M_c, we cannot generate all traditional frequent itemsets associated with frequencies (i.e. F_c).
2. Closed frequent itemsets are useful to remove some of the redundant association rules. According to the concept of closed frequent itemsets, we can define redundant association rules as follows. $X \rightarrow Y$ is redundant if there exists another rule $X' \rightarrow Y'$ where $X' \subset X$ and $Y' \subset Y$, and their support and confidence are the same. Such redundant rules are not generated if closed frequent itemsets are used for rule generation.

Disadvantages:

1. The number of possible closed frequent itemsets is much larger than that of frequent itemsets. So the storage is larger.
2. It is not easy for users to understand the closed frequent itemsets directly.

(c) We can use the same FP-growth approach to mine closed frequent itemset. We just need to make the following adaptions.

1. When we generate conditional FP-trees, we process the items in the header table from the bottom to the top.
2. Whenever we generate an itemset, we have to check whether this itemset is in the set of itemsets generated before.

Algorithm FP-growth (Tree,α)

1. A $\leftarrow \varnothing$; B $\leftarrow \varnothing$
2. If tree contains a single path P
   - Partition the path into different segments such that all nodes in each segment have the same node support
   - For each segment S
     - Choose the node N in S at the deepest height
     - $\beta \leftarrow$ a set of nodes N to the root
     - Generate the pattern $\beta \cup \alpha$ with support equal to minimum support of nodes in β
     - If there does **NOT** exist an itemset I in A s.t. I.count= $(\beta \cup \alpha)$.count,

       Insert $\beta \cup \alpha$ into A

   else
   - For each a_i in the header of Tree,

     - Generate pattern $\beta = a_i \cup \alpha$ with support equal to a_i .support
     - Insert this pattern into B
     - Construct β's conditional pattern base and then β's conditional FP-tree $Tree_\beta$
     - If $Tree_\beta \neq \varnothing$

       Call FP-growth($Tree_\beta$, β)
3. Last Step

   (a) Process itemsets in B in descending order of the size of the itemsets

   (b) For each itemset I_B in B,

   (c) Check whether there exists an itemset I_A in A s.t. I_A is a super-itemset of I_B and I_A .count = I_B .count. If no, insert I_B into A

   (d) Output A.

E.g.:

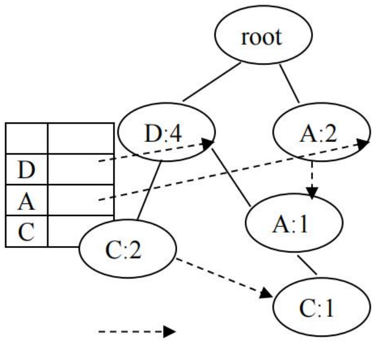
	frequency
A	3
B	1
C	3
D	4

	frequency
A	3
C	3
D	4

Sorted:

	frequency
D	4
A	3
C	3

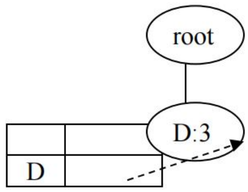
Trans: D,C
A
D, C
D, A, C
A
D



Then, conditional tree on C is:
Trans:
D:1, A:1, C:1
D:2, C:2

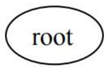
	freq
A	1
D	3
C	3

	freq
D	3
C	3



B={<{C},3>}
A={<{C,D},3>}

Conditional tree on A is:
Trans:
D:1, A:1
A:2



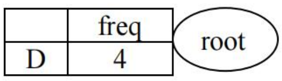
	freq
D	1
A	3

	freq
A	3

B={<{C},3>,<{A},3>}
A={<{C,D},3>}

Conditional tree on D is:
Trans:
D: 4

	freq
D	4



B={<{C},3>,<{A},3>,<{D},4>}
A={<{C,D},3>}

Last Step:
Finally, A={<{C,D},3>,<{D},4>,<{A},3>}

Q4:

(a) (5 marks){ 'Upload Songs'}: 80%, {'Add Tags'}: 60%, {'Share'}: 60%, {'Listen'}: 60%

(b) **(10marks) 2-sequences Candidate Generation:**

<{ 'Upload Songs', 'Add Tags'}>, <{ 'Upload Songs', 'Share'}>, <{ 'Upload Songs', 'Listen'}>,
<{'Add Tags', 'Share'}>, <{'Add Tags', 'Listen'}>, <{'Share', 'Listen'}>,
<{'Upload Songs'}, {'Upload Songs'}>, <{'Upload Songs'}, {'Add Tags'}>, <{'Upload Songs'}, {'Share'}>, <{'Upload Songs'}, {'Listen'}>,
<{'Add Tags'}, {'Upload Songs'}>, <{'Add Tags'}, {'Add Tags'}>, <{'Add Tags'}, {'Share'}>,
<{'Add Tags'}, {'Listen'}>,
<{'Share'}, {'Upload Songs'}>, <{'Share'}, {'Add Tags'}>, <{'Share'}, {'Share'}>,
<{'Share'}, {'Listen'}>,
<{'Listen'}, {'Upload Songs'}>, <{'Listen'}, {'Add Tags'}>, <{'Listen'}, {'Share'}>,
<{'Listen'}, {'Listen'}>

Candidate Pruning: Remain unchanged

(c) (5 marks) Frequent 2-sequences:

<{ 'Upload Songs', 'Add Tags'}>: 40%, <{'Share', 'Listen'}>: 40%,
<{'Upload Songs'}, {'Listen'}>: 40%, <{'Add Tags'}, {'Listen'}>: 40%

(d) (10 marks) 3-sequences Candidate Generation:

<{ 'Upload Songs', 'Add Tags'}, {'Listen'}>, <{'Upload Songs'}, {'Share', 'Listen'}>,
<{'Add Tags'}, {'Share', 'Listen'}>

Candidate Pruning:

(1) <{ 'Upload Songs', 'Add Tags'}, {'Listen'}> should not be pruned

(2) <{'Upload Songs'}, {'Share', 'Listen'}> should be pruned because one 2-subsequence <{'Upload Songs'}, {'Share'}> is not frequent

(3) <{'Add Tags'}, {'Share', 'Listen'}> should be pruned because one 2-subsequence <{'Add Tags'}, {'Share'}> is not frequent

(e) (5 marks) Frequent 3-sequences**:**

Since the support of <{ 'Upload Songs', 'Add Tags'}, {'Listen'}> **is 20%. There is no frequent 3-sequence.**