Jennifer MacDonald
604501712
02/01/17

Homework 1

CS 249: Big Data Analytic, Winter 2017
Instructor: Prof. Wei Wang

Due on: Wednesday, February 1, 2017

Instructions

- Please be short and concise in your answers. Do not present unnecessary details.
- A submission link will be created in CCLE few days prior to the due date. Acceptable formats include pdf, doc, docx.
- Please mention your full name and UCLA ID in your submission.

Table 1: A transactional database containing 6 transactions

Trans. ID	Items
1	c, r, a
2	p, b
3	r, c, b
4	p, c, a
5	a, b, s, c
6	t, a, b, c

Table 2: Table for Problem D

Trans. ID	Items
T1	<(a,d)(c,d)>
T2	<adca>
T3	<c>
T4	<(a,b,d)(a,c)(c,e)>
T5	<dc>

Problem A. Apriori and Condensed Representations (10 pts)

1. What is Apriori property? (2 pt)
Apriori property states that any subset of a frequent itemset must also be frequent.

2. Consider the transactional database shown in table 1. Assuming the minimum support as 2, find out all the frequent itemsets with their supports from the given database using the Apriori algorithm. (8 pts)
For each scan of the database, you must answer the following:

- What are the candidates and their corresponding supports?
- Which of these candidates are frequent?
- What are the results obtained from the self-join between the frequent candidates?
- Which self-join results remain after pruning and why?

1st Scan:

1-candidates

Itemset	Sup
a	4
b	4
c	5
p	2
r	2
s	1
t	1

Freq 1-itemsets

Itemset	Sup
a	4
b	4
c	5
p	2
r	2

Freq Self-join Results

Itemset	Sup
ab	2
ac	4
ap	1
ar	1
bc	3
bp	1
br	1
cp	1
cr	2
pr	0

Results After Pruning (2nd Scan)

Itemset	Sup
ab	2
ac	4
bc	3
cr	2

ab, ac, bc, and cr remain after pruning because they have a frequency at or above the minimum support of 2.

2nd Scan:

2-candidates

Itemset	Sup
ab	2
ac	4
bc	3
cr	2

Freq Self-join Results

Itemset	Sup
abc	2
acr	1
bcr	1

Freq 2-itemsets

Itemset	Sup
ab	2
ac	4
bc	3
cr	2

Results After Pruning (3rd Scan)

Itemset	Sup
abc	2

abc remains after pruning because it has a frequency at or above the minimum support of 2.

3rd Scan:

3-candidates

Itemset	Sup
abc	2

Freq Self-join Results

Itemset	Sup
abc	2

Freq 3-itemsets

Itemset	Sup
abc	2

Results After Pruning

Itemset	Sup
abc	2

Frequent itemsets with their supports from the given database are:
{a:4, b:4, c:5, p:2, r:2, ab:2, ac:4, bc:3, cr:2, abc:2}

Problem B. Dynamic Itemset Counting (DIC) (10 pts)

1. Apply Dynamic Itemset Counting (DIC) to the above problem. You can assume that the transactions are read one by one (sequentially) from disk, starting from the transaction ID #1. You only need to show when an itemset becomes frequent (during which scan and after reading which transaction). Compare the number of scans required in DIC to Apriori. (10 pts)

1st Scan:

After transaction 3: b, c, and r become frequent
After transaction 4: a and p become frequent
After transaction 6: ab, ac, and bc become frequent

2nd Scan:

After transaction 3, cr becomes frequent
After transaction 6, abc becomes frequent

The number of scans required using DIC is 2, whereas the number of scans required using Apriori is 3.

Problem C. FP-Tree (20 pts)

1. Using the same transactional database and minimum support constraint as problem A, build the corresponding FP-tree step by step. While ordering the frequent items, use alphabetical order to break ties between the items having same support. (12 pts)

Itemset	Freq
c	5
r	2
a	4
p	2
b	4
s	1
t	1

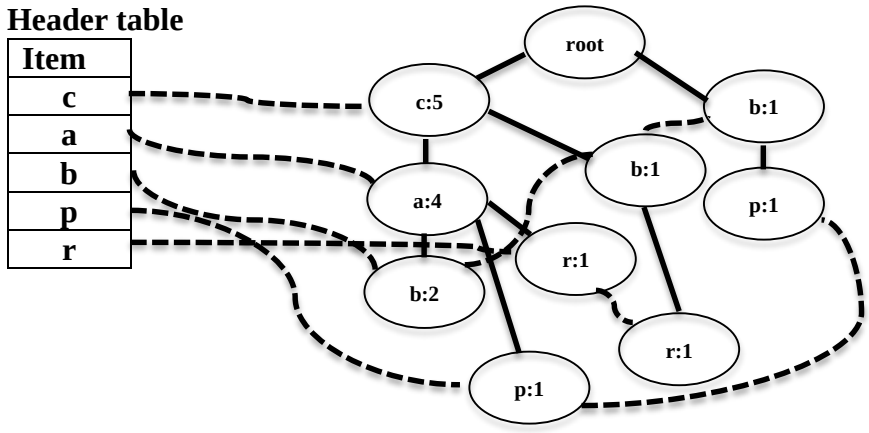
Itemset	Freq
c	5
a	4
b	4
p	2
r	2

F-list: c-a-b-p-r

TID	Items bought	(ordered) freq items
1	c, r, a	c, a, r
2	p, b	b, p
3	r, c, b	c, b, r
4	p, c, a	c, a, p
5	a, b, s, c	c, a, b
6	t, a, b, c	c, a, b

Header table

Item
c
a
b
p
r



2. Use this FP-tree to find all the frequent itemsets that contain 'b' with their supports. You must show all the projected databases and conditional FP-trees generated step by step for this computation. (8 pts)

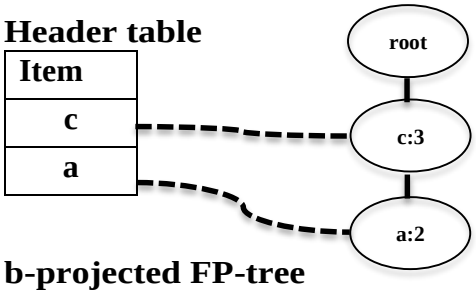
b-projected database $TDB|_b$

ca: 2

c: 1

Local frequent items: c, a

Header table



b-projected FP-tree

Frequent itemsets that contain 'b' with their supports: {cb, ab, acb}

Problem D. Sequential Pattern Mining (10 pts)

1. Compute the support of these three sequences for Table 2: (4 pts)

   (a) <ac> = **3**

   (b) <(a,c)> = **1**

2. Assume that the minimum support is 2, use PrefixScan to find out all the frequent sequential patterns. (6 pts)

Cand	**Sup**
<a>	**3**
<b>	**1**
<c>	**5**
<d>	**4**
<e>	**1**

Prefix			**Projected (suffix) databases**	**Projected Sup**	**Seq Pattern Sup**
<a>			**<(_d)(cd)>, <dca>, <(_bd)(ac)(ce)>**	**<a>:2, <b>:1, <_b>:1, <c>:3, <d>:3, <_d>:2, <e>:1**	**<aa>:2, <ac>:3, <ad>:3, <(ad)>:2**
	<aa>		**<(_c)(ce)>**	**<c>:1, <_c>:1, <e>:1**	
	<ac>		**<(_d)>, <a>, <(ce)>**	**<a>:1, <c>:1, <d>:1, <_d>:1, <e>:1**	
	<ad>		**<(cd)>, <ca>, <(ac)(ce)>**	**<a>:2, <c>:3, <d>:1, <e>:1**	**<ada>:1, <adc>:3**
		<adc>	**<(_d)>, <a>, <(ce)>**	**<a>:1, <c>:1, <d>:1, <_d>:1, <e>:1**	

	<(ad)>		**<(cd)>, <(ac)(ce)>**	**<a>:1, <c>:2, <d>:1, <e>:1**	**<(ad)c>:2**
		<(ad)c>	**<(_d)>, <(ce)>**	**<c>:1, <d>:1, <_d>:1, <e>:1**	
<c>			**<(_d), <a>, <(ce)>**	**<a>:1, <c>:1, <d>:1, <_d>:1, <e>:1**	
<d>			**<(cd)>, <ca>, <(ac)(ce)>, <c>**	**<a>:2, <c>:3, <d>:1, <e>:1**	**<da>:2, <dc>:4**
<da>			**<(_c)(ce)>**	**<c>:1, <_c>:1, <e>:1**	
<dc>			**<(_d)>, <a>, <(ce)>**	**<a>:1, <c>:1, <d>:1, <_d>:1, <e>:1**	

The frequent sequential patterns are: <a>, <aa>, <ac>, <ad>, <adc>, <(ad)>, <(ad)c>, <c>, <d>, <da>, <dc>.

Problem E. Programming Assignment: Apriori (40 pts)

Implement a program that solves a general case of problem A. That is, your code should take in two parameters, minimum support and a file that contains a list of transactions and find all the frequent itemsets in the file. Each row of the file contains one or a combination of items, each separated by a comma delimiter. The item names in the file can be any lower case alphabetical character from a-z. For example possible rows in the file can be:

a,c,e,q
d,e
z

You may assume the inputs are always valid, rows are alphabetically sorted, and there are no duplicate items in each row. You may program in either R, MATLAB, OCTAVE, or python. You may only use the core library functions provided by the language. If you have any doubts of what functions you can use please contact the TA.

While you are free to design the format of your output, your output should be easy to understand. Any ambiguity should be stated in your README file.