HW6 - James Lowrey

Sunday, December 06, 2015 4:25 PM

8.1)a)

Q1. Given the database in Table 8.2.

(a) Using *minsup* = 3/8, show how the Apriori algorithm enumerates all frequent patterns from this dataset.

Table 8.2. Transaction database for Q1

tid	itemset
t_1	*ABCD*
t_2	*ACDF*
t_3	*ACDEG*
t_4	*ABDF*
t_5	*BCG*
t_6	*DFG*
t_7	*ABG*
t_8	*CDFG*

Scan	Candidates	Large Itemsets
1	{A} {B} {C} {D} {E} {F} {G}	{A} {B} {C} {D} {F} {G}
2	{AB} {AC} {AD} {AF} {AG} {BC} {BD} {BF} {BG} {CD} {CF} {CG} {DF} {DG} {FG}	{AB} {AC} {AD} {CD} {CG} {DF} {DG}
3	{ACD} {CDG}	{ACD}

8.2)

Q2. Consider the vertical database shown in Table 8.3. Assuming that *minsup* = 3, enumerate all the frequent itemsets using the Eclat method.

Table 8.3. Dataset for Q2

A	*B*	*C*	*D*	*E*
1	2	1	1	2
3	3	2	6	3
5	4	3		4
6	5	5		5
	6	6		

ItmSet	tid list
A	1 3 5 6
B	2 3 4 5 6
C	1 2 3 5 6
D	1 6
E	2 3 4 5

Itemset	tid list
AB	3 5 6
AC	1 3 5 6
AE	3 5
BC	2 3 5 6
BE	2 3 4 5
CE	2 3 5

Itemset	TIS
ABC	3 5 6
ABE	3 5
ACE	3 5
BCE	2 3 5

8.5)

Q5. Consider the *partition* algorithm for itemset mining. It divides the database into k partitions, not necessarily equal, such that $\mathbf{D} = \cup_{i=1}^{k} \mathbf{D}_i$, where $\mathbf{D}_i$ is partition i, and for any $i \neq j$, we have $\mathbf{D}_i \cap \mathbf{D}_j = \emptyset$. Also let $n_i = |\mathbf{D}_i|$ denote the number of transactions in partition $\mathbf{D}_i$. The algorithm first mines only locally frequent itemsets, that is, itemsets whose relative support is above the *minsup* threshold specified as a fraction. In the second step, it takes the union of all locally frequent itemsets, and computes their support in the entire database $\mathbf{D}$ to determine which of them are globally frequent. Prove that if a pattern is globally frequent in the database, then it must be locally frequent in at least one partition.

Let D be a database with partitions $D_{1 \dots} D_n$. Let x be a frequent pattern in some partition D_i but not globally (the union of all partitions).

$$x \notin \bigcup_{i=1}^{j} D_i$$

with support threshold δ.

Thus support of X (δx) is less than the overall support times the size of D_i's partition: δ * $|D_i|$

Next, run a summation over all the data partitions

$SUM_{i=0 \text{ to } n}$ (δx* D_i) < δ * $|D_i|$

$SUM_{i=0 \text{ to } n}$ (δx* D_i) < δ * |D| as |D| > every $|D_i|$

δx* $SUM_{i=0 \text{ to } n}$ (D_i) < δ * |D|

δx* |D| < δ * |D|

The final result is that δx is less than the overall support required for a global partition, which implies x is not a frequent pattern! The contradict s our original assumption. Thus δx must be >= δ

8.6)

Q6. Consider Figure 8.10. It shows a simple taxonomy on some food items. Each leaf is a simple item and an internal node represents a higher-level category or item. Each item (single or high-level) has a unique integer label noted under it. Consider the database composed of the simple items shown in Table 8.5 Answer the following questions:

a) There are 11 leaf nodes, or simple items. Thus the search space is he size of the power-set of these 11 items: 2 ^11, or 2048
b) The support will increase. More general itemsets (the parents) will be found more frequently than a specific leaf/simple item. Thus its support will increase
c) In this problem we only care-about high level items. Thus, simple nodes should be preprocessed to represent their high level parents. In the following table (Dataset 2) the original dataset has been processed to transform simple nodes to their high level counterparts. IF there is no high level counterpart, the node item is removed from the dataset (as it cannot contribute to a frequent itemset

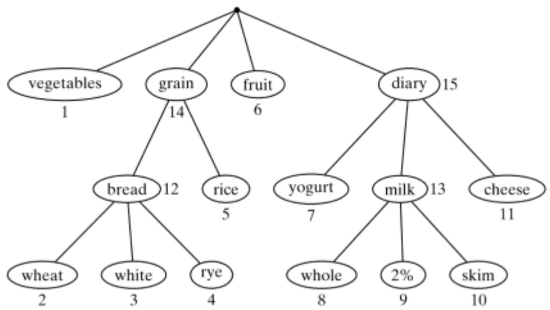

Figure 8.10. Item taxonomy for Q6.

Table 8.5. Dataset for Q6

tid	itemset
1	2 3 6 7
2	1 3 4 8 11
3	3 9 11
4	1 5 6 7
5	1 3 8 10 11
6	3 5 7 9 11
7	4 6 8 10 11
8	1 3 5 8 11

(a) What is the size of the itemset search space if one restricts oneself to only itemsets composed of simple items?

(b) Let $X = \{x_1, x_2, \ldots, x_k\}$ be a frequent itemset. Let us replace some $x_i \in X$ with its parent in the taxonomy (provided it exists) to obtain X', then the support of the new itemset X' is:

- **i.** more than support of X
- **ii.** less than support of X
- **iii.** not equal to support of X
- **iv.** more than or equal to support of X
- **v.** less than or equal to support of X

(c) Use $minsup = 7/8$. Find all frequent itemsets composed only of high-level items in the taxonomy. Keep in mind that if a simple item appears in a transaction, then its high-level ancestors are all assumed to occur in the transaction as well.

c) In this problem we only care-about high level items. Thus, simple nodes should be preprocessed to represent their high level parents. In the following table (Dataset 2) the original dataset has been processed to transform simple nodes to their high level counterparts. IF there is no high level counterpart, the node item is removed from the dataset (as it cannot contribute to a frequent itemset of only high-level nodes). Also, simple nodes with multiple high level distinctions (a node can be both a grain and a bread) are noted (by the '/' symbol) to facilitate in frequent pattern matching.

Dataset 2

TID	itemset
1	12/14, 12/14, 15
2	12/14, 12/14, 13/15, 15
3	12/14, 13/15, 15
4	14, 15
5	12/14, 13/15, 13/15, 15
6	12/14, 14, 15, 13/15, 15
7	12/14, 13/15, 13/15, 15
8	12/14, 14, 13/15, 15

Eclat method to find frequent itemsets

Itemset	12	14	13	15
TID	1, 2, 3, 5, 6, 7, 8	all 8	2, 3, 5, 6, 7, 8	all 8

Itemset	12, 14	14, 15	12, 15
TID	1, 2, 3, 5, 6, 7, 8	all 8	1, 2, 3, 5, 6, 7, 8

Itemset	12, 14, 15
TID	1, 2, 3, 5, 6, 7, 8

Frequent, high level itemsets are: {12} {14} {15} {12, 14} {14, 15} {12, 15} {12, 14, 15}

9.1)

Q1. True or False:

(a) Maximal frequent itemsets are sufficient to determine all frequent itemsets with their supports.

(b) An itemset and its closure share the same set of transactions.

(c) The set of all maximal frequent sets is a subset of the set of all closed frequent itemsets.

(d) The set of all maximal frequent sets is the set of longest possible frequent itemsets.

A) false: cannot find all frequent itemsets but not their supports
B) false: c(X) = X only if X is a closed set. Otherwise they can be different
C) true. Maximal is a subset of closed frequent
D) true. This implies that any other frequent set is a subset of a maximal frequent set

9.2.b)

Q2. Given the database in Table 9.1

(b) Find all frequent, closed, and maximal itemsets using $minsup = 2/6$.

Table 9.1. Dataset for Q2

Tid	Itemset
t_1	*ACD*
t_2	*BCE*
t_3	*ABCE*
t_4	*BDE*
t_5	*ABCE*
t_6	*ABCD*

Frequent sets: All sets in the tables below except for {DE}

itemset	tid set
A	1, 3, 5, 6
B	2, 3, 4, 5, 6
C	1, 2, 3, 5, 6
D	1, 4, 6
E	2, 3, 4, 5

itemset	tid set
AB	3, 5, 6
AC	3, 5, 6
AD	1, 3, 5, 6
AE	3, 5
BC	2, 3, 5, 6
BD	4, 6
BE	2, 3, 4, 5
CD	1, 6
CE	2, 3, 5
DE	4

itemset	tid set
ABC	3, 5, 6
ABD	3, 5, 6
ABE	3, 5
ACD	3, 5, 6
ACE	3, 5
BCD	4, 6
BCE	2, 3, 5

itemset	tid set
ABCD	3, 5, 6
ABCE	3, 5

Maximal frequent sets: {ABCD, ABCE}
Closed Frequent Sets: {ABCD, ABCE, ABC, ACE}

10.1)

Q1. Consider the database shown in Table 10.2. Answer the following questions:

(a) Let $minsup = 4$. Find all frequent sequences.

(b) Given that the alphabet is $\Sigma = \{A, C, G, T\}$. How many possible sequences of length k can there be?

Table 10.2. Sequence database for Q1

Id	Sequence

a) USING SPADE- 1 length sequences

A	
id	pos
1	1,2,4,6,7,9,10
2	3,9

C	
id	pos
1	5,11
2	

(b) Given that the alphabet is $\Sigma = \{A, C, G, T\}$. How many possible sequences of length k can there be?

Table 10.2. Sequence database for Q1

Id	Sequence
s_1	*AATACAAGAAC*
s_2	*GTATGGTGAT*
s_3	*AACATGGCCAA*
s_4	*AAGCGTGGTCAA*

2 length sequences (position is position of ending point of 2-len seq)

AG	
id	pos
1	8
2	5,6,8
3	6,7
4	3,5,7,8

AT	
id	pos
1	3
2	4,10
3	5
4	6,9

AA	
id	pos
1	2,4,6,7,9,10
2	9
3	2,4,10,11
4	2,11,12

GA	
id	pos
1	9,10
2	3,9
3	10,11
4	11,12

GT	
id	pos
1	
2	2,4,7,10
3	
4	6,9

GG	
id	pos
1	
2	5,6,8
3	7
4	5,7,8

TA	
id	pos
1	4,6,7,9,10
2	3,9
3	10,11
4	11,12

TG	
id	pos
1	8
2	5,6,8
3	6,7
4	7,8

TT	
id	pos
1	
2	4,7,10
3	
4	9

id	pos
1	1,2,4,6,7,9,10
2	3,9
3	1,2,4,10,11
4	1,2,11,12

id	pos
1	5,11
2	
3	3,8,9
4	4,10

G	
id	pos
1	8
2	1,5,6,8
3	6,7
4	3,5,7,8

T	
id	pos
1	3
2	2,4,7,10
3	5
4	6,9

3 length sequences

AGA	
id	pos
1	9,10
2	9
3	10,11
4	11,12

ATA	
id	pos
1	4,6,7,9,10
2	9
3	10,11
4	11,12

AAA	
id	pos
1	4,6,9,10
2	
3	4,10,11
4	11,12

AGG	
GG invalid	

ATG	
id	pos
1	8
2	5,6,8
3	6,7
4	7,8

AAG	
id	pos
1	8
2	
3	6,7
4	3,5,7,8

AGT	
GT invalid	

ATT	
TT invalid	

AAT	
id	pos
1	3
2	10
3	5
4	6,9

GAG	
GG invalid	

GAT	GT invalid

GAA	
id	pos
1	10
2	9
3	11
4	12

TAA	
id	pos
1	6,7,9,10
2	9
3	11
4	12

TAG	
id	pos
1	8
2	5,6,8
3	
4	

TAT	

TGA	

TGG	

TGT	

id	pos
1	9,10
2	9
3	10,11
4	11,12

4 length sequences

AGAA	
AAA invalid	

AGAT	
GT invalid	

AGAG	
GG invalid	

ATGA	
id	pos
1	9,10
2	9
3	7
4	8

ATGG	
G invalid	

ATGT	
GT invalid	

ATAA	
AAA invalid	

ATAG	
TAG invalid	

ATAT	
TT invalid	

GAAA	
AAA invalid	

GAAG	
GG invalid	

GAAT	
GT invalid	

AATA	
AAA invalid	

AATG	
AAG invalid	

AATT	
TT invalid	

TGAA	
id	pos
1	10
2	
3	11
4	12

TGAG	
GG invalid	

TGAT	
TT invalid	

TAAA	
AAA invalid	

TAAG	
AAG invalid	

TAAT	
TT invalid	

5 length sequences

ATGAA	
AAA invalid	

ATGAG	
GG invalid	

ATGAT	
TT invalid	

ALL FREQUENT SEQUENCES: {A, C, G, T, AG, AT, AA, GA, TA, TG, ATA, AGA, ATG, AAT, GAA, TAA, TGA, ATGA}

b) 4 items in the universe. Thus there can be 4^k possible sequences of length K. This is due to the fact that every element in the sequence can be 1 of 4 items contained in the universe

10.5)

Q5. Consider the database shown in Table 10.4. Each sequence comprises itemset events that happen at the same time. For example, sequence $\mathbf{s}_1$ can be considered to be a sequence of itemsets $(AB)_{10}(B)_{20}(AB)_{30}(AC)_{40}$, where symbols within brackets are considered to co-occur at the same time, which is given in the subscripts. Describe an algorithm that can mine all the frequent subsequences over itemset events. The

Table 10.4. Sequences for Q5

Id	Time	Items

The spade algorithm can find all frequent itemsets. It must also take into account that items that occur simultaneously can be interchangeable. Below is the transformed item set to something more manageable(empty parenthesis means nothing happened at that time, and can essentially be ignored)

s1 = (a,b) b (a,b) (a,c) () ()

s2 = () (a,c) (a,b,c) () b ()

considered to co-occur at the same time, which is given in the subscripts. Describe an algorithm that can mine all the frequent subsequences over itemset events. The

Table 10.4. Sequences for Q5

Id	Time	Items
s_1	10	A, B
	20	B
	30	A, B
	40	A, C
s_2	20	A, C
	30	A, B, C
	50	B
s_3	10	A
	30	B
	40	A
	50	C
	60	B
s_4	30	A, B
	40	A
	50	B
	60	C

itemsets can be of any length as long as they are frequent. Find all frequent itemset sequences with $minsup = 3$.

transformed item set to something more manageable(empty parenthesis means nothing happened at that time, and can essentially be ignored)

s1 = (a,b) b (a,b) (a,c) () ()
s2 = () (a,c) (a,b,c) () b ()
s3 = a () b a c b ()
s4 () () (a,b)a b c

A	
id	pos
1	1,3,4
2	2,3
3	1,4
4	3,4

B	
id	pos
1	1,2,3
2	3,5
3	3,6
4	3,5

C	
id	pos
1	4
2	2,3
3	5
4	6

AA	
id	pos
1	3,4
2	3
3	4
4	4

AB	
id	pos
1	2,3
2	3,5
3	3,6
4	5

AC	
id	pos
1	4
2	3
3	5
4	6

BA	
id	pos
1	3,4
2	
3	4
4	4

BB	
id	pos
1	2,3
2	5
3	6
4	5

BC	
id	pos
1	4
2	
3	5
4	6

CA	
id	pos
1	
2	3
3	
4	

CB	
id	pos
1	
2	3,5
3	6
4	

CC	
id	pos
1	
2	3
3	
4	

AAA	
id	pos
1	4
2	
3	
4	

AAB	
id	pos
1	
2	5
3	6
4	5

AAC	
id	pos
1	
2	
3	5
4	6

ABA	
id	pos
1	3,4
2	
3	4
4	

ABB	
id	pos
1	3
2	5
3	6
4	

ABC	
id	pos
1	4
2	
3	5
4	6

ACA	
CA invalid	

ACB	
CBinvalid	

ACC	
CC invalid	

BAA	
id	pos
1	4
2	
3	
4	

BAB	
id	pos
1	
2	
3	6
4	5

BAC	
id	pos
1	4
2	
3	5
4	6

BBA	
id	pos
1	3,4
2	
3	
4	

BBB	
id	pos
1	3
2	
3	
4	

BBC	
id	pos
1	4
2	
3	
4	

BCA	
CA invalid	

BCB	
CB invalid	

BCC	
CC invalid	

AABA	
AAA invalid	

AABB	
id	pos
1	
2	
3	
4	

AABC	
AAC invalid	

ABBA	
BBAinvalid	

ABBB	
BBB invalid	

ABBC	
BBC invalid	

ABCA	
CA invalid	

ABCB	
CB invalid	

ABCC	
CC invalid	

BACA	
CA invalid	

BACB	
CB invalid	

BACB	
CC invalid	

ALL FREQUENT SEQUENCES: {A, B, C, AA, AB, AC, BA, BB, BC, AAB, ABB, ABC, BAC}

4.1)

Q1. Given the graph in Figure 4.15, find the fixed-point of the prestige vector.

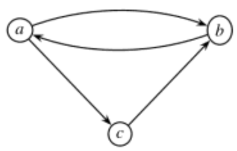

Figure 4.15. Graph for Q1

adjacency list
a: 0 1 1
b: 1 0 0
c: 0 1 0

adjacency list transposed = A
0 1 0
1 0 1
1 0 0

Prestige (PageRank without random surfer)

let p_0^T = (1 1 1). Use transposed adjacency list A

first iteration:
p = Ap_0 =
0 1 0   1   1   .5
1 0 1   1   2   1
1 0 0 * 1 = 1 normalized = .5 (norm val=2)

second iteration
0 1 0   .5   1   1
1 0 1   1   1   1
1 0 0 * .5 = .5 normalized = .5 (norm val = 1)

third iteration
0 1 0   1   1   .66
1 0 1   1   1.5   1
1 0 0 * .5 = 1 normalized = .33 (norm val = 1.5)

fourth iteration
0 1 0   .66   1   1
1 0 1   1   1   1
1 0 0 * .33 = .66 normalized = .66 (norm val = 1)

5th iteration
0 1 0   1   1   .60
1 0 1   1   1.66   1
1 0 0 * .66 = 1 normalized = .4 (norm val = 1.66)
Converge!

Page rank without random surfer:

probability adjacency list transposed = A
0 1 0
.5 0 .5
1 0 0

let p_0^T = (0.33 0.33 0.33). Use transposed adjacency list A

first iteration:
p = Ap_0 =
0 1 0   .33   .33   .5
.5 0 .5   .33   .33   1
1 0 0 * .33 = .33 normalized = .5 (norm val=2)

second iteration:
0 1 0   .33   .33
.5 0 .5   .33   .33
1 0 0 * .33 = .33 Converge!

3.3.1.a)

Exercise 3.3.1 : Verify the theorem from Section 3.3.3, which relates the Jaccard similarity to the probability of minhashing to equal values, for the particular case of Fig. 3.2.

(a) Compute the Jaccard similarity of each of the pairs of columns in Fig. 3.2.

Element	S_1	S_2	S_3	S_4
a	1	0	0	1
b	0	0	1	0
c	0	1	0	1
d	1	0	1	1
e	0	0	1	0

Figure 3.2: A matrix representing four sets

a) Jaccard similarity = |A intersection B| / |A union B|

J(AB) = 0 / 3 = 0
J(AC) = 1 / 3 = 0.3333
J(AD) = 2 / 3 = 0.6667
J(AE) = 0 / 3 = 0
J(BC) = 0 / 3 = 0
J(BD) = 1 / 3 = 0.3333
J(BE) = 1 / 1 = 1
J(CD) = 1 / 4 = 0.25
J(CE) = 0 / 3 = 0
J(DE) = 1 / 3 = 0.3333

3.3.2)

Exercise 3.3.2 : Using the data from Fig. 3.4, add to the signatures of the columns the values of the following hash functions:

(a) $h_3(x) = 2x + 4 \mod 5$.

(b) $h_4(x) = 3x - 1 \mod 5$.

Row	S_1	S_2	S_3	S_4	$x+1 \mod 5$	$3x+1 \mod 5$
0	1	0	0	1	1	1
1	0	0	1	0	2	4
2	0	1	0	1	3	2
3	1	0	1	1	4	0
4	0	0	1	0	0	3

Figure 3.4: Hash functions computed for the matrix of Fig. 3.2

row	2x+4 mod 5	3x-1mod 5
0	4	4
1	1	2
2	3	0
3	0	3
4	2	1