Data Mining Package TAKE() Documentation

version: 1.2,2.0

Revision History:
October 6, 2014 : added mclque2g.rb,mbiclique.rb; changed specification of edge file and node file.
March 10, 2014 : changes in installation procedure with integration into nysol package
March 10, 2014 : improved mitemset.rb, and added msequence.rb,mpolishing.rb
February 28, 2014 : first release

April 30, 2015

Contents

Chapter 1

Introduction

1.1 Summary

This package "TAKE (bamboo)" is developed at the centre led by Professor Takeaki Uno (Associate professor of National Institute of Informatics). This is a user-friendly group of commands as an extension of data mining software [3]. The package "TAKE" is therefore named after the developer. Many of the commands in this package is based on the pattern enumeration with a variety of target data such as an itemset, series, and general graph.

For example, the products in supermarket shopping baskets are treated as a set of items, the command is capable of enumerating combination of all products common in many shopping baskets at a high speed. In addition, with the concept of class is introduced, the specific patterns can be enumerated to describe a group of target / favourable customers.

Knowledge of popular sequential pattern from circular logging of web pages among most users provide useful insight to the web page structure. The concept of a class is can be used in conjunction with enumeration of sequential patterns. As a result, different traffic patterns from circular logging such as patterns specific to men and women can be enumerated.

In addition, the package includes commands for processing general graph based data. Some possible usages include analysis of companies' trading data network, user data network in SNS, and similarity graph that represents the resemblance among items. For example, similarity graph can be constructed given the co-occurrence information of the merchandises. In addition, it is possible to extract merchandises with strong relationships by enumerating the maximal cliques of the graph, these extracted clusters can be used as explanatory variables in the product purchase model.

The data polishing method can be used in pre-processing stage to suppress massive number of maximal cliques, which in turn enumerates fewer medium sized maximal cliques.

All commands in this package is written in Ruby language. Internally, the native commands are executed through shell interface, where data is exchanged with native command in basic file format.

1.1.1 Installation

This package is part of the complete NYSOL package. Thus, all software within the NYSOL package should be installed as prerequisite. For details on the installation of NYSOL package, please refer to the URL: http://www.nysol.jp/install

1.1.2 License

This package comprised of source code of the command developed by Professor Uno, please refer to the `readme.txt` file in the archive on licensing details [3].

Other software is distributed through GNU AGPL(AFFERO GENERAL PUBLIC LICENSE: http://www.gnu.org/licenses/agpl-3.0.html) for public usage.

Chapter 2

Commands

2.1 mitemset.rb - Enumerate Frequent Itemsets

This program enumerates frequent itemsets from the transaction data using LCM (Linear time Closed itemset Miner) as the core algorithm for itemset enumeration[1, 3].

This command have the following features:

- Enumerate "maximal itemsets" that is not contained in other itemsets.
- Enumerate "emerging itemsets" that contains the maximal itemsets with similar features.
- Use item taxonomy for hierarchical classification.
- By specifying classification class, patterns with specific features particular to the class is enumerated (emerging patterns). Supports 3 or more classes.

The different types of input data are shown in Table 2.1,2.2,2.3, however, note that this command can only process key-based data (Table 2.1). The `mtra` and `mtraflg` in MCMD package can be used to convert other data types to key-based data beforehand.

Table 2.1 contains records with the same `key` field, on the other hand, Table 2.2,2.3 display transaction containing relevant items in one record.

In case of supermarket, consider items as merchandise purchased for each transaction receipt.

Table 2.1: Key based data

key	item
T1	C
T1	E
T2	D
T2	E
T2	F
:	:

Table 2.2: Tra type data

id	item
T1	C E
T2	D E F
T3	A B D F
T4	B D F
T5	A B D E
T6	A B D E F

Table 2.3: Row type data

id	A	B	C	D	E	F
T1			1		1	
T2				1	1	1
T3	1	1		1		1
T4		1		1		1
T5	1	1		1	1	
T6	1	1		1	1	1

Frequent itemset

A Frequent itemset refers to a set of items of which the frequency (known as support) is more than or equal to the minimum support provided by the user.

Using the input data in Table 2.2 as an example, when the minimum support is set as 3, itemset {B,D,F} appeared in T3,T5,T6 and is considered as frequent. However, itemset {B,D,E} only appeared in T5,T6 and it is not considered.

There are a total of 13 frequent itemsets meeting the minimum support of 3 including {A}, {A,B}, {A,B,D}, {A,D}, {B}, {B,D}, {B,D,F}, {B,F}, {D}, {D,E}, {D,F}, {E}, {F}.

Since a large number of frequent itemset candidates are enumerated, two enumeration methods namely maximum itemset and closed itemset can be used to select representative itemsets in the output.

Maximal itemset

A maximal itemset is a frequent itemset which is included in no other frequent itemsets. In Table 2.2, three itemsets {A,B,D},{B,D,F},{D,E} are not included in any other itemsets. Therefore, it is referred to as maximal item set. Other itemsets are not maximal since they are included in more than 3 maximal itemsets.

Closed itemset

Select any two frequent itemsets, if they appear in the same transaction, they are considered as the same group of itemset. By grouping all frequent itemsets, if there are no superset that has the same support as the frequent itemset, the itemset is referred to as closed set. For example, {A},{A,B},{A,D},{A,B,D} appear in transactions T3,T5,T6 and they are classified as the same group.

{A,B,D} is returned as closed itemset with maximum frequency, and itemsets {A},{A,B},{A,D} are not included in output.

Table 2.4 a total of seven patterns are enumerated from closed itemsets. The number of itemsets enumerated are reduced significantly as closed itemsets only enumerate representative itemsets.

Table 2.4: Closed itemset

Closed set	Transaction	Group
{A,B,D}	T3,T5,T6	{A},{A,B},{A,D},{A,B,D}
{B,D}	T3,T4,T5,T6	{B},{B,D}
{B,D,F}	T3,T4,T6	{B,F},{B,D,F}
{D}	T2,T3,T4,T5,T6	{D}
{D,E}	T2,T5,T6	{D,E}
{D,F}	T2,T3,T4,T6	{F},{D,F}
{E}	T1,T2,T5,T6	{E}

Emerging patterns

Emerging patterns enumerate particular patterns (frequent itemset) from pre-defined class that it is frequent for one data class and not frequent for another class, and whose support changes significantly. The feature characteristics in one class is not frequent in other classes. For instance, it can be used to identify different items purchased by men or women in a supermarket.

Please refer to Appendix 1 for more detailed definition of emerging pattern. The class data shown in 2.5 is created by combining the class item to the transaction data. Rows with the same transaction key will be assigned with the same class value.

Table 2.5: Key-based data

key	item	class
T1	C	pos
T1	E	pos
T2	D	neg
T2	E	neg
T2	F	neg
:	:	

Hierarchical classification

Hierarchical classification can be designated for individual items. For example, in a supermarket, an item "Milk 500ml" is classified as "Milk", subsequently, "Milk" is classified under "Milk product", and "Milk product" is classified as "Food". By using hierarchical classification, it is possible to find out whether "Milk 500ml" and "Fruit" are frequently purchased at the same time. Note that the current version only allows for one hierarchy.

The internal processing can be simplified as follows. Given the corresponding relationship between item and classification (Table 2.7), corresponding classifications are combined with certain items in the input data (Table 2.6) as shown in Table 2.8. Alternatively, the classifications substitute the corresponding itemsets in the input data as shown in Table 2.9.

Output

This command returns two main output data, the first is the enumerated pattern data (`patterns.csv`), the other contains information about the transaction for the corresponding pattern (`tid_pats.csv`). CSV columns in output pattern data are different for emerging pattern. The sample is shown in Table 2.10 to Table 2.12.

Table 2.6: Original data

id	item
T1	C E
T2	D E F
T3	A B D F
T4	B D F
T5	A B D E
T6	A B D E F

Table 2.7: Item-classification mapping table

item	taxonomy
A	X
B	X
C	Y
D	Z
E	Z
F	Z

Table 2.8: Data combined with classification

id	item
T1	C E Y Z
T2	D E F Z
T3	A B D F X Z
T4	B D F Y Z
T5	A B D E X Z
T6	A B D E F X Z

Table 2.9: Data substituted with classification

id	item
T1	Y Z
T2	Z
T3	X Z
T4	Y Z
T5	X Z
T6	X Z

Table 2.10: Example of patterns.csv data. The column pid contains the unique ID which differentiates each pattern, size refers to the number of items consists in the pattern, count refers to the number of transactions the pattern appears, total refers to the number of all transactions. Support of probability of occurrence is calculated by count/total. Lift compares the expected probability with actual probability to measure the performance of target model. The last column contains pattern of itemset, the items are delimited by space.

pid	size	count	total	support	lift	pattern
1	1	5	6	0.8333333333	1	D
7	2	4	6	0.6666666667	1.2	D F
6	1	4	6	0.6666666667	1	F
4	1	4	6	0.6666666667	1	E
2	1	4	6	0.6666666667	1	B
3	2	4	6	0.6666666667	1.2	B D
8	2	3	6	0.5	1.125	B F
13	2	3	6	0.5	1.2	A D

Value of lift lift(I) for itemset $I = \{i_1, i_2, \cdots, i_n\}$ is defined as follows.

$$\text{lift}(I) = \frac{\Pr(I)}{\prod_{k=1}^{n} \Pr(i_k)}$$

Table 2.11: Contents of tid-pats.csv. Tid is the transaction ID, which corresponds to the column in the input data defined at tid= parameter. The ID of each pattern for each transaction is identified by pid.

tid	pid
T1	4
T2	1
T2	4
T2	7
T2	6
T2	5
T3	10
T3	6

Table 2.12: Example of emerging patterns in patterns.csv. The class field indicates the target class based on the characteristics shown in emerging patterns. Attributes pid, pattern, size, total shown in Table 2.10 are defined in the previous table. Pos refers to the number of target class appeared in the transaction, neg is the number of other classes in the transaction. The total transaction numbers of target and and non target classes are indicated in posTotal and negTotal respectively. Support is the probability of occurrence, calculated by pos/posTotal. The change is represented by growthRate, calculated by support/(neg/negTotal). The result is shown as inf when the denominator is 0. As this value increases, the key feature for target class emerges. Posterior probability of the target class is represented by postProb, as with growthRate, as the value grow larger, it shows the key feature for the target class. Detailed definition is illustrated in Appendix 1 in mitemset.rb command manual.

class	pid	pattern	size	pos	neg	posTotal	negTotal	total	support	growthRate	postProb
cls2	13	A E	2	2	0	2	4	6	1	inf	1
cls2	15	A B E	3	2	0	2	4	6	1	inf	1
cls2	10	A B D E	4	2	0	2	4	6	1	inf	1
cls2	14	B E	2	2	0	2	4	6	1	inf	1
cls2	17	A D E	3	2	0	2	4	6	1	inf	1
cls2	18	B D E	3	2	0	2	4	6	1	inf	1
cls2	12	A B D	3	2	1	2	4	6	1	4	0.6666666667
cls2	11	A D	2	2	1	2	4	6	1	4	0.6666666667
cls2	16	D E	2	2	1	2	4	6	1	4	0.6666666667

Format

```
mitemset.rb i= [x=] [O=] [tid=] [item=] [class=] [taxo=] [s=|S=] [sx=|SX=] [l=] [u=]
            [p=] [g=] [top=] [T=] [--help]

  i=       File name of key type transaction data [required parameter]
  x=       File name of hierarchical classification data [optional parameter]
  O=       Output path name  [optional: default=./take_#{DateTime}]
  tid=     Field name of transaction ID [required parameter]
  item=    Time based field name (field name in i=) [optional: default="time"]
  class=   Field name of class (field name in c=) [optional parameter]
           Emerging patterns is enumerated based on the class field defined.
  taxo=    Field name of taxonomy [required parameter with conditions: x=]
  s=       Minimum support (probability) [select either one parameter: s=, S=]
  S=       Minimum support (hits)  [select either one parameter: s=, S=]
  sx=      Maximum support (probability) [optional parameter]
  SX=      Maximum support (hits)  [optional parameter]
  l=       Minimum itemset size [optional parameter]
  u=       Maximum itemset size [optional parameter]
  p=       Minimum posterior probability for emerging patterns.   [optional: default=0.5]
  g=       Minimum growth rate for emerging patterns  [optional parameter]
  top=     Upper limit of number of patterns to enumerate [optional: default: without limit]
   T=       Working directory [optional parameter]
  --help   Show help information
```

2.1.1 Examples

Example 1: Basic Example

Enumerate frequent itemset that appear more than 3 times.

```
$ more dat1.csv
tid,item
T1,C
T1,E
T2,D
T2,E
T2,F
T3,A
T3,B
T3,D
T3,F
T4,B
T4,D
T4,F
T5,A
T5,B
T5,D
T5,E
T6,A
T6,B
T6,D
T6,E
T6,F
$ mitemset.rb  S=3 tid=tid item=item i=dat1.csv O=result1
#MSG# lcm_20140215 FIf /tmp/__MTEMP_18926_70182748393500_0 3 /tmp/__MTEMP_18926_7018274838
9080_0
trsact: /tmp/__MTEMP_18926_70182748393500_0 ,#transactions 6 ,#items 6 ,size 21 extracted
database: #transactions 6 ,#items 5 ,size 20
 output to: /tmp/__MTEMP_18926_70182748389080_0
separated at 0
iters=8
14
1
5
6
2
```

```
trsact: /tmp/__MTEMP_18926_70182748393500_0 ,#transactions 6 ,#items 6 ,size 21 extracted
database: #transactions 6 ,#items 6 ,size 21
 output to: /tmp/__MTEMP_18926_70182748389080_1
separated at 0
iters=7
6
0
6
#MSG# output patterns to CSV file ...
#MSG# the number of patterns enumerated is 13
#MSG# output tid-patterns ...
#MSG# The final results are in the directory ‘result1’
#END# /Users/stephane/.rvm/rubies/ruby-1.9.3-p448/bin/mitemset.rb S=3 tid=tid item=item i=
dat1.csv O=result1
$ more result1/patterns.csv
pid,size,count,total,support,lift,pattern
1,1,5,6,0.8333333333,1,D
7,2,4,6,0.6666666667,1.2,D F
6,1,4,6,0.6666666667,1,F
4,1,4,6,0.6666666667,1,E
2,1,4,6,0.6666666667,1,B
3,2,4,6,0.6666666667,1.2,B D
8,2,3,6,0.5,1.125,B F
13,2,3,6,0.5,1.2,A D
5,2,3,6,0.5,0.9,D E
12,3,3,6,0.5,1.8,A B D
11,2,3,6,0.5,1.5,A B
10,1,3,6,0.5,1,A
9,3,3,6,0.5,1.35,B D F
$ more result1/tid_pats.csv
tid,pid
T1,4
T2,1
T2,4
T2,7
T2,6
T2,5
T3,10
T3,6
T3,13
T3,7
T3,11
T3,8
T3,3
T3,12
T3,1
T3,2
T3,9
T4,6
T4,7
T4,8
T4,2
T4,1
T4,3
T4,9
T5,11
T5,13
T5,3
T5,1
T5,4
T5,10
T5,5
T5,2
T5,12
T6,2
T6,11
T6,6
T6,7
T6,5
T6,10
T6,1
T6,8
T6,12
T6,4
T6,9
T6,13
T6,3
```

Example 2: Set a limit on the size of itemset

For itemsets that appear more than 3 or more times, patterns of itemsets with size 3 is enumerated.

```
$ mitemset.rb S=3 l=3 u=3 tid=tid item=item i=dat1.csv O=result2
#MSG# lcm_20140215 FIf -l 3 -u 3 /tmp/__MTEMP_19010_70360847449800_0 3 /tmp/__MTEMP_19010_
70360847467580_0
trsact: /tmp/__MTEMP_19010_70360847449800_0 ,#transactions 6 ,#items 6 ,size 21 extracted
database: #transactions 6 ,#items 5 ,size 20
 output to: /tmp/__MTEMP_19010_70360847467580_0
separated at 0
iters=8
2
0
0
0
2
trsact: /tmp/__MTEMP_19010_70360847449800_0 ,#transactions 6 ,#items 6 ,size 21 extracted
database: #transactions 6 ,#items 6 ,size 21
 output to: /tmp/__MTEMP_19010_70360847467580_1
separated at 0
iters=7
6
0
6
#MSG# output patterns to CSV file ...
#MSG# the number of patterns enumerated is 2
#MSG# output tid-patterns ...
#MSG# The final results are in the directory ‘result2’
#END# /Users/stephane/.rvm/rubies/ruby-1.9.3-p448/bin/mitemset.rb S=3 l=3 u=3 tid=tid item
=item i=dat1.csv O=result2
$ more result2/patterns.csv
pid,size,count,total,support,lift,pattern
0,3,3,6,0.5,1.35,B D F
1,3,3,6,0.5,1.8,A B D
```

Example 3: Enumerate closed itemsets

```
$ mitemset.rb S=3 type=C tid=tid item=item i=dat1.csv O=result3
#MSG# lcm_20140215 CIf /tmp/__MTEMP_19093_70290667606020_0 3 /tmp/__MTEMP_19093_7029066760
1800_0
trsact: /tmp/__MTEMP_19093_70290667606020_0 ,#transactions 6 ,#items 6 ,size 21 extracted
database: #transactions 6 ,#items 5 ,size 20
 output to: /tmp/__MTEMP_19093_70290667601800_0
separated at 0
iters=8
8
1
2
3
2
trsact: /tmp/__MTEMP_19093_70290667606020_0 ,#transactions 6 ,#items 6 ,size 21 extracted
database: #transactions 6 ,#items 6 ,size 21
 output to: /tmp/__MTEMP_19093_70290667601800_1
separated at 0
iters=7
6
0
6
#MSG# output patterns to CSV file ...
#MSG# the number of patterns enumerated is 7
#MSG# output tid-patterns ...
#MSG# The final results are in the directory ‘result3’
#END# /Users/stephane/.rvm/rubies/ruby-1.9.3-p448/bin/mitemset.rb S=3 type=C tid=tid item=
item i=dat1.csv O=result3
$ more result3/patterns.csv
pid,size,count,total,support,lift,pattern
1,1,5,6,0.8333333333,1,D
2,2,4,6,0.6666666667,1.2,B D
3,1,4,6,0.6666666667,1,E
5,2,4,6,0.6666666667,1.2,D F
4,2,3,6,0.5,0.9,D E
6,3,3,6,0.5,1.35,B D F
7,3,3,6,0.5,1.8,A B D
```

Example 4: Enumerate maximal itemsets

```
$ mitemset.rb S=3 type=M tid=tid item=item i=dat1.csv O=result4
#MSG# lcm_20140215 MIf /tmp/__MTEMP_19176_70274959882120_0 3 /tmp/__MTEMP_19176_7027495987
7480_0
trsact: /tmp/__MTEMP_19176_70274959882120_0 ,#transactions 6 ,#items 6 ,size 21 extracted
database: #transactions 6 ,#items 5 ,size 20
 output to: /tmp/__MTEMP_19176_70274959877480_0
separated at 0
iters=8
3
0
0
1
2
trsact: /tmp/__MTEMP_19176_70274959882120_0 ,#transactions 6 ,#items 6 ,size 21 extracted
database: #transactions 6 ,#items 6 ,size 21
 output to: /tmp/__MTEMP_19176_70274959877480_1
separated at 0
iters=7
6
0
6
#MSG# output patterns to CSV file ...
#MSG# the number of patterns enumerated is 3
#MSG# output tid-patterns ...
#MSG# The final results are in the directory 'result4'
#END# /Users/stephane/.rvm/rubies/ruby-1.9.3-p448/bin/mitemset.rb S=3 type=M tid=tid item=
item i=dat1.csv O=result4
$ more result4/patterns.csv
pid,size,count,total,support,lift,pattern
0,2,3,6,0.5,0.9,D E
1,3,3,6,0.5,1.35,B D F
2,3,3,6,0.5,1.8,A B D
```

Example 5: Usage of hierarchical classification

```
$ more taxo.csv
item,taxonomy
A,X
B,X
C,Y
D,Z
E,Z
F,Z
$ mitemset.rb S=4 tid=tid item=item i=dat1.csv x=taxo.csv taxo=taxonomy O=result5
#MSG# lcm_20140215 FIf /tmp/__MTEMP_19260_70240579278840_0 4 /tmp/__MTEMP_19260_7024057927
5400_0
trsact: /tmp/__MTEMP_19260_70240579278840_0 ,#transactions 6 ,#items 10 ,size 32 extracted
 database: #transactions 6 ,#items 6 ,size 27
 output to: /tmp/__MTEMP_19260_70240579275400_0
separated at 0
iters=6
22
1
6
9
5
1
trsact: /tmp/__MTEMP_19260_70240579278840_0 ,#transactions 6 ,#items 10 ,size 32 extracted
 database: #transactions 6 ,#items 9 ,size 32
 output to: /tmp/__MTEMP_19260_70240579275400_1
separated at 0
iters=9
9
0
9
#MSG# output patterns to CSV file ...
#MSG# reducing redundant rules in terms of taxonomy ...
#MSG# the number of patterns enumerated is 11
#MSG# output tid-patterns ...
#MSG# The final results are in the directory 'result5'
#END# /Users/stephane/.rvm/rubies/ruby-1.9.3-p448/bin/mitemset.rb S=4 tid=tid item=item i=
dat1.csv x=taxo.csv taxo=taxonomy O=result5
$ more result5/patterns.csv
pid,size,count,total,support,lift,pattern
```

```
1,1,6,6,1,1,Z
2,1,5,6,0.8333333333,1,D
19,2,4,6,0.6666666667,1.2,D X
13,2,4,6,0.6666666667,1,B Z
14,1,4,6,0.6666666667,1,X
6,1,4,6,0.6666666667,1,F
11,2,4,6,0.6666666667,1.2,B D
21,2,4,6,0.6666666667,1,X Z
4,1,4,6,0.6666666667,1,E
10,1,4,6,0.6666666667,1,B
7,2,4,6,0.6666666667,1.2,D F
```

Example 6: Replace original items with hierarchical classification

```
$ more taxo.csv
item,taxonomy
A,X
B,X
C,Y
D,Z
E,Z
F,Z
$ mitemset.rb S=4 tid=tid item=item i=dat1.csv x=taxo.csv taxo=taxonomy -replaceTaxo O=resul
t6
#MSG# lcm_20140215 FIf /tmp/__MTEMP_19394_70212028633240_0 4 /tmp/__MTEMP_19394_7021202864
5160_0
trsact: /tmp/__MTEMP_19394_70212028633240_0 ,#transactions 6 ,#items 4 ,size 11 extracted
database: #transactions 6 ,#items 2 ,size 10
 output to: /tmp/__MTEMP_19394_70212028645160_0
separated at 0
iters=2
4
1
2
1
trsact: /tmp/__MTEMP_19394_70212028633240_0 ,#transactions 6 ,#items 4 ,size 11 extracted
database: #transactions 6 ,#items 3 ,size 11
 output to: /tmp/__MTEMP_19394_70212028645160_1
separated at 0
iters=3
3
0
3
#MSG# output patterns to CSV file ...
#MSG# the number of patterns enumerated is 3
#MSG# output tid-patterns ...
#MSG# The final results are in the directory ‘result6’
#END# /Users/stephane/.rvm/rubies/ruby-1.9.3-p448/bin/mitemset.rb S=4 tid=tid item=item i=
dat1.csv x=taxo.csv taxo=taxonomy -replaceTaxo O=result6
$ more result6/patterns.csv
pid,size,count,total,support,lift,pattern
1,1,6,6,1,1,Z
2,1,4,6,0.6666666667,1,X
3,2,4,6,0.6666666667,1,X Z
```

Example 7: Enumerate emerging patterns

```
$ more dat2.csv
tid,item,class
T1,C,cls1
T1,E,cls1
T2,D,cls1
T2,E,cls1
T2,F,cls1
T3,A,cls1
T3,B,cls1
T3,D,cls1
T3,F,cls1
T4,B,cls1
T4,D,cls1
T4,F,cls1
T5,A,cls2
T5,B,cls2
T5,D,cls2
```

```
T5,E,cls2
T6,A,cls2
T6,B,cls2
T6,D,cls2
T6,E,cls2
T6,F,cls2
$ mitemset.rb S=2 tid=tid item=item class=class i=dat2.csv p=0.6 O=result7
#MSG# lcm_20140215 FIA -w /tmp/__MTEMP_19510_70322251041060_1 /tmp/__MTEMP_19510_703222510
41060_0 1073741817 /tmp/__MTEMP_19510_70322251051500_0
trsact: /tmp/__MTEMP_19510_70322251041060_0 ,#transactions 6 ,#items 6 ,size 21 extracted
database: #transactions 6 ,#items 4 ,size 17 ,weightfile /tmp/__MTEMP_19510_70322251041060
_1
 output to: /tmp/__MTEMP_19510_70322251051500_0
separated at 0
iters=6
9
1
4
3
1
#MSG# output patterns to CSV file ...
#MSG# the number of contrast patterns on class 'cls1' enumerated is 8
#MSG# output tid-patterns ...
#MSG# lcm_20140215 FIA -w /tmp/__MTEMP_19510_70322251041060_2 /tmp/__MTEMP_19510_703222510
41060_0 2147483645 /tmp/__MTEMP_19510_70322251051500_2
trsact: /tmp/__MTEMP_19510_70322251041060_0 ,#transactions 6 ,#items 6 ,size 21 extracted
database: #transactions 6 ,#items 4 ,size 16 ,weightfile /tmp/__MTEMP_19510_70322251041060
_2
 output to: /tmp/__MTEMP_19510_70322251051500_2
separated at 0
iters=14
11
0
1
5
4
1
#MSG# output patterns to CSV file ...
#MSG# the number of contrast patterns on class 'cls2' enumerated is 11
#MSG# output tid-patterns ...
#MSG# the number of emerging patterns enumerated is 11
#MSG# The final results are in the directory 'result7'
#END# /Users/stephane/.rvm/rubies/ruby-1.9.3-p448/bin/mitemset.rb S=2 tid=tid item=item cl
ass=class i=dat2.csv p=0.6 O=result7
$ more result7/patterns.csv
class,pid,pattern,size,pos,neg,posTotal,negTotal,total,support,growthRate,postProb
cls2,13,A E,2,2,0,2,4,6,1,inf,1
cls2,15,A B E,3,2,0,2,4,6,1,inf,1
cls2,10,A B D E,4,2,0,2,4,6,1,inf,1
cls2,14,B E,2,2,0,2,4,6,1,inf,1
cls2,17,A D E,3,2,0,2,4,6,1,inf,1
cls2,18,B D E,3,2,0,2,4,6,1,inf,1
cls2,12,A B D,3,2,1,2,4,6,1,4,0.6666666667
cls2,11,A D,2,2,1,2,4,6,1,4,0.6666666667
cls2,16,D E,2,2,1,2,4,6,1,4,0.6666666667
cls2,9,A B,2,2,1,2,4,6,1,4,0.6666666667
cls2,8,A,1,2,1,2,4,6,1,4,0.6666666667
```

Appendix 1: parameter gr=,post=,-uniform

Given class set $C = \{c_1, c_2, \cdots, c_m\}$, each transaction corresponds to one class. The emerging patterns of interest that appears frequently in a certain target class will not be frequent patterns in other classes. For example, The itemset appears frequently in the class of c_1, will not appear frequently in the class $c_2, c_3, \cdots, c_m$. In the following, the target class is set as c_t, and other classes are set as c_o.

There are three different ways to define emerging patterns in this command.

1. Specify the threshold of the growth rate (Ratio of the emerging patterns probability among classes)
2. Specify the threshold of posterior probability (Prior probability is estimated from the distribution of data)
3. Specify the threshold of posterior probability (Prior probability is uniform across all classes)

1.Growth rate

The growth rate $GR_t(I)$ of itemset I in the target class c_t is represented in format (2.1), it is defined as the ratio of probability of occurrence of a set of items in the target class against other classes.

In addition, emerging pattern refers to itemsets with growth rate more than the minimum growth rate γ specified by the user. The γ is specified by the parameter `gr=`.

$$GR_t(I) = \frac{\Pr(I|c_t)}{\Pr(I|c_o)} \geq \gamma \tag{2.1}$$

2. Posterior probability

For transactions with unknown class, given observing itemset I, the probability that the transaction belongs to class c_t is represented by the formula (2.2) according to the Bayes' theorem.

The formula updates the posterior probability $\Pr(c_t|I)$ by which the prior probability $\Pr(c_t)$ of class c_t by observing itemset I. The prior probability $\Pr(c_t)$ is estimated based on the class distribution in the given data. Here, the emerging pattern refers to itemsets with a posterior probability more than the minimum posterior probability π specified by the user. π is specified at the parameter `post=`.

$$\Pr(c_t|I) = \frac{\Pr(I|c_t)\Pr(c_t)}{\Pr(I|c_t)\Pr(c_t) + \Pr(I|c_o)\Pr(c_o)} \geq \pi \tag{2.2}$$

3. Posterior probability (Uniform prior probability)

Calculate the posterior probability assuming the prior probability of all the class is uniform. The equation (2.3) is obtained by substituting (2.2) into the equation $\Pr(c_t) = \frac{1}{m}$,$\Pr(c_o) = \frac{m-1}{m}$.

Further, assuming the prior probability is uniform, emerging pattern refers to the itemset that has a posterior probability more than the minimum posterior probability π_u specified by users. π_u is specified at the `post=` parameter and the `-uniform` option.

$$\Pr(c_t|I) = \frac{\Pr(I|c_t)}{\Pr(I|c_t) + (m-1)\Pr(I|c_o)} \geq \pi_u \tag{2.3}$$

Relationship of $GR_t(I)$ and $Pr(c_t|I)$

From the equation (2.1) and (2.2), the relationship of $GR_t(I)$ and $Pr(c_t|I)$ is represented by the equation (2.4). When enumerating emerging pattern by specifying the minimum posterior probability π according to the internal equation (2.4), π is converted to the minimum growth rate γ.

$$GR_t(I) = \frac{\Pr(c_o)}{\Pr(c_t)} \cdot \frac{\Pr(c_t|I)}{1 - \Pr(c_t|I)} \tag{2.4}$$

Appendix 2: Enumerate emerging patterns with LCM

The relationship of size of positive and negative growth examples of data items, D_t and D_o are expressed as $|D_t| = W|D_o|$. Now, for itemset I (referred as pattern I in the following), the growth rate $GR_t(I)$ in D_t and the occurrence gain $Gain_t(I)$ are defined in the respective formula (2.5) and (2.6).

$$GR_t(I) = \frac{sup(I, D_t)/|D_t|}{sup(I, D_o)/|D_o|} = W\frac{sup(I, D_t)}{sup(I, D_o)} \tag{2.5}$$

$$Gain_p(I) = \omega sup(I, D_t) - sup(I, D_o) \tag{2.6}$$

Here, $sup(I, D_t), sup(I, D_o)$ are the number of occurrence in D_t, D_o of pattern I. ω represents the weight assigned to the number of occurrence in an example with positive growth.

For pattern I, when $GR_t(I)$ is more than the minimum growth rate γ specified by user, the pattern is known as emerging pattern. When $Gain_p(I)$ is more than the minimum support σ specified by user, the pattern is known as contrast pattern. Equation (2.7) and (2.8) shows the emerging pattern and contrast pattern by the number of positive and negative occurrence.

$$sup(I, D_o) \leq \frac{W}{\gamma} sup(I, D_t) \tag{2.7}$$

$$sup(I, D_o) \leq \omega sup(I, D_t) - \sigma \tag{2.8}$$

When charting the number of occurrence $sup(I, D_o)$ from the negative example on the y-axis against $sup(I, D_t)$ from the positive example on the x-axis, emerging patterns are indicated by the shaded area in 2.1, and contrast patterns are indicated by the shaded area in 2.2.

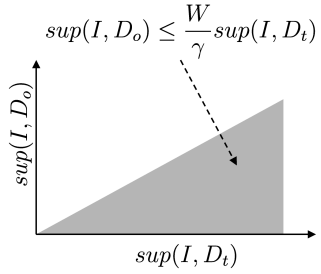

Figure 2.1: appearance pattern

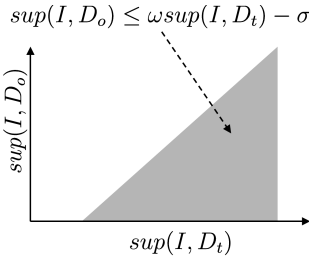

Figure 2.2: contrast pattern

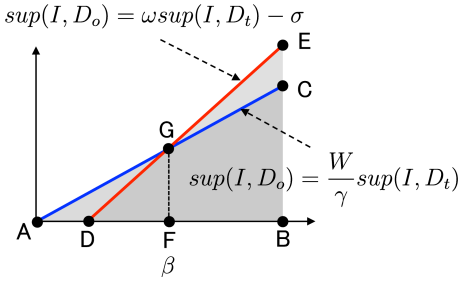

Figure 2.3: the relationship between the two patterns

LCM enumerates contrast patterns at high speed based on the parameters ω, σ specified by users. In enumerating contrast patterns, when $sup(I, D_t)$ becomes larger, sometimes it enumerates patterns with relatively small differences with $sup(I, D_o)$. This pattern is not representative as there is no distinguishable features to class c_t. In order to avoid this drawback, this command uses enumeration of emerging patterns.

The problem is how to enumerate contrast patterns by using LCM to enumerate emerging patterns. The following illustrates the method adopted for this command.

Diagram 2.3, 2.1 and 2.2 are artificially generated.

Instead of enumerating all emerging patterns from area ABC, consider enumerating emergence pattern from area GFCB which satisfy $sup(I, D_t) \geq \beta$ (β is specified at S= parameter in this command). Patterns enumerated by LCM belongs to $\triangle DEB$. The straight line DE is determined by defining σ and ω.

Set σ as the intersection point G of straight line AC and DE on x-coordinate, and β is set (The method of determining the equation 2.9):ω will be discussed below).

By removing $\triangle DFG$ and the pattern from $\triangle EGC$ from all patterns enumerated from LCM, emerging patterns with distinguish features can be enumerated.

$$\sigma = \beta(\omega - \frac{W}{\gamma}) \tag{2.9}$$

Next, find the slope of the line DE corresponding to ω. In general, the patterns of $\triangle DGF$ is far more than the patterns of $\triangle EGC$.Thus, it is far more efficient to increase ω.

However, ω is the weight of transaction, counting on the computer is limited to the maximum value of the variable type. If the maximum value is $maxInt$, it must satisfy the constraint of $\omega sup(I, D_t) \leq maxInt$.

Since $sup(I, D_t)$ will not exceed $|D_p|$, by defining ω according to the equation (2.10), the number of enumerations from $\triangle DFG$ can be minimised.

$$\omega = \frac{maxInt}{|D_t|} \tag{2.10}$$

2.2 msequence.rb - Enumerate Frequent Sequential Patterns

Enumerate frequent sequential pattern from time series data items. Sequential item data is a set of item arranged in sequential order. This command enumerates partial sequential patterns that appears frequently in sequential item data.

Sequential data emerging (or matching) in a data series refers to all items that make up the pattern are arranged in order according to series. LCMseq is used as the core algorithm for enumeration (LCM algorithm for enumerating all frequently emerging sequences). This command has the following characteristics.

- Define gap length and window width constraint (upper limit).
- Define gap length and window width constraint (upper limit) as computation time limit.
- Use item taxonomy for hierarchical classification.
- When classification class is specified, it is possible to enumerate patterns with specific characteristics (emerging sequential pattern) for a certain class. 3 or more classes can be defined.
- Cannot handle multiple sequential itemsets (different items with same time sequence).

Table 2.13 shows the examples of input data used in this command. Each unique sequence is identified by tid, data in time column and item column are displayed in order. The command do not support data with multiple items with the same time stamp (The operation result is undefined when there are multiple items with the same time stamp). However, time is basically used to determine the order of items. When the `-padding` option is specified, it is possible to set the gap length and the window width corresponding to the specified time value as integer value (refer to later section). Data shown in Table 2.13 is arranged by item sequence in 2.14 and by time sequence in 2.15.

Table 2.13: Data Series

tid	time	item
T1	0	C
T1	2	B
T1	3	A
T1	7	C
T2	2	D
	:	

Table 2.14: Display as vector

T1	C B A C
T2	D A B C
T3	C B D E
T4	A C B
T5	B A D D C C
T6	A B D B C

Table 2.15: Display based on time

	0	1	2	3	4	5	6	7	8	9
T1	C		B	A				C		
T2			D	A		B	C			
T3		C	B		D				E	
T4			A				C			B
T5	B	A	D	D				C		C
T6	A					B	D		B	C

In this data, sequential patterns that appear 2 or more times include (AC), (BC), further (DBC) surfaced for 20 or more times (refer to Example 1). The sequential pattern (BC) appear in records with tid T1, T2, T5, T6. Although T3, T4 contains item B and C, they are not included due to the difference in sequential order.

Frequent Sequential Pattern

Frequent sequential pattern refers to sequence patterns with a frequency of occurrence (support) greater than the minimum support defined by the user. Given a minimum supports of 3, sequential pattern (BD) is frequent since the sequential data T3, T5, T6 appeared 3 times.

On contrary, (BDC) is not frequent since it only appeared 2 times in T5, T6. Table 2.16 shows all frequent sequential patterns which meet the minimum support of 3 and its occurrence. When the sequential pattern is reversed (DB), sequential data T2, T6 only appeared 2 times and thus is not a frequent sequential pattern since it does not meet the minimum support.

Table 2.16 displays all frequent sequential patterns that meet the minimum support of 3.

Emerging Sequential Pattern

Use "class" to classify corresponding data, and enumerate sequential patterns with specific feature for each class. "Feature" is unique characteristics frequently found in one class but not in other classes. For instance, this technique can be used to identify the difference of the order of items purchased by men and women

Table 2.16: Shows all frequent sequential patterns and the respective transactions that meet the minimum support of 3.

Sequential Pattern	Occurrence	Transaction
C	6	T1,T2,T3,T4,T5,T6
B	6	T1,T2,T3,T4,T5,T6
A	5	T1,T2,T4,T5,T6
A C	5	T1,T2,T4,T5,T6
B C	4	T1,T2,T5,T6
D	4	T2,T3,T5,T6
A B	3	T2,T4,T6
B D	3	T3,T5,T6
C B	3	T1,T3,T4
D C	3	T2,T5,T6

in supermarket purchase data. The examples of enumerating emerging sequential patterns are illustrated in Example 5. In addition, growth rate and posterior probability is used as an index to evaluate emerging sequential patterns as discussed in Appendix 1 in mitemset.rb command.

Hierarchical Classification

An item can be expressed in terms of hierarchical classification. Please refer to mitemset.rb command for more details.

Upper Limit of Gap

The gap length between two items of any adjacent sequential pattern is defined as the distance between partial sequences matching the serial data (number of items between 2 items -1). For example, given the sequential pattern (ABC) and sequential data (ADDDBDC), the gap length between 2 adjacent items AB in the sequential pattern is 4, and the gap length between BC is 2. According to the definition of "emergence", specifying the maximum gap length constrains the gap length below the user specified value. When there are multiple matches, any matches that satisfy the constraint are considered. When the gap length limit is set as 1, adjacent frequent sequential patterns in the data will be enumerated.

Example of computation of gap length is shown in Table 2.17.

Upper Limit of Window

The window width is the length (number of items) of partial sequence on matched sequential data from the starting point to the ending point. For example, given the pattern (ABC) and sequential data (CADCBDC), the starting point for matching starts from the second item, and the ending point is at the seventh item, with a window width of 6. By specifying the upper limit of window width, the emerging pattern is constrained to any matches below the specified limit. When there are multiple matches, any match that satisfy the constraint is considered. Table 2.17 shows an example to calculate window width.

Time limit

In LCMseq, it is not possible to specify a gap length and a window width by directly defining the time of occurrence of the item. Therefore, the time limit is realized by the introducing a fictitious item ("!": exclamation mark) during pre-processing stage to represent the absence of time for the item.

[1] For example, the sequential data shown in Table 2.15 is converted to the data shown in Table 2.18. In addition, the sequential pattern is enumerated with gap length and window width constrains and the inclusion of fictitious items. Finally, the sequential pattern including fictitious items is suppressed in the output.

[1] Thus, "!" cannot be use as a character string of an item.

Table 2.17: The matching position of patterns (ABC), the gap length and window width. Below shows four matching records in the sequential data AAABCC, and the corresponding gap length and window width. Gap length and window with for all matches are shown. For example, when the upper limit of window is set to 3 results in emerging pattern ABC. When the upper limit of window is set to 2 results in no emerging pattern.

Sequential Data	Gap length bet. A-B	Gap length bet. B-C	Window
A D D D D **B** D **C** D	5	2	8
A **B** **C** D	1	1	3
C **A** A C **B** A **C** C	3	2	6
C **A** C **B** B A **C** B C	2	3	6
A A **B** **C** C	2	1	4
A **A** **B** **C** C	1	1	3
A A **B** C **C**	2	2	5
A **A** **B** C **C**	1	2	4

Table 2.18: To diagnose gap length and window constraints based on time considerations, replace empty item with the fictitious item "!".

	Sequence
T1	C ! B A ! ! ! C
T2	D A ! B C
T3	C B ! D ! ! ! E
T4	A ! ! ! C ! ! B
T5	B A D D ! ! ! C ! C
T6	A ! ! ! ! B D B ! C

Output

This command returns two sets of output data, the first set is the enumerated sequential pattern data (`patterns.csv`), the second set of data contains the corresponding transaction information of the pattern (`tid_pats.csv`). Note that the CSV fields in pattern data output will be returned for emerging patterns. The samples are shown in Table 2.19 to Table 2.21.

Table 2.19: Example of patterns.csv data. Column pid contains ID which identifies individual sequential pattern, size refers to the number of items that make up the item set pattern, count refers to the number of patterns in the sequential data, and total refers to the number of sequential data. Support is the probability of occurrence, calculated by count/total. Finally, pattern is the sequential pattern, with items delimited by single space character.

Pid	Pattern	Size	Count	Total	Support
1	C	1	6	6	1
4	B	1	6	6	1
11	A C	2	5	6	0.8333333333
10	A	1	5	6	0.8333333333
16	D	1	4	6	0.6666666667
7	B C	2	4	6	0.6666666667
12	A B	2	3	6	0.5
2	C B	2	3	6	0.5
19	D C	2	3	6	0.5
3	C C	2	2	6	0.3333333333
:	:	:	:	:	:

Table 2.20: Snapshot of data in tid-pats.csv. Tid is the ID of sequential data, it corresponds to the input field specified at the tid= parameter. The ID of sequential pattens in the sequential data is represented by Pid.

Tid	Pid
T1	0
T1	1
T1	10
T1	2
T2	0
T2	1
T2	10
T2	11
T3	0
T3	1
:	:

Table 2.21: Example of emerging patterns in patterns.csv. The class field indicates the target class based on the characteristics shown in emerging patterns. Attributes pid,pattern,size,total shown in Table 2.19 are defined in the previous table. Pos refers to the number of target class appeared in the transaction, neg is the number of other classes in the transaction. The total transaction numbers of target and and non target classes are indicated in posTotal and negTotal respectively. Support is the probability of occurrence, calculated by pos/posTotal. The change is represented by growthRate, calculated by support/(neg/negTotal). The result is shown as inf when the denominator is 0. As this value increases, the key feature for target class emerges. Posterior probability of the target class is represented by postProb, as with growthRate, as the value grow larger, it shows the key feature for the target class. Detailed definition is illustrated in Appendix 1 in mitemset.rb command manual.

class	pid	pattern	size	pos	neg	posTotal	negTotal	total	support	growthRate	postProb
cls1	1	B C	2	3	0	4	2	6	0.75	inf	1
cls2	9	B C D	3	2	0	2	4	6	1	inf	1
cls2	10	A D	2	2	0	2	4	6	1	inf	1
cls2	11	A C D	3	2	0	2	4	6	1	inf	1
cls2	8	B D	2	2	1	2	4	6	1	4	0.6666666667
cls2	12	C D	2	2	1	2	4	6	1	4	0.6666666667

2.2.1 Format

```
msequence.rb i= [x=] [O=] [tid=] [item=] [class=] [taxo=] [s=|S=] [sx=|SX=] [l=] [u=]
             [gap=] [win=] [p=] [g=] [top=] [-padding] [T=] [--help]

  i=       File name of key type transaction data [required parameter]
  x=       File name of hierarchical classification data [optional parameter]
  O=       Output path name  [optional: default=./take_#{DateTime}]
  tid=     Field name of transaction ID [required parameter]
  time=    Field name of transaction ID [required parameter]
  item=    Time based field name (field name in i=) [optional: default="time"]
  class=   Field name of class (field name in c=) [optional parameter]
           Emerging patterns is enumerated based on the class field defined.
  taxo=    Field name of taxonomy [required parameter with conditions: x=]
  s=       Minimum support (probability) [select either one parameter: s=, S=]
  S=       Minimum support (hits)  [select either one parameter: s=, S=]
  sx=      Maximum support (probability) [optional parameter]
  SX=      Maximum support (hits)  [optional parameter]
  l=       Minimum itemset size [optional parameter]
  u=       Maximum itemset size [optional parameter]
  gap=     Upper limit of pattern gap length (integer above 0) [optional: 0=without limit, default:0]
  win=     Upper limit of window size of pattern (integer above 0)
  [optional: 0=without limit, default:0]
  p=       Minimum posterior probability for emerging patterns.  [optional: default=0.5]
  g=       Minimum growth rate for emerging patterns  [optional parameter]
  top=     Upper limit of number of patterns to enumerate [optional: default: without limit]
  -padding Assume time is an integer, emulate special items which are not in time series.
           This will affect the definition of gap= and win= parameter.
  T=       Working directory [optional parameter]
  --help   Show help information
```

2.2.2 Example

Example 1: Basic Example

Display sequential patterns which occured more than 2 times. Note that the field names in the input data are the same as default parameters and thus the specification of field names is not required.

```
$ more dat1.csv
tid,time,item
T1,0,C
T1,2,B
T1,3,A
T1,7,C
T2,2,D
T2,3,A
T2,5,B
T2,6,C
T3,1,C
T3,2,B
T3,4,D
T3,8,E
T4,2,A
T4,5,C
T4,6,B
T5,0,B
T5,1,A
T5,2,D
T5,3,D
T5,7,C
T5,9,C
T6,0,A
T6,5,B
T6,6,D
T6,8,B
T6,9,C
$ msequence.rb  O=result1 i=dat1.csv S=2
#MSG# lcm_seq_20140215 CIf /tmp/__MTEMP_19670_70131942364220_0 2 /tmp/__MTEMP_19670_701319
42378600_0
trsact: /tmp/__MTEMP_19670_70131942364220_0 ,#transactions 6 ,#items 5 ,size 26 extracted
database: #transactions 6 ,#items 4 ,size 25
 output to: /tmp/__MTEMP_19670_70131942378600_0
iters=20
20
1
4
10
5
#MSG# output tid-patterns ...
#MSG# the number of contrast patterns enumerated is 19
#MSG# The final results are in the directory 'result1'
#END# /Users/stephane/.rvm/rubies/ruby-1.9.3-p448/bin/msequence.rb O=result1 i=dat1.csv S=
2
$ more result1/patterns.csv
pid,pattern,size,count,total,support
1,C,1,6,6,1
4,B,1,6,6,1
11,A C,2,5,6,0.8333333333
10,A,1,5,6,0.8333333333
16,D,1,4,6,0.6666666667
7,B C,2,4,6,0.6666666667
12,A B,2,3,6,0.5
8,B D,2,3,6,0.5
2,C B,2,3,6,0.5
19,D C,2,3,6,0.5
3,C C,2,2,6,0.3333333333
18,D B C,3,2,6,0.3333333333
13,A B C,3,2,6,0.3333333333
14,A D,2,2,6,0.3333333333
15,A D C,3,2,6,0.3333333333
9,B D C,3,2,6,0.3333333333
17,D B,2,2,6,0.3333333333
6,B A C,3,2,6,0.3333333333
5,B A,2,2,6,0.3333333333
```

Example 2: Limit of pattern length

Enumerate sequential patterns with length of 2 which occurred more than 3 or more times.

```
$ msequence.rb  O=result2 i=dat1.csv S=3 l=2 u=2
#MSG# lcm_seq_20140215 CIf -l 2 -u 2 /tmp/__MTEMP_19729_70097587586600_0 3 /tmp/__MTEMP_19
729_70097587584720_0
trsact: /tmp/__MTEMP_19729_70097587586600_0 ,#transactions 6 ,#items 5 ,size 26 extracted
database: #transactions 6 ,#items 4 ,size 25
```

```
 output to: /tmp/__MTEMP_19729_70097587584720_0
iters=11
6
0
0
6
#MSG# output tid-patterns ...
#MSG# the number of contrast patterns enumerated is 6
#MSG# The final results are in the directory 'result2'
#END# /Users/stephane/.rvm/rubies/ruby-1.9.3-p448/bin/msequence.rb O=result2 i=dat1.csv S=
3 l=2 u=2
$ more result2/patterns.csv
pid,pattern,size,count,total,support
3,A C,2,5,6,0.8333333333
1,B C,2,4,6,0.6666666667
0,C B,2,3,6,0.5
2,B D,2,3,6,0.5
4,A B,2,3,6,0.5
5,D C,2,3,6,0.5
$ more result2/tid_pats.csv
tid,pid
T1,0
T1,1
T1,3
T2,1
T2,3
T2,4
T2,5
T3,0
T3,2
T4,0
T4,3
T4,4
T5,1
T5,2
T5,3
T5,5
T6,1
T6,2
T6,3
T6,4
T6,5
```

Example 3: Specify gap length and window size

Enumerate sequential patterns with length above 2 which occurred more than 2 or more times, with gap length at 2 and window size at 4.

```
$ msequence.rb  O=result3 i=dat1.csv S=2 l=2 gap=2 win=4
#MSG# lcm_seq_20140215 CIf -l 2 -g 2 -G 4 /tmp/__MTEMP_19789_70106029991460_0 2 /tmp/__MTE
MP_19789_70106029989580_0
trsact: /tmp/__MTEMP_19789_70106029991460_0 ,#transactions 6 ,#items 5 ,size 26 extracted
database: #transactions 6 ,#items 5 ,size 26
 output to: /tmp/__MTEMP_19789_70106029989580_0
iters=15
10
0
0
9
1
#MSG# output tid-patterns ...
#MSG# the number of contrast patterns enumerated is 10
#MSG# The final results are in the directory 'result3'
#END# /Users/stephane/.rvm/rubies/ruby-1.9.3-p448/bin/msequence.rb O=result3 i=dat1.csv S=
2 l=2 gap=2 win=4
$ more result3/patterns.csv
pid,pattern,size,count,total,support
0,C B,2,3,6,0.5
2,B C,2,3,6,0.5
3,B D,2,3,6,0.5
4,A C,2,3,6,0.5
5,A B,2,3,6,0.5
1,B A,2,2,6,0.3333333333
6,A D,2,2,6,0.3333333333
7,D B,2,2,6,0.3333333333
8,D B C,3,2,6,0.3333333333
9,D C,2,2,6,0.3333333333
```

Example 4: Dealing with time with padding

Given the same criteria as example 3, when -padding is specified, enumerate sequential patterns in consideration of time based on gap length and window size constraints.

```
$ msequence.rb  O=result4 i=dat1.csv S=2 l=2 gap=2 win=4 -padding
#MSG# lcm_seq_zero_20140215 CIf -l 2 -g 2 -G 4 /tmp/__MTEMP_19848_70219658610980_0 2 /tmp/
__MTEMP_19848_70219658608980_0
trsact: /tmp/__MTEMP_19848_70219658610980_0 ,#transactions 6 ,#items 6 ,size 46 extracted
database: #transactions 6 ,#items 6 ,size 46
 output to: /tmp/__MTEMP_19848_70219658608980_0
iters=33
4
0
0
4
#MSG# output tid-patterns ...
#MSG# the number of contrast patterns enumerated is 4
#MSG# The final results are in the directory 'result4'
#END# /Users/stephane/.rvm/rubies/ruby-1.9.3-p448/bin/msequence.rb O=result4 i=dat1.csv S=
2 l=2 gap=2 win=4 -padding
$ more result4/patterns.csv
pid,pattern,size,count,total,support
0,C B,2,3,6,0.5
3,B D,2,3,6,0.5
1,B A,2,2,6,0.3333333333
2,B C,2,2,6,0.3333333333
```

Example 5: Enumerate emerging patterns

Given the same criteria as in example 1, enumerate emerging patterns by specifying class field.

```
$ more dat2.csv
tid,time,item,class
T1,0,C,cls1
T1,2,B,cls1
T1,3,A,cls1
T1,7,C,cls1
T2,2,D,cls1
T2,3,A,cls1
T2,5,B,cls1
T2,6,C,cls1
T3,1,C,cls1
T3,2,B,cls1
T3,4,D,cls1
T3,8,E,cls1
T4,2,A,cls1
T4,5,C,cls1
T4,6,B,cls1
T5,0,B,cls2
T5,1,A,cls2
T5,2,D,cls2
T5,3,D,cls2
T5,7,C,cls2
T5,9,C,cls2
T6,0,A,cls2
T6,5,B,cls2
T6,6,D,cls2
T6,8,B,cls2
T6,9,C,cls2
$ msequence.rb  O=result5 i=dat2.csv S=2 class=class -padding
#MSG# lcm_seq_zero_20140215 CIA -w /tmp/__MTEMP_19909_70295357403720_1 /tmp/__MTEMP_19909_
70295357403720_0 1073741815 /tmp/__MTEMP_19909_70295357416500_0
trsact: /tmp/__MTEMP_19909_70295357403720_0 ,#transactions 6 ,#items 6 ,size 46 extracted
database: #transactions 6 ,#items 5 ,size 45 ,weightfile /tmp/__MTEMP_19909_70295357403720
_1
 output to: /tmp/__MTEMP_19909_70295357416500_0
iters=33
9
1
4
4
#MSG# output patterns to CSV file ...
#MSG# the number of contrast patterns on class 'cls1' enumerated is 8
#MSG# output tid-patterns ...
#MSG# lcm_seq_zero_20140215 CIA -w /tmp/__MTEMP_19909_70295357403720_2 /tmp/__MTEMP_19909_
```

```
70295357403720_0 2147483645 /tmp/__MTEMP_19909_70295357416500_2
trsact: /tmp/__MTEMP_19909_70295357403720_0 ,#transactions 6 ,#items 6 ,size 46 extracted
database: #transactions 6 ,#items 5 ,size 45 ,weightfile /tmp/__MTEMP_19909_70295357403720
_2
 output to: /tmp/__MTEMP_19909_70295357416500_2
iters=36
5
0
0
3
2
#MSG# output patterns to CSV file ...
#MSG# the number of contrast patterns on class 'cls2' enumerated is 5
#MSG# output tid-patterns ...
#MSG# the number of emerging sequence patterns enumerated is 6
#MSG# The final results are in the directory 'result5'
#END# /Users/stephane/.rvm/rubies/ruby-1.9.3-p448/bin/msequence.rb O=result5 i=dat2.csv S=
2 class=class -padding
$ more result5/patterns.csv
class,pid,pattern,size,pos,neg,posTotal,negTotal,total,support,growthRate,postProb
cls1,1,B C,2,3,0,4,2,6,0.75,inf,1
cls2,9,B C D,3,2,0,2,4,6,1,inf,1
cls2,10,A D,2,2,0,2,4,6,1,inf,1
cls2,11,A C D,3,2,0,2,4,6,1,inf,1
cls2,8,B D,2,2,1,2,4,6,1,4,0.6666666667
cls2,12,C D,2,2,1,2,4,6,1,4,0.6666666667
```

2.3 mpolishing.rb - Polish general graph

Data polishing algorithm can be applied to general graph data to remove noise, and generate "polished graph" to increase distinction of graph. In order to illustrate the effect of polishing, Figure 2.4 and Figure 2.5 shows the graphs before and after polishing respectively. In the original graph, the dense structure is polished to enhance the density, while the sparse structure is thinned out. As a result, many medium-sized maximal cliques (other complete subgraph that does not exists exclusively in complete subgraph) are generated.

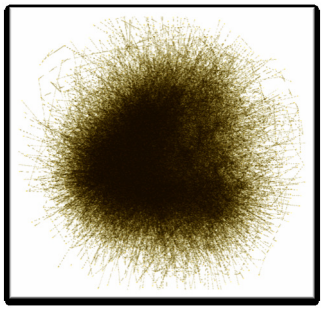

Figure 2.4: Graph before polishing

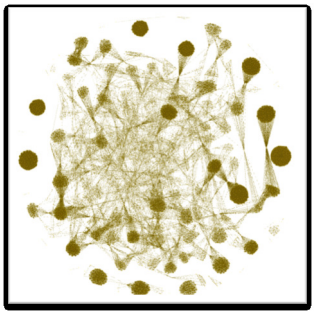

Figure 2.5: Graph after polishing

The polishing algorithm is shown in Algorithm 1. The algorithm shown below is not the most efficient but the mechanics shows how it works. Please refer to [2] for detailed reference on the algorithm which was implemented in the command. Polishing is based on a simple methodology by connecting all pairs of nodes (vertices) if the degree of similarity is more than or equal to threshold specified by user, otherwise, a new graph will be created from rules that were not connected.

Algorithm 1 Graph polishing algorithm

1: **function** POLISHING($G = (V, E), \sigma$)
2: $V : Vertexset, E : Edgeset, \sigma : Lowerlimitofsimilarity$
3: $E' = \phi$; $V' = \phi$ ▷ Initialization of edge set and vertex set after polishing
4: **for all** $u \in V$ **do**
5: **for all** $v \in V$ **do** ▷ # Search for all vertex pair u, v
6: **if** $sim(u, v) \geq \sigma$ **then** ▷ If the vertices pair u, v are similar, add a new edge, otherwise, do not add
7: $E' = E' \cup (u, v)$
8: $V' = V' \cup u$
9: $V' = V' \cup v$
10: **end if**
11: **end for**
12: **end for**
13: **return**(V', E')
14: **end function**

The same polishing procedure is repeatedly applied to newly constructed graph until there are no more changes to the graph structure, or when the number of iterations reaches the maximum value specified by user. The final graph obtained is the polished graph.

The definition of similarity between two nodes (Refer to line 6 of Algorithm 1:$sim(u, v)$) is shown. Basically, let's consider "Friends who have a lot of common friends". Figure 2.6 shows the structure where the nodes in the middle are connected to nodes u, v. Even though u and v are not directly connected, among the 5 connected nodes, nodes 3,4 have 2 common nodes connected. This information is used to define degree of similarity.

There are 6 types of similarity measures which can be applied in this command as shown in 2.22. The desired similarity measure can be defined at the `sim=` parameter, and the lower limit of degree of similarity can be specified at `th=` parameter.

2.3.1 Examples

Input data of general graph for this command is shown in Table 2.23, edge data and node pair is expressed in CSV format. The graph is processed as undirected graph, with multiple islands. Single node which does not

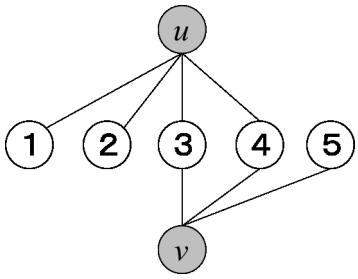

Figure 2.6: Connection relationship of vertex u, v

Table 2.22: Similarity definition of graph $G = (V, E)$ with nodes u, v

#	Degree of Similarity	Equation	sim= parameter	Range
1	resemblance	$\frac{\lvert N(u)\cap N(v)\rvert}{\lvert N(u)\cup N(v)\rvert}$	R	0.01.0
2	normalized PMI	$\log \frac{P(u,v)}{P(u)P(v)} / (-\log P(u,v))$ $= \frac{\lvert V\rvert \lvert N(u)\cap N(v)\rvert}{\lvert N(u)\rvert\lvert N(v)\rvert} / (-\log \frac{\lvert N(u)\cap N(v)\rvert}{\lvert V\rvert})$	P	-1.01.0
3	intersection	$\lvert N(u)\cap N(v)\rvert$	T	0
4	cosine	$\frac{\lvert N(u)\cap N(v)\rvert}{\sqrt{\lvert N(u)\rvert}\sqrt{\lvert N(v)\rvert}}$	C	0.01.0
5	max-confidence	$\frac{\lvert N(u)\cap N(v)\rvert}{\max(\lvert N(u)\rvert,\lvert N(v)\rvert)}$	S	0.01.0
6	min-confidence	$\frac{\lvert N(u)\cap N(v)\rvert}{\min(\lvert N(u)\rvert,\lvert N(v)\rvert)}$	s	0.01.0

$N(u)$ denotes the set of nodes adjacent to node u. $P(u)$ represents the probability that the edge will reach node u where $P(u) = N(u)/|V|$.

have an edge is not subjected to polishing (no friend), and the node data is not used as input. Data shown in Table 2.23 is visualised as graph format in Figure 2.7. Example of data polishing using the sample data is shown below.

Table 2.23: Input Data (Edge data)

node1	node2
a	b
a	c
a	e
b	c
b	e
b	g
c	d
c	g
d	e
e	f

For simplicity, similarity is defined as intersection with a lower limit of 2. In other words, when the adjacent nodes has 2 or more common nodes, extend the edge, and vice versa. For example, Figure 2.8, Figure 2.7 is a subgraph extracted of which are connected with nodes a, b. Having the 2 nodes e, c in common, the edge of node a, b is extended in the polished graph. While the common node for e, g(Figure 2.9) is only b, the edge is not extended in the polished graph.

Figure 2.10 shows a subgraph related to node a, b. The difference from the past 2 examples is that nodes a, b are directly connected. In this case, there are two consideration. First, in Figure 2.11, without considering direct relationship, only take into account common relationships with friends. The other method assumes nodes a, b have common friends, fictitious node a', b' is added, and it is assumed that both a, b are connected as shown in Figure 2.12. Therefore, if there is a direct connection, there will only be two people as common friends.

When updating new connection relationship for all nodes pairs as described above, the polished graph excluding direct relationship is shown in Figure 2.13, and the polished graph with direct relationship is shown in Figure

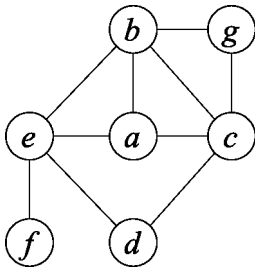

Figure 2.7: Visualization of graph data in Table 2.23

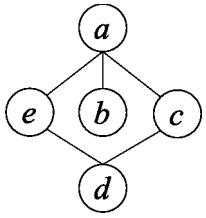

Figure 2.8: Relationship between node a and node d

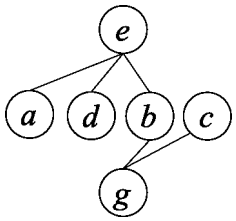

Figure 2.9: Relationship between node e and node$g$

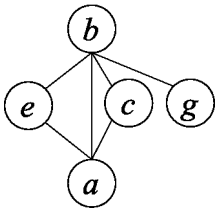

Figure 2.10: Relationship between node a and node b

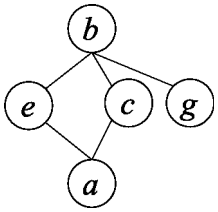

Figure 2.11: Example without direct connection

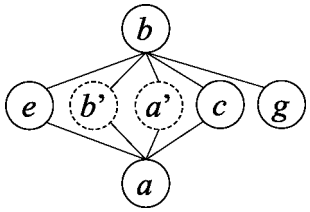

Figure 2.12: Example with direct connection

2.15. When the newly graphs are repeatedly polished and the graph structure did not change (converge) for 3 times, Figure 2.14 and Figure2.16 are obtained. When direct connection is not considered, all relationship connections are removed. On the other hand, if direct connect is considered, the six nodes a, b, c, d, e, g are connected to each other and maximal cliques are created. However, node f is still connected with e, which also forms maximal cliques e, f. Finally, when data polishing is repeated applied, the graph structure will gradually be stabilized. However, in some cases, the structure does not converge.

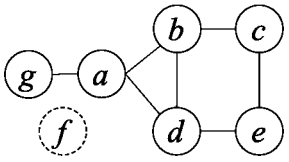

Figure 2.13: Graph polished once without direct connection

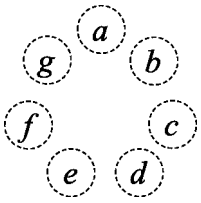

Figure 2.14: Graph polished 3 times without direct connection

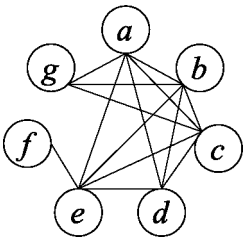

Figure 2.15: Graph polished once with direct connection

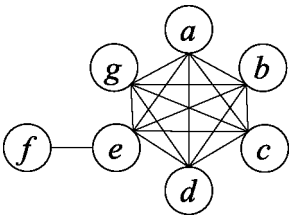

Figure 2.16: Graph polished 3 times with direct connection

Similarity in simple terms, can be defined as the number of edges in common for data polishing. However, as the size of the graph increases, there are more cases where the edges cannot be polished. As the number of adjacent nodes increases, the number of nodes become huge, even when edges are extended for nodes pairs sharing common nodes, the relationship is relatively weak.

Therefore, by applying resemblance, the degree of similarity can be used to evaluate relative common similarity measure. When resemblance is used as similarity measure, graphs that are polished until convergence at lower limit at 0.4 and 0.5 is shown in Figure 2.17 and 2.18.

Figure 2.19 is a result of applying normalized PMI as degree of similarity, with a lower limit of 0.2.

Other than the polished graph, various operation statistics can be returned in the output as shown in Table 2.24 (File specified at `log=`).

The characteristics of data polishing is summarized below.

- Extend edges according to information on common adjacent nodes.
- Select from 6 types of similarity measures, the results vary according to the type of similarity measure.
- The graph structure is stabilised by iterating data polishing cycles (converges at most times).

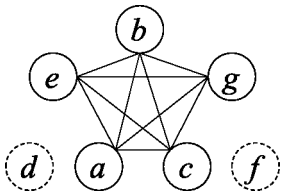

Figure 2.17: Polished graph with resemblance=0.4

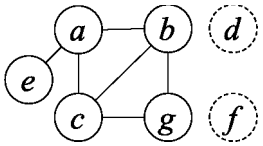

Figure 2.18: Polished graph with resemblance=0.5

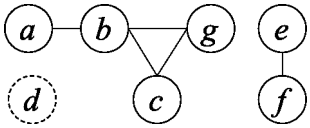

Figure 2.19: Polished graph with normalized PIM=0.2

Table 2.24: Statistics to be displayed in log file

Key	Description
iter	Iterations for data polishing
time	Execution time
nSize0	Number of graph nodes before polishing
eSize0	Number of graph edges before polishing
dens0	Density of edges before polishing
nSize#	Number of nodes after # polishing iterations
eSize#	Number of edges after # polishing iterations
dens#	Density of edges after # polishing iterations

- Number of maximal cliques is reduced drastically when compared to the original graph.
- Maximal cliques are formed when the lower limit of degree of similarity decreases.

2.3.2 Format

```
mpolishing.rb i= f= o= [sim=R|P|C|s|S|T] th= [sup=] [-int] [-indirect] [iter=] [log=] [T=] [--help]

  Specification of file
  i=     : Edge data file
  f=     : Field name of 2 nodes in edge data (cannot be used when -int is specified)
  o=     : Edge data file after data polishing
  sim=   : Definition of degree of similarity as benchmark for edge extension
               R: resemblance
              P: normalized PMI
              C: cosine
              S: max-confidence
              s: min-confidence
              T: intersection
  th=    : Lower limit of degree of similarity (specified by sim=) as benchmark for edge extension
  sup=   : Add the condition intersection>=sup for the calculation of similarity. Default is set as sup=0
  -int   : Treat item as integer for processing.
  -indirect: Exclude direct relationship  for adjacent nodes when calculating degree of similarity.
  iter=  : Maximum number of iterations for data polishing (default=30)
  log=   : Output file name of parameter settings and convergence information in CSV key-value format.

Others
  T= : Working directory (default:/tmp)
  O=     : Directory name to save the data from polishing process in debug mode
  --help : Help information
```

2.3.3 Examples

Example 1: Basic Example

Use resemblance(sim=R) as similarity measure, the polished graph is obtained by extending the branch at th=0.4. The number of polishing iterations are converged 3 times as shown in `log1.csv` (iter,4) The output result is shown in Figure 2.17.

```
$ more dat1.csv
node1,node2
a,b
a,c
a,e
b,c
b,e
b,g
c,d
c,g
d,e
e,f
$ mpolishing.rb i=dat1.csv f=node1,node2 sim=R th=0.4 o=result1.csv log=log1.csv
#MSG# converting graph files into a pair of numbered nodes ...
input-file /tmp/__MTEMP_20042_70166767738540_2, output-file /tmp/__MTEMP_20042_70166767738
540_3
degree threshold:
first & second scan end: /tmp/__MTEMP_20042_70166767738540_2
file read end: /tmp/__MTEMP_20042_70166767738540_2
iterative scan: #nodes=7, #edges = 20
forwardstar graph: /tmp/__MTEMP_20042_70166767738540_2 ,#nodes 7(7,7) ,#edges 20
#MSG# polishing iteration #0 (tra size=61
sspc_20140215 R -l 0 /tmp/__MTEMP_20042_70166767738540_3 0.4 /tmp/__MTEMP_20042_7016676773
8540_2
trsact: /tmp/__MTEMP_20042_70166767738540_3 ,#transactions 7 ,#items 7 ,size 27 extracted
database: #transactions 7 ,#items 7 ,size 27
 output to: /tmp/__MTEMP_20042_70166767738540_2
separated at 0
11 pairs are found
0,1,2,4, :1.000000  (0)
0,1,2,4,6, :1.000000  (0)
0,1,2,3,6, :1.000000  (0)
2,3,4, :1.000000  (0)
0,1,3,4,5, :1.000000  (0)
4,5, :1.000000  (0)
1,2,6, :1.000000  (0)
come
input-file /tmp/__MTEMP_20042_70166767738540_2, output-file /tmp/__MTEMP_20042_70166767738
540_3
degree threshold:
first & second scan end: /tmp/__MTEMP_20042_70166767738540_2
file read end: /tmp/__MTEMP_20042_70166767738540_2
iterative scan: #nodes=7, #edges = 22
forwardstar graph: /tmp/__MTEMP_20042_70166767738540_2 ,#nodes 7(7,7) ,#edges 22
#MSG# polishing iteration #1 (tra size=65
sspc_20140215 R -l 0 /tmp/__MTEMP_20042_70166767738540_3 0.4 /tmp/__MTEMP_20042_7016676773
8540_2
trsact: /tmp/__MTEMP_20042_70166767738540_3 ,#transactions 7 ,#items 7 ,size 29 extracted
database: #transactions 7 ,#items 7 ,size 29
 output to: /tmp/__MTEMP_20042_70166767738540_2
separated at 0
11 pairs are found
0,1,2,3,4,6, :1.000000  (0)
0,1,2,4,6, :1.000000  (0)
0,1,2,4,6, :1.000000  (0)
0,3, :1.000000  (0)
0,1,2,4,5, :1.000000  (0)
4,5, :1.000000  (0)
0,1,2,6, :1.000000  (0)
come
input-file /tmp/__MTEMP_20042_70166767738540_2, output-file /tmp/__MTEMP_20042_70166767738
540_3
degree threshold:
first & second scan end: /tmp/__MTEMP_20042_70166767738540_2
file read end: /tmp/__MTEMP_20042_70166767738540_2
iterative scan: #nodes=6, #edges = 22
forwardstar graph: /tmp/__MTEMP_20042_70166767738540_2 ,#nodes 7(7,7) ,#edges 22
#MSG# polishing iteration #2 (tra size=63
```

```
sspc_20140215 R -l 0 /tmp/__MTEMP_20042_70166767738540_3 0.4 /tmp/__MTEMP_20042_7016676773
8540_2
trsact: /tmp/__MTEMP_20042_70166767738540_3 ,#transactions 7 ,#items 7 ,size 28 extracted
database: #transactions 7 ,#items 7 ,size 28
 output to: /tmp/__MTEMP_20042_70166767738540_2
separated at 0
10 pairs are found
0,1,2,4,6, :1.000000  (0)
0,1,2,4,6, :1.000000  (0)
0,1,2,4,6, :1.000000  (0)
 :1.000000  (0)
0,1,2,4,5,6, :1.000000  (0)
4,5, :1.000000  (0)
0,1,2,4,6, :1.000000  (0)
come
input-file /tmp/__MTEMP_20042_70166767738540_2, output-file /tmp/__MTEMP_20042_70166767738
540_3
degree threshold:
first & second scan end: /tmp/__MTEMP_20042_70166767738540_2
file read end: /tmp/__MTEMP_20042_70166767738540_2
iterative scan: #nodes=5, #edges = 20
forwardstar graph: /tmp/__MTEMP_20042_70166767738540_2 ,#nodes 7(7,7) ,#edges 20
#MSG# polishing iteration #3 (tra size=57
sspc_20140215 R -l 0 /tmp/__MTEMP_20042_70166767738540_3 0.4 /tmp/__MTEMP_20042_7016676773
8540_2
trsact: /tmp/__MTEMP_20042_70166767738540_3 ,#transactions 7 ,#items 7 ,size 25 extracted
database: #transactions 7 ,#items 7 ,size 25
 output to: /tmp/__MTEMP_20042_70166767738540_2
separated at 0
10 pairs are found
0,1,2,4,6, :1.000000  (0)
0,1,2,4,6, :1.000000  (0)
0,1,2,4,6, :1.000000  (0)
 :1.000000  (0)
0,1,2,4,6, :1.000000  (0)
 :1.000000  (0)
0,1,2,4,6, :1.000000  (0)
come
input-file /tmp/__MTEMP_20042_70166767738540_2, output-file /tmp/__MTEMP_20042_70166767738
540_3
degree threshold:
first & second scan end: /tmp/__MTEMP_20042_70166767738540_2
file read end: /tmp/__MTEMP_20042_70166767738540_2
iterative scan: #nodes=5, #edges = 20
forwardstar graph: /tmp/__MTEMP_20042_70166767738540_2 ,#nodes 7(7,7) ,#edges 20
#MSG# converting the numbered nodes into original name ...
#END# /Users/stephane/.rvm/rubies/ruby-1.9.3-p448/bin/mpolishing.rb i=dat1.csv f=node1,nod
e2 sim=R th=0.4 o=result1.csv log=log1.csv
$ more result1.csv
node1,node2
a,b
a,c
a,e
a,g
b,c
b,e
b,g
c,e
c,g
e,g
$ more log1.csv
key,value
i=,dat1.csv
f=,"node1,node2"
sim=,R
th=,0.4
o=,result1.csv
log=,log1.csv
-int,false
-indirect,false
iter,4
time,0.113653
nSize0,7
eSize0,10
dens0,0.4761904762
nSize1,7
eSize1,11
dens1,0.5238095238
nSize2,6
```

```
eSize2,11
dens2,0.7333333333
nSize3,5
eSize3,10
dens3,1
nSize4,5
eSize4,10
dens4,1
```

Example 2: Polishing by PMI

Use normalized PMI(sim=P) as similarity measure, the polished graph is obtained by extending the branch at th=0.2. The output result is shown in Figure 2.19.

```
$ mpolishing.rb i=dat1.csv f=node1,node2 sim=P th=0.2 o=result2.csv
#MSG# converting graph files into a pair of numbered nodes ...
input-file /tmp/__MTEMP_20101_70350905868560_2, output-file /tmp/__MTEMP_20101_70350905868
560_3
degree threshold:
first & second scan end: /tmp/__MTEMP_20101_70350905868560_2
file read end: /tmp/__MTEMP_20101_70350905868560_2
iterative scan: #nodes=7, #edges = 20
forwardstar graph: /tmp/__MTEMP_20101_70350905868560_2 ,#nodes 7(7,7) ,#edges 20
#MSG# polishing iteration #0 (tra size=61
sspc_20140215 P -l 0 /tmp/__MTEMP_20101_70350905868560_3 0.2 /tmp/__MTEMP_20101_7035090586
8560_2
trsact: /tmp/__MTEMP_20101_70350905868560_3 ,#transactions 7 ,#items 7 ,size 27 extracted
database: #transactions 7 ,#items 7 ,size 27
 output to: /tmp/__MTEMP_20101_70350905868560_2
separated at 0
5 pairs are found
0,1,2,4, :1.000000  (0)
0,1,2,4,6, :1.000000  (0)
0,1,2,3,6, :1.000000  (0)
2,3,4, :1.000000  (0)
0,1,3,4,5, :1.000000  (0)
4,5, :1.000000  (0)
1,2,6, :1.000000  (0)
come
input-file /tmp/__MTEMP_20101_70350905868560_2, output-file /tmp/__MTEMP_20101_70350905868
560_3
degree threshold:
first & second scan end: /tmp/__MTEMP_20101_70350905868560_2
file read end: /tmp/__MTEMP_20101_70350905868560_2
iterative scan: #nodes=6, #edges = 10
forwardstar graph: /tmp/__MTEMP_20101_70350905868560_2 ,#nodes 7(7,7) ,#edges 10
#MSG# polishing iteration #1 (tra size=39
sspc_20140215 P -l 0 /tmp/__MTEMP_20101_70350905868560_3 0.2 /tmp/__MTEMP_20101_7035090586
8560_2
trsact: /tmp/__MTEMP_20101_70350905868560_3 ,#transactions 7 ,#items 7 ,size 16 extracted
database: #transactions 7 ,#items 7 ,size 16
 output to: /tmp/__MTEMP_20101_70350905868560_2
separated at 0
5 pairs are found
0,1, :1.000000  (0)
0,1,2,6, :1.000000  (0)
1,2,6, :1.000000  (0)
 :1.000000  (0)
4,5, :1.000000  (0)
4,5, :1.000000  (0)
1,2,6, :1.000000  (0)
come
input-file /tmp/__MTEMP_20101_70350905868560_2, output-file /tmp/__MTEMP_20101_70350905868
560_3
degree threshold:
first & second scan end: /tmp/__MTEMP_20101_70350905868560_2
file read end: /tmp/__MTEMP_20101_70350905868560_2
iterative scan: #nodes=6, #edges = 10
forwardstar graph: /tmp/__MTEMP_20101_70350905868560_2 ,#nodes 7(7,7) ,#edges 10
#MSG# converting the numbered nodes into original name ...
#END# /Users/stephane/.rvm/rubies/ruby-1.9.3-p448/bin/mpolishing.rb i=dat1.csv f=node1,nod
e2 sim=P th=0.2 o=result2.csv
$ more result2.csv
node1,node2
```

```
a,b
b,c
b,g
c,g
e,f
```

Example 3: Polish once at intersection

Based on the explanation in the previous example. Use intersection(sim=T) as similarity measure, consider cases with 2 or more (th=2) branch extension with direct connection. Polish graph with 1 iteration (iter=1). The output result is shown in Figure 2.15.

```
$ mpolishing.rb i=dat1.csv f=node1,node2 sim=T th=2 iter=1 o=result3.csv
#MSG# converting graph files into a pair of numbered nodes ...
input-file /tmp/__MTEMP_20151_70168512314200_2, output-file /tmp/__MTEMP_20151_70168512314
200_3
degree threshold:
first & second scan end: /tmp/__MTEMP_20151_70168512314200_2
file read end: /tmp/__MTEMP_20151_70168512314200_2
iterative scan: #nodes=7, #edges = 20
forwardstar graph: /tmp/__MTEMP_20151_70168512314200_2 ,#nodes 7(7,7) ,#edges 20
#MSG# polishing iteration #0 (tra size=61
sspc_20140215 T -l 0 /tmp/__MTEMP_20151_70168512314200_3 2.0 /tmp/__MTEMP_20151_7016851231
4200_2
trsact: /tmp/__MTEMP_20151_70168512314200_3 ,#transactions 7 ,#items 7 ,size 27 extracted
database: #transactions 7 ,#items 7 ,size 27
 output to: /tmp/__MTEMP_20151_70168512314200_2
separated at 0
14 pairs are found
0,1,2,4, :1.000000  (0)
0,1,2,4,6, :1.000000  (0)
0,1,2,3,6, :1.000000  (0)
2,3,4, :1.000000  (0)
0,1,3,4,5, :1.000000  (0)
4,5, :1.000000  (0)
1,2,6, :1.000000  (0)
come
input-file /tmp/__MTEMP_20151_70168512314200_2, output-file /tmp/__MTEMP_20151_70168512314
200_3
degree threshold:
first & second scan end: /tmp/__MTEMP_20151_70168512314200_2
file read end: /tmp/__MTEMP_20151_70168512314200_2
iterative scan: #nodes=7, #edges = 28
forwardstar graph: /tmp/__MTEMP_20151_70168512314200_2 ,#nodes 7(7,7) ,#edges 28
#MSG# converting the numbered nodes into original name ...
#END# /Users/stephane/.rvm/rubies/ruby-1.9.3-p448/bin/mpolishing.rb i=dat1.csv f=node1,nod
e2 sim=T th=2 iter=1 o=result3.csv
$ more result3.csv
node1,node2
a,b
a,c
a,d
a,e
a,g
b,c
b,d
b,e
b,g
c,d
c,e
c,g
d,e
e,f
```

Example 4: Exclude direct connection

When `-indirect` is specified, direct connection is ignored when calculating degree of similarity. The output graph is shown in Figure 2.14, to remove all branches, branch data is not returned after polishing.

```
$ mpolishing.rb i=dat1.csv f=node1,node2 sim=T th=2 o=result4.csv -indirect
#MSG# converting graph files into a pair of numbered nodes ...
input-file /tmp/__MTEMP_20194_70296082596580_2, output-file /tmp/__MTEMP_20194_70296082596
```

```
580_3
degree threshold:
first & second scan end: /tmp/__MTEMP_20194_70296082596580_2
file read end: /tmp/__MTEMP_20194_70296082596580_2
iterative scan: #nodes=7, #edges = 20
forwardstar graph: /tmp/__MTEMP_20194_70296082596580_2 ,#nodes 7(7,7) ,#edges 20
#MSG# polishing iteration #0 (tra size=47
sspc_20140215 T -l 0 /tmp/__MTEMP_20194_70296082596580_3 2.0 /tmp/__MTEMP_20194_7029608259
6580_2
trsact: /tmp/__MTEMP_20194_70296082596580_3 ,#transactions 7 ,#items 7 ,size 20 extracted
database: #transactions 7 ,#items 7 ,size 20
 output to: /tmp/__MTEMP_20194_70296082596580_2
separated at 0
6 pairs are found
1,2,4, :1.000000  (0)
0,2,4,6, :1.000000  (0)
0,1,3,6, :1.000000  (0)
2,4, :1.000000  (0)
0,1,3,5, :1.000000  (0)
4, :1.000000  (0)
1,2, :1.000000  (0)
come
input-file /tmp/__MTEMP_20194_70296082596580_2, output-file /tmp/__MTEMP_20194_70296082596
580_3
degree threshold:
first & second scan end: /tmp/__MTEMP_20194_70296082596580_2
file read end: /tmp/__MTEMP_20194_70296082596580_2
iterative scan: #nodes=6, #edges = 12
forwardstar graph: /tmp/__MTEMP_20194_70296082596580_2 ,#nodes 7(7,7) ,#edges 12
#MSG# polishing iteration #1 (tra size=31
sspc_20140215 T -l 0 /tmp/__MTEMP_20194_70296082596580_3 2.0 /tmp/__MTEMP_20194_7029608259
6580_2
trsact: /tmp/__MTEMP_20194_70296082596580_3 ,#transactions 7 ,#items 7 ,size 12 extracted
database: #transactions 7 ,#items 7 ,size 12
 output to: /tmp/__MTEMP_20194_70296082596580_2
separated at 0
0 pairs are found
1,3,6, :1.000000  (0)
0,2,3, :1.000000  (0)
1,4, :1.000000  (0)
0,1, :1.000000  (0)
2, :1.000000  (0)
 :1.000000  (0)
0, :1.000000  (0)
come
input-file /tmp/__MTEMP_20194_70296082596580_2, output-file /tmp/__MTEMP_20194_70296082596
580_3
degree threshold:
first & second scan end: /tmp/__MTEMP_20194_70296082596580_2
file read end: /tmp/__MTEMP_20194_70296082596580_2
iterative scan: #nodes=0, #edges = 0
forwardstar graph: /tmp/__MTEMP_20194_70296082596580_2 ,#nodes 0(0,0) ,#edges 0
#MSG# polishing iteration #2 (tra size=6
sspc_20140215 T -l 0 /tmp/__MTEMP_20194_70296082596580_3 2.0 /tmp/__MTEMP_20194_7029608259
6580_2
trsact: /tmp/__MTEMP_20194_70296082596580_3 ,#transactions 2 ,#items 4 ,size 1 extracted d
atabase: #transactions 2 ,#items 4 ,size 1
 output to: /tmp/__MTEMP_20194_70296082596580_2
separated at 0
0 pairs are found
 :1.000000  (0)
3, :1.000000  (0)
come
input-file /tmp/__MTEMP_20194_70296082596580_2, output-file /tmp/__MTEMP_20194_70296082596
580_3
degree threshold:
first & second scan end: /tmp/__MTEMP_20194_70296082596580_2
file read end: /tmp/__MTEMP_20194_70296082596580_2
iterative scan: #nodes=0, #edges = 0
forwardstar graph: /tmp/__MTEMP_20194_70296082596580_2 ,#nodes 0(0,0) ,#edges 0
#MSG# converting the numbered nodes into original name ...
#END# /Users/stephane/.rvm/rubies/ruby-1.9.3-p448/bin/mpolishing.rb i=dat1.csv f=node1,nod
e2 sim=T th=2 o=result4.csv -indirect
$ more result4.csv
node1,node2
```

2.4 mtra2g.rb - Construct item similarity graph

This command constructs a general graph (referred to as "similarity graph") to reflect the structural similarity between items within the transaction data . The `lcm` [3] command is used in mtra2g.rb. Degree of similarity is defined by the co-occurrence information of two items, by extending an edge between items a similarity measure higher than the lower limit specified by the user. Probability of item occurrence (support) and number of occurrence can be specified to derive degree of similarity. In addition, resemblance and the normalized PMI can be added as additional criteria. The definition of each is shown in Table **??**.

Table 2.25: Definition of degree of similarity between items a and b

Degree of similarity	Formula	Parameter	Range
support	$\frac{\lvert Occ(a,b)\rvert}{n}$	s=	0.01.0
occurence	$\lvert Occ(a,b)\rvert$	S=	1
recemblance	$\frac{\lvert Occ(a)\cap Occ(b)\rvert}{\lvert Occ(a)\cup Occ(b)\rvert}$	sim=R th=	0.01.0
normalized PMI	$\log \frac{P(a,b)}{P(a)P(b)}/(-\log P(a,b))$ $= \frac{n\lvert Occ(a)\cap Occ(b)\rvert}{\lvert Occ(a)\rvert\lvert Occ(b)\rvert}/(-\log \frac{\lvert Occ(a)\cap Occ(b)\rvert}{n})$	sim=P th=	-1.01.0

n represents the total number of transactions. $Occ(a)$ represents the transaction set which item a appeared. $P(a)$ represents the probability of occurrence for item a denoted as $P(a) = Occ(a)/n$.

The same key-based transaction data is used as the input data used in mitemset.rb command is used as input data in this section as shown in (Table 2.26). The data is converted from key based to transaction based formatted data by `mtra` command in MCMD package as shown in Table 2.27.

Given the input data, Table 2.28 shows the result of similarity graph with 2 or more occurrence, the graph structure is shown in Figure 2.20 .

Table 2.26: Key based data

key	item
T1	C
T1	E
T2	D
T2	E
T2	F
:	:

Table 2.27: Tra based data

id	item
T1	C E
T2	D E F
T3	A B D F
T4	B D F
T5	A B D E
T6	A B D E F

Table 2.28: Item similarity graph with 2 or more occurrence. The probability of occurrence indicates the number of occurrence. When sim= parameter is specified, the value will be shown in the final column (void).

node1	node2	support	void
a	b	0.6	
a	d	0.4	
a	f	0.4	
d	b	0.6	
e	d	0.4	
f	b	0.6	
f	d	0.6	

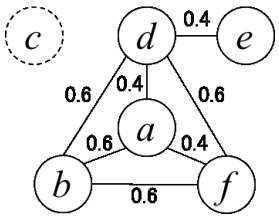

Figure 2.20: The corresponding similarity graph is shown in Figure 2.28. Each edge shows the co-occurrence probability of the two items.

2.4.1 Format

```
Format) mtra2g.rb i= tid= item= [on=] oe= s= [sim=] [th=] [log=] [T=] [--help]
```

```
  Specification of file name
  i=     : Transaction data file
  tid=   : Field name of Transaction ID
  item=  : Field name of item
  on=    : Output file (node)
  oe=    : Output file (side:node pair)
  s=     : Minimum support [(specified by percentage of total number of transactions): real number between
  S=     : Minimum support [(specify the number of transactions): integer above 1]
         : When both s=,S= is not specified, default setting becomes S=1.
  sim=   : Degree of similarity
           R: resemblance
           P: normalized PMI
           When not specified, the edge is extended based on the criteria specified at s= and S=.
  th=    :  Extend an edge between items with the degree of similarity above  the threshold value specifie
 log=   : Specify file name to save the parameter settings in key-based CSV format.

  Others
  T= : Working directory (default:/tmp)
  --help : Help information
```

2.4.2 Examples

Example 1: Basic Example

Display similarity graph with 2 or more occurrence. The example is shown above.

```
$ more dat1.csv
tid,item
T1,C
T1,E
T2,D
T2,E
T2,F
T3,A
T3,B
T3,D
T3,F
T4,B
T4,D
T4,F
T5,A
T5,B
T5,D
T5,E
T6,A
T6,B
T6,D
T6,E
T6,F
$ mtra2g.rb  S=2 tid=tid item=item i=dat1.csv oe=edge1.csv
#MSG# converting a named item into a numbered item ...
#MSG# run lcm enumerating 2 itemset ...
#MSG# creating the edge file ...
#END# /Users/stephane/.rvm/rubies/ruby-1.9.3-p448/bin/mtra2g.rb S=2 tid=tid item=item i=da
t1.csv oe=edge1.csv
$ more edge1.csv
node1,node2,support,void
A,B,0.5,
A,D,0.5,
A,E,0.3333333333,
A,F,0.3333333333,
B,D,0.6666666667,
E,B,0.3333333333,
E,D,0.5,
F,B,0.5,
F,D,0.6666666667,
F,E,0.3333333333,
```

Example 2: Add resemblance

Based on example 1, add the degree of similarity criteria where resemblance is above 0.4. By specifying `on=`, only the frequency of appearance of item is returned as node information.

```
$ mtra2g.rb  S=2 sim=R th=0.4 tid=tid item=item i=dat1.csv oe=edge2.csv on=node2.csv
#MSG# converting a named item into a numbered item ...
#MSG# run lcm enumerating 2 itemset ...
#MSG# creating the edge file ...
#MSG# creating the node file ...
#END# /Users/stephane/.rvm/rubies/ruby-1.9.3-p448/bin/mtra2g.rb S=2 sim=R th=0.4 tid=tid i
tem=item i=dat1.csv oe=edge2.csv on=node2.csv
$ more node2.csv
node,support
A,0.5
B,0.6666666667
C,0.1666666667
D,0.8333333333
E,0.6666666667
F,0.6666666667
$ more edge2.csv
node1,node2,support,resemblance
A,B,0.5,0.75
A,D,0.5,0.6
A,E,0.3333333333,0.4
A,F,0.3333333333,0.4
B,D,0.6666666667,0.8
E,D,0.5,0.5
F,B,0.5,0.6
F,D,0.6666666667,0.8
```

2.5 mclique.rb - Enumerate Maximal Cliques

This command enumerates the maximal cliques from general graph. The `mace` command [3] is used in mclique.rb. Given $G = (V, E)$ is an undirected graph with node (referred as vertex) set V and edge set E, a clique is a subset of its nodes (or vertices) V such that every two nodes in the subset are connected by an edge in the subgraph of G . A maximal clique is a clique that is not an exclusive subset of a larger clique (Figure 2.21).

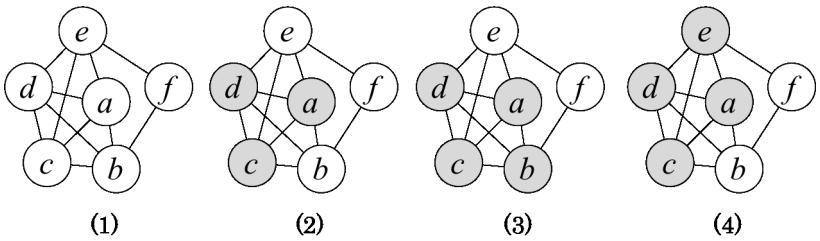

Figure 2.21: General graph (1) consists of a clique in the subgraph indicated by the shaded nodes in (2), but the subset is not maximal clique as shown in (3) and (4). On the other hand, (3) and (4) are maximal cliques since they are not a subset of other cliques. (3) and (4) is constructed by 4 nodes, with 3 common nodes.

In general cases, the data points grow to a huge number when enumerating maximal cliques. This is due to the fact that the nodes of maximal cliques overlap each other as shown in Figure 2.21 (3) and (4). To date, there are several proposals to overcome this problem. One proposal is to treat as a complete graph if the density reaches a particular level, this is known as pseudo clique. However, sometimes more pseudo cliques are enumerated then maximal cliques, and the fundamental problem is not resolved. Another approach is to merge similar cliques after enumerating maximal cliques, which make use of different available clustering algorithms. Although this approach is rather promising, the run time required could be a problem depending on the number of maximal cliques enumerated. The third approach is to polish the general graph before enumeration (this cleaning technique is referred to as "data polishing") to reduce the number of maximal cliques enumerated. Data polishing is recently proposed by Professor Uno [2], if the statistical proof of data polishing becomes more apparent, clique enumeration will become an effective methodology. In section 2.3, this technique is implemented in `mpolishing.rb` command. When mpolishing is used in conjunction with this command, the number of maximal cliques enumerated can be reduced drastically without altering the nature of the graph.

Table 2.29 shows the input data for this command, edge data is represented in node pair in CSV format (corresponds to Figure 2.21 (1)). The graph is treated as undirected graph and may have more than one island.

Given the graph data, maximal cliques are enumerated as shown in Table 2.30.

The field `id` denotes clique ID, this field identifies records in the same cliques. Clique where `id`=2 corresponds to Figure 2.21(4), and clique where `id`=3 corresponds to Figure 2.21(3). In addition, maximal cliques comprised of the nodes $\{e, f\}$ and $\{b, f\}$ are enumerated. The last column `size` contains the number of nodes that made up the maximal cliques.

2.5.1 Format

```
Format) mclique.rb i= f= [o=] [l=] [u=] [-all] [-node] [-all] [-int] [log=] [T=] [--help]

  Specify File Name
  i=      : Edge data file
  f=      : Column name of 2 nodes in edge data (invalid parameter when -int is specified)
  o=      : Output file (Clique ID-Edge: cliqueID-node can be modified when -node is specified)
  l=      :  Minimal number of nodes constructing the clique (clique smaller than the value
  specified will not be enumerated)
 u=      : Maximal number of nodes constructing the clique (clique larger than the value
 specified will not be enumerated)
  -node   : Output in clique ID-node name pair format
```

Table 2.29: Input Data (Edge data)

node1	node2
a	b
a	c
a	d
a	e
b	c
b	d
b	f
c	d
c	e
d	e
e	f

Table 2.30: Output Results

id	node1	node2	size
0	e	f	2
1	b	f	2
2	a	c	4
2	a	d	4
2	a	e	4
2	c	d	4
2	c	e	4
2	d	e	4
3	a	b	4
3	a	c	4
3	a	d	4
3	b	c	4
3	b	d	4
3	c	d	4

```
  -all   : Enumerate all cliques instead of only maximal cliques
  -int   : Process items as integers
  log=   : File name of the parameter settings saved in key based CSV format

 Others
  T= : Working directory (default:/tmp)
  --help : Help information
```

2.5.2 Examples

Example 1: Basic Example

Example illustrated from the above section.

```
$ more dat1.csv
node1,node2
a,b
a,c
a,d
a,e
b,c
b,d
b,f
c,d
c,e
d,e
e,f
$ mclique.rb i=dat1.csv f=node1,node2 o=result1.csv log=log1.csv
/Users/stephane/.rvm/gems/ruby-1.9.3-p448/gems/nysol-1.5-x86_64-darwin/lib/nysol/margs.rb:
154:in `block in initialize': I don't know such a argument: `i=' (RuntimeError)
        from /Users/stephane/.rvm/gems/ruby-1.9.3-p448/gems/nysol-1.5-x86_64-darwin/lib/nysol/mar
gs.rb:152:in `each'
        from /Users/stephane/.rvm/gems/ruby-1.9.3-p448/gems/nysol-1.5-x86_64-darwin/lib/nysol/mar
gs.rb:152:in `initialize'
        from /Users/stephane/.rvm/gems/ruby-1.9.3-p448/gems/nysol-1.5-x86_64-darwin/bin/mclique.r
b:124:in `new'
        from /Users/stephane/.rvm/gems/ruby-1.9.3-p448/gems/nysol-1.5-x86_64-darwin/bin/mclique.r
b:124:in `<top (required)>'
        from /Users/stephane/.rvm/rubies/ruby-1.9.3-p448/bin/mclique.rb:23:in `load'
        from /Users/stephane/.rvm/rubies/ruby-1.9.3-p448/bin/mclique.rb:23:in `<main>'
$ more result1.csv
result1.csv: No such file or directory
$ more cn1.csv
cn1.csv: No such file or directory
$ more ce1.csv
ce1.csv: No such file or directory
$ more log1.csv
log1.csv: No such file or directory
```

Example 2: Enumerate maximal cliques with size 4 or above

```
$ mclique.rb i=dat1.csv f=node1,node2 l=4 o=result2.csv
/Users/stephane/.rvm/gems/ruby-1.9.3-p448/gems/nysol-1.5-x86_64-darwin/lib/nysol/margs.rb:
154:in ‘block in initialize’: I don’t know such a argument: ‘i=’ (RuntimeError)
        from /Users/stephane/.rvm/gems/ruby-1.9.3-p448/gems/nysol-1.5-x86_64-darwin/lib/nysol/mar
gs.rb:152:in ‘each’
        from /Users/stephane/.rvm/gems/ruby-1.9.3-p448/gems/nysol-1.5-x86_64-darwin/lib/nysol/mar
gs.rb:152:in ‘initialize’
        from /Users/stephane/.rvm/gems/ruby-1.9.3-p448/gems/nysol-1.5-x86_64-darwin/bin/mclique.r
b:124:in ‘new’
        from /Users/stephane/.rvm/gems/ruby-1.9.3-p448/gems/nysol-1.5-x86_64-darwin/bin/mclique.r
b:124:in ‘<top (required)>’
        from /Users/stephane/.rvm/rubies/ruby-1.9.3-p448/bin/mclique.rb:23:in ‘load’
        from /Users/stephane/.rvm/rubies/ruby-1.9.3-p448/bin/mclique.rb:23:in ‘<main>’
$ more result2.csv
result2.csv: No such file or directory
```

Example 3: Enumerate all cliques with size of 3

```
$ mclique.rb i=dat1.csv f=node1,node2 l=3 u=3 -all o=result3.csv
/Users/stephane/.rvm/gems/ruby-1.9.3-p448/gems/nysol-1.5-x86_64-darwin/lib/nysol/margs.rb:
154:in ‘block in initialize’: I don’t know such a argument: ‘i=’ (RuntimeError)
        from /Users/stephane/.rvm/gems/ruby-1.9.3-p448/gems/nysol-1.5-x86_64-darwin/lib/nysol/mar
gs.rb:152:in ‘each’
        from /Users/stephane/.rvm/gems/ruby-1.9.3-p448/gems/nysol-1.5-x86_64-darwin/lib/nysol/mar
gs.rb:152:in ‘initialize’
        from /Users/stephane/.rvm/gems/ruby-1.9.3-p448/gems/nysol-1.5-x86_64-darwin/bin/mclique.r
b:124:in ‘new’
        from /Users/stephane/.rvm/gems/ruby-1.9.3-p448/gems/nysol-1.5-x86_64-darwin/bin/mclique.r
b:124:in ‘<top (required)>’
        from /Users/stephane/.rvm/rubies/ruby-1.9.3-p448/bin/mclique.rb:23:in ‘load’
        from /Users/stephane/.rvm/rubies/ruby-1.9.3-p448/bin/mclique.rb:23:in ‘<main>’
$ more result3.csv
result3.csv: No such file or directory
```

2.6 mclique2g.rb Generate Maximal Clique

This command enumerates nodes of each maximal clique [2] from general graph (this is referred to as "original graph" below) to create new graph (referred to as maximal clique graph). An edge is added to connect common nodes between cliques.

An example is shown as follows. Graph in Figure 2.22 contains 4 maximal cliques, Figure 2.23 shows the new graph generated from the original graph. Node 0 corresponds to clique $\{a, b, c, d\}$ in Figure 2.21, node 1 corresponds to $\{d, e, f\}$, and finally node 2 corresponds to $\{e, f, h\}$. Similarly, node 3 corresponds to $\{e, f, h\}$.

Node 0 and 1 of maximal clique graph has 1 $(d)commonnodewiththeoriginalgraph, thusanedgeisextendedtonode0and1, withawe$

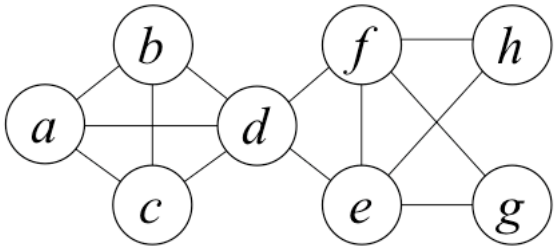

Figure 2.22: Contains 4 maximal cliques $\{a, b, c, d\}$ $\{d, e, f\}$ $\{e, f, g\}$ $\{e, f, h\}$.

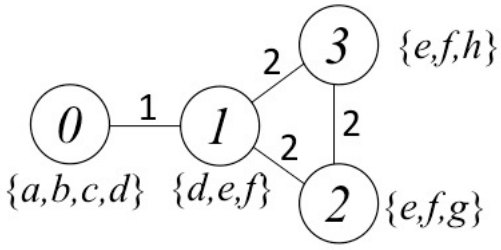

Figure 2.23: Newly generated maximal clique graph

The original graph input data of this command is shown in **??**, the data in CSV format, consisting of 2 columns, clique ID and node respectively. This data is corresponds to the data for mclique.rb command shown in 2.5 when `-node` is specified.

In addition, the output results of maximal cliques are created separately as node data (Table 2.32) and edge data (Table2.33). The column `weight` in the node data is the number of nodes that make up maximum clique. The column `weight` in the edge data is the number of nodes that make up maximum clique.

Table 2.31: Original graph

id	node
0	a
0	b
0	c
0	d
1	d
1	e
1	f
2	e
2	f
2	g
3	e
3	f
3	h

Table 2.32: Maximal clique graph (node data)

node	weight
0	4
1	3
2	3
3	3

Table 2.33: Maximal clique graph (edge data)

node1	node2	weight
0	1	1
1	2	2
1	3	2
2	3	2

[2] refer to section 2.5 on the definition of maximal clique.

2.6.1 Format

```
mclique2g.rb i= [id=] [f=] eo= no= [T=] [--help]

  Specification of file name
  i=  : Clique file name
id= : Clique ID field name (default:"id")
  f=  : Field name of node from the clique generated (default:"node")
  eo= : Edge output filename
  no= : Node output filename

  Others
  T= : Working directory (default:/tmp)
  --help : Show help information
```

2.6.2 Example

1: Basic Example

Example shown in the previous section.

```
$ more clique.csv
id,node
0,a
0,b
0,c
0,d
1,d
1,e
1,f
2,e
2,f
2,g
3,e
3,f
3,h
$ mclique2g.rb i=clique.csv eo=edge.csv no=node.csv id=id f=node
#END# /Users/stephane/.rvm/gems/ruby-2.0.0-p247/bin/mclique2g.rb i=clique.csv eo=edge.csv
no=node.csv id=id f=node
$ more edge.csv
node1%0,node2%1,weight
0,1,1
1,2,2
1,3,2
2,3,2
$ more node.csv
node%0,weight
0,4
1,3
2,3
3,3
```

2.7 mbiclique.rb - Enumerate Maximal Bipartite Cliques

This command enumerates maximal bipartite clique from bipartite graph. The `lcm` command [3] is used in mbiclique.rb. It is an bipartite graph represented as $G = (V_1 \cup V_2, E)$, with 2 nodes set V_1, V_2 and edge set connecting to the subsets $E \subset V_1 \times V_2$, any node in subgraph V_1, V_2 is connected with an edge in G is a directional subgraph.

In addition, if the bipartite clique is independent and does not exist in other bipartite cliques, it is referred to as maximal bipartite clique (Figure 2.24).

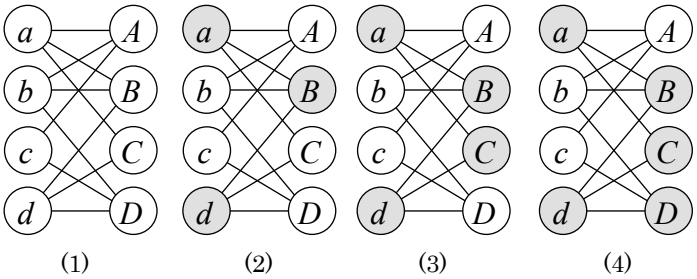

Figure 2.24: The shaded nodes in the subgraph from bipartite graph (1) is known as bipartite clique (2), (3) is a subset of bipartite clique, therefore it is not maximal bipartite clique. On the other hand, since (3) is not a subset of other bipartite clique, and thus it is maximal. In (4), there is no edge between a, D, thus it is not a bipartite clique.

Let's consider some real world application for maximal bipartite clique. For example, the set of panel data on consumer food purchasing consists of product information and shop information, a bipartite graph can be constructed by connecting product-shop information with an edge for purchases above a certain threshold. By enumerating maximal bipartite clique, it is possible to group products and shops with strong relationships.

This is regarded as particle and noun phrases in text mining. In application, bipartite graph is constructed of particle-noun phrases connected by an edge. The basic concept is to enumerate maximal bipartite clique, case frame (particle-noun phrase) with strong relationships will be grouped together.

When applying this concept in auction data, when the bidder and artifact is taken into consideration, bipartite graph can be constructed by adding edges between bidder-artifact whenever there is a bid.

The input data for bipartite graph used for the command is shown in 2.34, edge data is represented in node pairs in CSV format (corresponds to Figure 2.24 (1)). Each item represents a section.

In Table 2.34, in column `node1`, the lower-case letters represent the elements, in column `node2`, the upper-case letters represent elements in other parts.

The graph is treated as undirected graph and may have more than one island.

The output of maximal bipartite cliques enumerated from the bipartite graph is shown in Table 2.35.

One row corresponds to maximal bipartite clique, the elements that created each section is saved in column `node1,node2` in vector format. The size (number of nodes) of each section is saved in `size1,size2`.

Figure 2.24(3) corresponds to maximal bipartite clique in row 5 ($V_1 = \{a, d\}, V_2 = \{B, C\}$).

2.7.1 Format

```
Format) mbiclique.rb i= f= [o=] [l=] [u=] [T=] [-debug] [--help]

  Specify File Name
  i= :  Edge data file
  f= :  Column names of the 2 nodes in edge data
  o= : Output file
  l= : Minimal number of nodes constructing the maximal bipartite cliques.
```

Table 2.34: Input Data(Edge Data)

node1	node2
a	A
a	B
a	C
b	A
b	B
b	D
c	A
c	D
d	B
d	C
d	D

Table 2.35: Output Results

node1	node2	size1	size2
a	A B C	1	3
a b	A B	2	2
a b c	A	3	1
a b d	B	3	1
a d	B C	2	2
b	A B D	1	3
b c	A D	2	2
b c d	D	3	1
b d	B D	2	2
d	B C D	1	3

```
   : Maximal bipartite cliques smaller than the the value specified here will not be enumerated.
   : If the values are separated by comma, the size of each can be limited.
   : Order of values separated by comma corresponds to the order of items specified at f=.
   : When the limit is not defined, null character can be used  as in "l=2," and "l=,2".
   : Null character in the end can be omitted ("l=2," and "l=2" has the same meaning).
u= : Maximal number of nodes constructing maximal bipartite clique
   : Specification details is the same as I= parameter.

Others
T= : Working directory (default:/tmp)
--help : Help information
```

2.7.2 Notes

The output format of bipartite clique in this command is in vector format (a CSV column with character string delimited by space), sometimes the length of one field can become very long. Thus, an error may return if the length of one row of CSV data exceeds the maximum limit when processing the data within MCMD command. In this case, the error can be avoided by setting constraints for the bipartite clique at `l=` and `u=`.

2.7.3 Examples

Example 1: Basic Example

Example illustrated from the previous section.

```
$ more dat.csv
node1,node2
a,A
a,B
a,C
b,A
b,B
b,D
c,A
c,D
d,B
d,C
d,D
$ mbiclique.rb i=dat.csv f=node1,node2 o=result1.csv
#MSG# converting paired form into transaction form ...
#MSG# lcm_20140215 CIf /tmp/__MTEMP_20452_70343276945380_0 1 /tmp/__MTEMP_20452_7034327694
5380_3
trsact: /tmp/__MTEMP_20452_70343276945380_0 ,#transactions 4 ,#items 4 ,size 11 extracted
database: #transactions 4 ,#items 4 ,size 11
 output to: /tmp/__MTEMP_20452_70343276945380_3
separated at 0
iters=11
11
```

```
1
3
4
3
#END# /Users/stephane/.rvm/rubies/ruby-1.9.3-p448/bin/mbiclique.rb i=dat.csv f=node1,node2
 o=result1.csv
$ more result1.csv
node1,node2,size1,size2
a,A B C,1,3
a b,A B,2,2
a b c,A,3,1
a b d,B,3,1
a d,B C,2,2
b,A B D,1,3
b c,A D,2,2
b c d,D,3,1
b d,B D,2,2
d,B C D,1,3
```

Example 2: Example with size limit

Enumerate maximal bipartite clique with size of 2 in columns `node1,node2`.

```
$ mbiclique.rb i=dat.csv f=node1,node2 o=result2.csv l=2,2 u=2,2
#MSG# converting paired form into transaction form ...
#MSG# lcm_20140215 CIf -l 2 -u 2 /tmp/__MTEMP_20505_70100128947220_0 1 /tmp/__MTEMP_20505_
70100128947220_3
trsact: /tmp/__MTEMP_20505_70100128947220_0 ,#transactions 4 ,#items 4 ,size 11 extracted
database: #transactions 4 ,#items 4 ,size 11
 output to: /tmp/__MTEMP_20505_70100128947220_3
separated at 0
iters=10
4
0
0
4
#END# /Users/stephane/.rvm/rubies/ruby-1.9.3-p448/bin/mbiclique.rb i=dat.csv f=node1,node2
 o=result2.csv l=2,2 u=2,2
$ more result2.csv
node1,node2,size1,size2
a b,A B,2,2
a d,B C,2,2
b c,A D,2,2
b d,B D,2,2
```

Example 3: Example to limit the partial size

Enumerate maximal bipartite clique where column `node1` with lower limit of 1 (Since the default lower limit is 1, this example does not reflect additional meaning), and column `node2` has a upper limit of 3. The first value at `u=` parameter is null, since the upper limit of column `node1`.

```
$ mbiclique.rb i=dat.csv f=node1,node2 o=result3.csv l=1, u=,3
#MSG# converting paired form into transaction form ...
#MSG# lcm_20140215 CIf -u 3 /tmp/__MTEMP_20558_70109238975520_0 1 /tmp/__MTEMP_20558_70109
238975520_3
trsact: /tmp/__MTEMP_20558_70109238975520_0 ,#transactions 4 ,#items 4 ,size 11 extracted
database: #transactions 4 ,#items 4 ,size 11
 output to: /tmp/__MTEMP_20558_70109238975520_3
separated at 0
iters=11
11
1
3
4
3
#END# /Users/stephane/.rvm/rubies/ruby-1.9.3-p448/bin/mbiclique.rb i=dat.csv f=node1,node2
 o=result3.csv l=1, u=,3
$ more result3.csv
node1,node2,size1,size2
a,A B C,1,3
a b,A B,2,2
a b c,A,3,1
a b d,B,3,1
```

```
a d,B C,2,2
b,A B D,1,3
b c,A D,2,2
b c d,D,3,1
b d,B D,2,2
d,B C D,1,3
```

2.8 mgdiff.rb - Graph Difference

This command returns the difference between two general graph. However, as of this version, this command can be used to compare presence and absence of edge in undirected graph.

2.8.1 Format

```
Format) mgdiff.rb i= f= m= [F=] [o=] [T=] [--help]

  Specify File Name
  i= : File name of graph in edge data (node pair)
  f= : Field name of the 2 nodes in edge data
  m= : File name of reference graph (return the difference with this graph)
  F= : Field name of 2 nodes connected to the edge  in the reference file  (this parameter is not required
  o= : Output file name
       Output the edge (node pair) from graph 1 data and graph 2 data based on the following values.
        Field name: content
        i    :  file name specified at i=  if there is connected pair in the row
       m    : file name specified at m= if there is connected pair in the row
        diff : Classification of state
                1: only exist in graph specified at i=
                0: exist in graph in both i=,m=
               -1: only exist in graph specified at m=
  T= : Working directory (default:/tmp)
  --help : Help information
```

2.8.2 Examples

Example 1: Basic Example

```
$ more g1.csv
node1,node2
a,b
b,c
c,d
$ more g2.csv
node1,node2
a,b
c,d
d,e
$ mgdiff.rb i=g1.csv m=g2.csv o=result1.csv f=node1,node2
#END# /Users/stephane/.rvm/rubies/ruby-1.9.3-p448/bin/mgdiff.rb i=g1.csv m=g2.csv o=result
1.csv f=node1,node2
$ more result1.csv
node1,node2,fld,i,m,diff
b,c,file,g1.csv,,-1
a,b,file,g1.csv,g2.csv,0
c,d,file,g1.csv,g2.csv,0
d,e,file,,g2.csv,1
```

2.9 mcliqueInfo.rb Display Information on Clique Enumeration

This command displays information on clique enumerated from mclique.rb command. The main purpose is to generate information on connection relationship between cliques.

Figure 2.25 shows the graph used for the illustration of mclique2g.rb command. 4 maximal cliques are obtained from this graph. For each of these four cliques, 5 types of information are generated as show in Table 2.36.

Table 2.36: Output Results of this Command

Output Column name	Description	Example of id=1($\{d, e, f\}$)
nSize	Number of nodes	3 (3 nodes d, e, f)
eSize	Number of edges (nSize∗(nSize-1)/2)	3 ($3 * 2/2 = 3$)
extNodes	Number of nodes with external connection	5 (Fig. 2.27 on right)
extEdges	Number of edges with external connection	7 (Fig. 2.27 on left)
extCliques	Number of Cliques with external connection	3 (Fig. 2.26)

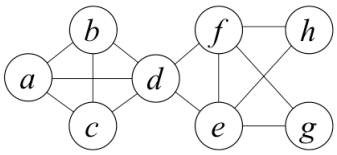

Figure 2.25: Graph of enumeration of maximal clique. Contains 4 maximal cliques {a,b,c,d}(id=0){d,e,f}(id=1){e,f,g}(id=2){e,f,h}(id=3).

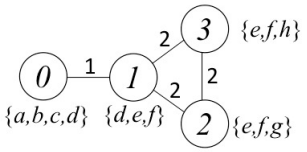

Figure 2.26: Maximal clique graph. The graph in Figure 2.25 shows clique id at the node. Edges with common nodes. Clique id=1 is directly connected to 3 other cliques. Refer to section 2.6 if you want to output to graph format.

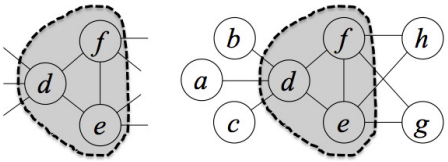

Figure 2.27: 7 branches connects from clique id=1 (region shown in grey) to other nodes outside (left figure). Clique id=1 (region shown in grey) are connected to 5 nodes (right figure).

Input data of this command is the same as the one used in mclique2g.rb command in section **??**, Table **??** shows the CSV formatted data consisting of two columns as cliqueID and node respectively. The output results are shown in Table **??**, and information of maximal clique is shown in Table 2.36.

2.9.1 Format

```
mcliqueInfo.rb i= [id=] [f=] [m=] [F=] o= [T=] [--help]
  i=     : File name of clique data
id=    : Field name of clique ID (default:"id")
  f=     : Field name of node in clique (default:"node")
  T= : Working directory (default:/tmp)
  --help : Show help information
```

Table 2.37: Input Data

id	node
0	a
0	b
0	c
0	d
1	d
1	e
1	f
2	e
2	f
2	g
3	e
3	f
3	h

Table 2.38: Output Results

id	nSize	eSize	extNodes	extEdges	extCliques
0	4	6	2	2	1
1	3	3	5	7	3
2	3	3	2	4	2
3	3	3	2	4	2

2.9.2 Example

1: Basic Example

Example illustrated in the previous section.

```
$ more clique.csv
id,node
0,a
0,b
0,c
0,d
1,d
1,e
1,f
2,e
2,f
2,g
3,e
3,f
3,h
$ mcliqueInfo.rb i=clique.csv o=result1.csv id=id f=node
#END# /Users/stephane/.rvm/gems/ruby-2.0.0-p247/bin/mcliqueInfo.rb i=clique.csv o=result1.
csv id=id f=node
$ more result1.csv
id,nSize,eSize,extNodes,extEdges,extCliques
0,4,6,2,2,1
1,3,3,5,7,3
2,3,3,2,4,2
3,3,3,2,4,2
```

Bibliography

[1] Takeaki Uno, Tatsuya Asai, Yuzo Uchida, Hiroki Arimura, "An Efficient Algorithm for Enumerating Closed Patterns in Transaction Databases", *Discovery Science 2004*, LNAI 3245, pp.16-31.

[2] , , , , 2014-AL-146(2), pp. 1-8, 2014.

[3] http://research.nii.ac.jp/ uno/codes-j.htm