Mining Frequent Patterns with Relative Support in Transactional Databases

What is frequent pattern mining with relative support?

Frequent pattern mining aims to discover all interesting patterns in a transactional database that have **support** no less than the user-specified **minimum support** (**minSup**) constraint and **Relative Support** no less than the user-specified **minimum relative support** (**minRSup**). The **minSup** controls the minimum number of transactions that a pattern must appear in a database. **minRSup** is the conditional minimum support calculated as ratio of pattern support to the minimum support of all items in pattern.

Reference : R. Uday Kiran and Masaru Kitsuregawa. 2012. Towards efficient discovery of frequent patterns with relative support. In Proceedings of the 18th International Conference on Management of Data (COMAD '12). Computer Society of India, Mumbai, Maharashtra, IND, 92–99.

What is a transactional database?

A transactional database is a collection of transactions, where each transaction contains a transaction-identifier and a set of items.
A hypothetical transactional database containing the items ***a, b, c, d, e, f, and g*** as shown below

tid	Transactions
1	a b c g
2	b c d e
3	a b c d
4	a c d f
5	a b c d g
6	c d e f
7	a b c d
8	a e f
9	a b c d
10	b c d e

Note: Duplicate items must not exist in a transaction.

What is acceptable format of transactional databases in PAMI?

Each row in a transactional database must contain only items. The frequent pattern mining algorithms in PAMI implicitly assume the row number of a transaction as its transactional-identifier to reduce storage and processing costs. A sample transactional database, say sampleInputFile.txt, is provided below.

a b c g
b c d e
a b c d
a c d f
a b c d g
c d e f
a b c d
a e f
a b c d
b c d e

Understanding the statisctics of database

To understand about the database. The below code will give the detail about the transactional database.

- Total number of transactions (Database size)
- Total number of unique items in database
- Minimum lenth of transaction that existed in database
- Average length of all transactions that exists in database
- Maximum length of transaction that existed in database
- Standard deviation of transaction length
- Variance in transaction length
- Sparsity of database

The below sample code prints the statistical details of a database.

In [ ]:
```
import PAMI.extras.dbStats.transactionalDatabaseStats as stats
obj = stats.transactionalDatabaseStats('sampleInputFile.txt', ' ')
obj.run()
obj.printStats()
```

What are the input parameters?

Algorithms to mine the frequent patterns with relative support requires transactional database and minSup (specified by user).

- Transactional database can be provided in following formats:
  - String : E.g., 'transactionalDatabase.txt'
  - URL : E.g., https://u-aizu.ac.jp/~udayrage/datasets/transactionalDatabases/transactional_T10
  - DataFrame. Please note that dataframe must contain the header titled 'Transactions'
- **minSup**
  specified in
  - **count (beween 0 to length of a database)** or
  - [0, 1]
- **minRatio**
  specified in
  - [0, 1]
- **seperator**
  default seperator is '\t' (tab space)

How to store the output of a frequent pattern mining algorithm?

The patterns discovered by a frequent pattern mining algorithm can be saved into a file or a data frame.

How to run the frequent pattern mining algorithms in a terminal?

- Download the PAMI source code from github.
- Unzip the PAMI source code folder and enter into frequent pattern using other measures folder.
- Enter into frequentPatternUsingOtherMeasures folder
- Enter into a specific folder of your choice and execute the following command on terminal.

syntax: python3 algorithmName.py `<path to the input file>` `<path to the output file>` `<minSup>` `<maxRatio>` `<seperator>`

Example: python3 `RSFPGrowth.py` `inputFile.txt` `outputFile.txt` `4` `0.7` `' '`

How to execute a frequent pattern mining algorithm in a Jupyter Notebook?

- Install the PAMI package from the PYPI repository by executing the following command: **pip3 install PAMI**
- Run the below sample code by making necessary changes

In [ ]:
```
import PAMI.frequentPatternUsingOtherMeasures.RSFPGrowth as alg

iFile = 'sampleInputFile.txt'  #specify the input transactional database
minSup = 5  #specify the minSupvalue <br>
minRatio = 0.6  #specify the minimum ratio <br>
seperator = ' ' #specify the seperator. Default seperator is tab space. <
oFile = 'frequentPatterns.txt'   #specify the output file name<br>

obj = alg.RSFPGrowth(iFile, minSup, minRatio, seperator) #initialize the
obj.startMine()                         #start the mining process <br>
obj.savePatterns(oFile)                 #store the patterns in file <br>
df = obj.getPatternsAsDataFrame()       #Get the patterns discovered into a
obj.printStats()                        #Print the statistics of mining pro
```

The frequentPatterns.txt file contains the following patterns (*format:* pattern:support):!cat frequentPatterns.txt

In [ ]:
```
!cat frequentPatterns.txt
```

e : 4 : 1.0
d : 8 : 1.0
d c : 8 : 1.0
c : 9 : 1.0
b : 7 : 1.0
b d : 6 : 0.8571428571428571
b d c : 6 : 0.8571428571428571
b c : 7 : 1.0
b a : 5 : 0.7142857142857143
a : 7 : 1.0
a c : 5 : 0.7142857142857143
a d : 5 : 0.7142857142857143
a c d : 5 : 0.7142857142857143

The dataframe containing the patterns is shown below:

In [ ]:
```
df
```

	Patterns	Support	RelativeSupport
0	e	4	1.000000
1	d	8	1.000000
2	d c	8	1.000000
3	c	9	1.000000
4	b	7	1.000000
5	b d	6	0.857143
6	b d c	6	0.857143
7	b c	7	1.000000
8	b a	5	0.714286
9	a	7	1.000000
10	a c	5	0.714286
11	a d	5	0.714286
12	a c d	5	0.714286