Mining Weighted Frequent Patterns in Transactional Databases

What is weighted frequent pattern mining?

weighted 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 **weight** no less than the user-specified **minimum weight** (***minWeight***). The **minSup** controls the minimum number of transactions that a pattern must appear in a database. The **minWeight** controls the minimum weight of item.

What is the 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 c d f i m
2	a c d f m r
3	b d f m p r
4	b c f m p
5	c d f m r
6	d m r

Note: Duplicate items must not exist in a transaction.

What is acceptable format of a 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 c d f i m
a c d f m r
b d f m p r
b c f m p
c d f m r
d m r

What is the Weighted database?

A weight database is a collection of items with their weights.
A hypothetical weight database containing the items ***a, b, c, d, e, f, and g*** as shown below

a 1.3
b 1.1
c 1.4
d 1.2
f 1.5
i 1.1
m 1.3
p 1.0
r 1.5

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()
```

The input parameters to a frequent pattern mining algorithm are:

- **Transactional database**

  Acceptable formats:

  > - String : E.g., 'transactionalDatabase.txt'
  > - URL : E.g., https://u-aizu.ac.jp/~udayrage/datasets/transactionalDatabases/transactional_T10
  > - DataFrame with the header titled 'Transactions'

- **minSup**

  specified in

  > - **count (beween 0 to length of a database)** or
  > - [0, 1]

- **minWeight**

  specified in

  > - **count (beween 0 to length of a database)** or
  > - [0, 1]

- **seperator**

  default seperator is '\t' (tab space)

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

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

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

- Download the PAMI source code from github.
- Unzip the PAMI source code folder and enter into weighted frequent pattern folder.
- Enter into weightedFrequentPattern folder
- Enter into a specific folder and execute the following command on terminal.

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

Sample command to execute the WFIM code in weightedFrequentPattern folder

Example: python3 `WFIM.py` `inputFile.txt` `outputFile.txt` `weightSample.txt` `3` `2` `' '`

How to execute a weighted 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.weightedFrequentPattern.WFIM as alg

iFile = 'WFIMSample.txt'  #specify the input transactional database <br>
wFile = 'WFIMWeightSample.txt'  #specify the input transactional database
minSup = 3  #specify the minSupvalue <br>
minWeight = 1.2    #specify the minWeight value <br>
seperator = ' ' #specify the seperator. Default seperator is tab space. <
oFile = 'weightedPatterns.txt'   #specify the output file name<br>

obj = alg.WFIM(iFile, wFile, minSup, minWeight, seperator) #initialize th
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 weightedPatterns.txt file contains the following patterns (*format:* pattern:support): !cat weightedPatterns.txt

In [2]:

```
!cat weightedPatterns.txt
```

```
r :4
r d :4
r d m :4
r m :4
c :4
c f :4
c f m :4
c m :4
f :5
f d :4
f d m :4
f m :5
d :5
d m :5
m :6
```

The dataframe containing the patterns is shown below:

In [3]:

```
df
```

Out[3]:

	Patterns	Support
0	r	4
1	r d	4
2	r d m	4
3	r m	4
4	c	4
5	c f	4
6	c f m	4
7	c m	4
8	f	5
9	f d	4
10	f d m	4
11	f m	5
12	d	5
13	d m	5
14	m	6