Mining Frequent Patterns in Transactional Databases

What is frequent pattern mining?

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. The **minSup** controls the minimum number of transactions that a pattern must appear in a database.

Reference: *Rakesh Agrawal, Tomasz Imieliński, and Arun Swami. 1993. Mining association rules between sets of items in large databases. In Proceedings of the 1993 ACM SIGMOD international conference on Management of data (SIGMOD '93). Association for Computing Machinery, New York, NY, USA, 207–216.* *https://doi.org/10.1145/170035.170072*

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 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 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?

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]

- **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 folder.
- Enter into frequentPattern folder
- You will find different types of folders like **basic, closed, maximal, topk, cuda, pyspark**
- 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>` `<seperator>`

Example: python3 `Apriori.py` `inputFile.txt` `outputFile.txt` `3` `' '`

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.frequentPattern.basic.Apriori as alg

iFile = 'sampleInputFile.txt'  #specify the input transactional database
minSup = 5  #specify the minSupvalue
seperator = ' ' #specify the seperator. Default seperator is tab space.
oFile = 'frequentPatterns.txt'   #specify the output file name

obj = alg.Apriori(iFile, minSup, seperator) #initialize the algorithm
obj.startMine()                          #start the mining process
obj.savePatterns(oFile)                  #store the patterns in file
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
```

b :7
b a :5
b a c :5
b d :6
b d c :6
b c :7
a :7
a d :5
a d c :5
a c :6
d :8
d c :8
c :9

The dataframe containing the patterns is shown below:

In [ ]:
```
df
```

	Patterns	Support
0	b	7
1	b a	5
2	b a c	5
3	b d	6
4	b d c	6
5	b c	7
6	a	7
7	a d	5
8	a d c	5
9	a c	6
10	d	8
11	d c	8
12	c	9