Extraction de motifs séquentiels

Dans ce notebook nous présentons différentes librairies d'extraction de motifs séquentiels. Comme pour les règles d'association, il en existe relativement peu en Python. Nous pouvons citer Aprioriall, Wincopper et PrefixSpan. Il existe des implémentations en Java.

L'objectif est juste d'illustrer le fonctionnement de ce type d'algorithmes. Nous ne les détaillons pas tous et nous focalisons sur Aprioriall et PrefixSpan. Le lecteur intéressé par Wincopper peut se reporter à : https://github.com/bitmapup/prefixspanr/ (https://github.com/bitmapup/prefixspanr/) Ce dernier offre des fonctionnalités intéressantes en intégrant notamment des contraintes sur les temps. Les implémentations en java posant problème avec Colab, elles ne sont pas traitées.

Installation

Avant de commencer, il est nécessaire de déjà posséder dans son environnement toutes les librairies utiles. Dans la seconde cellule nous importons toutes les librairies qui seront utiles à ce notebook. Il se peut que, lorsque vous lanciez l'éxecution de cette cellule, une soit absente. Dans ce cas il est nécessaire de l'installer. Pour cela dans la cellule suivante utiliser la commande :

! pip install nom_librairie

Attention : il est fortement conseillé lorsque l'une des librairies doit être installer de relancer le kernel de votre notebook.

Remarque : même si toutes les librairies sont importées dès le début, les librairies utiles pour des fonctions présentées au cours de ce notebook sont ré-importées de manière à indiquer d'où elles viennent et ainsi faciliter la réutilisation de la fonction dans un autre projet.

In [1]:
```
# utiliser cette cellule pour installer les librairies manquantes
# pour cela il suffit de taper dans cette cellule : !pip install no
m_librairie_manquante
# d'exécuter la cellule et de relancer la cellule suivante pour voi
r si tout se passe bien
# recommencer tant que toutes les librairies ne sont pas installées
...


#!pip install ..
!pip install prefixspan

# ne pas oublier de relancer le kernel du notebook
```

```
Requirement already satisfied: prefixspan in /usr/local/lib/python
3.7/dist-packages (0.5.2)
Requirement already satisfied: docopt>=0.6.2 in /usr/local/lib/pyt
hon3.7/dist-packages (from prefixspan) (0.6.2)
Requirement already satisfied: extratools>=0.8.1 in /usr/local/lib
/python3.7/dist-packages (from prefixspan) (0.8.2.1)
Requirement already satisfied: toolz>=0.9.0 in /usr/local/lib/pyth
on3.7/dist-packages (from extratools>=0.8.1->prefixspan) (0.11.1)
Requirement already satisfied: sortedcontainers>=1.5.10 in /usr/lo
cal/lib/python3.7/dist-packages (from extratools>=0.8.1->prefixspa
n) (2.4.0)
```

In [2]:
```
# Importation des différentes librairies utiles pour le notebook

#Sickit learn met régulièrement à jour des versions et
#indique des futurs warnings.
#ces deux lignes permettent de ne pas les afficher.
import warnings
warnings.filterwarnings("ignore", category=FutureWarning)
warnings.filterwarnings("ignore", category=DeprecationWarning)
warnings.filterwarnings('ignore','SelectableGroups dict interface')

with warnings.catch_warnings():
    warnings.simplefilter('ignore')
    # do something here and its warning is suppressed

# librairies générales
import pandas as pd

import numpy as np
```

Pour pouvoir sauvegarder sur votre répertoire Google Drive, il est nécessaire de fournir une autorisation.
Pour cela il suffit d'éxecuter la ligne suivante et de saisir le code donné par Google.

In [3]:
```
# pour monter son drive Google Drive local
from google.colab import drive
drive.mount('/content/gdrive')
```

```
Drive already mounted at /content/gdrive; to attempt to forcibly r
emount, call drive.mount("/content/gdrive", force_remount=True).
```

Corriger éventuellement la ligne ci-dessous pour mettre le chemin vers un répertoire spécifique dans votre répertoire Google Drive :

In [4]:
```
import sys
my_local_drive='/content/gdrive/My Drive/Colab Notebooks/ML_FDS'
# Ajout du path pour les librairies, fonctions et données
sys.path.append(my_local_drive)
# Se positionner sur le répertoire associé
%cd $my_local_drive

%pwd
```

```
/content/gdrive/My Drive/Colab Notebooks/ML_FDS
```

Out[4]:
```
'/content/gdrive/My Drive/Colab Notebooks/ML_FDS'
```

The Aprioriall algorithm

In [5]:
```
import aprioriall
```

The format of the dataset is:

- An event is a list of strings.
- A sequence is a list of events.
- A dataset is a list of sequences.

Example:

```
dataset =  [
      [["a"], ["a", "b", "c"], ["a", "c"], ["c"]],
      [["a"], ["c"], ["b", "c"]],
      [["a", "b"], ["d"], ["c"], ["b"], ["c"]],
      [["a"], ["c"], ["b"], ["c"]]
]
```

In [6]:
```
dataset =  [
    [["a"], ["a", "b", "c"], ["a", "c"], ["c"]],
    [["a"], ["c"], ["b", "c"]],
    [["a", "b"], ["d"], ["c"], ["b"], ["c"]],
    [["a"], ["a", "b", "c"], ["a", "c"], ["c"]],
    [["a"], ["c"], ["b", "c"]],
    [["a", "b"], ["d"], ["c"], ["b"], ["c"]],
    [["a"], ["a", "b", "c"], ["a", "c"], ["c"]],
    [["a"], ["c"], ["b", "c"]],
    [["a", "b"], ["d"], ["c"], ["b"], ["c"]],
    [["a"], ["a", "b", "c"], ["a", "c"], ["c"]],
    [["a"], ["c"], ["b", "c"]],
    [["a", "b"], ["d"], ["c"], ["b"], ["c"]],
    [["a"], ["a", "b", "c"], ["a", "c"], ["c"]],
    [["a"], ["c"], ["b", "c"]],
    [["a", "b"], ["d"], ["c"], ["b"], ["c"]],
    [["a"], ["a", "b", "c"], ["a", "c"], ["c"]],
    [["a"], ["c"], ["b", "c"]],
    [["a", "b"], ["d"], ["c"], ["b"], ["c"]],
    [["a"], ["a", "b", "c"], ["a", "c"], ["c"]],
    [["a"], ["c"], ["b", "c"]],
    [["a", "b"], ["d"], ["c"], ["b"], ["c"]],
    [["a"], ["c"], ["b"], ["c"]]
]
```

In [7]:
```
dataset =  [
    [["R1"], ["G2", "R1"], ["R1"]],
    [["e"], ["a"], ["e"], ["b","c"], ["f"], ["d"]],
    [["h"], ["h", "i"], ["j"]],
    [["h"], ["i"], ["j"], ["k"]]
]

result = aprioriall.apriori(dataset, 2, verbose=False)
aprioriall.filterMaximal(result)
print(result)
result = aprioriall.apriori(dataset, 2, verbose=False)
aprioriall.filterClosed(result)
print(result)
```

```
Result, lvl 1: [([['h']], 2), ([['i']], 2), ([['j']], 2)]
[([['h'], ['i'], ['j']], 2)]
Result, lvl 1: [([['h']], 2), ([['i']], 2), ([['j']], 2)]
[([['h'], ['i'], ['j']], 2)]
```

Running aprioriall: aprioriall.apriori (nameofthedataset, support=number of minimal occurrences, verbose={false/true})

In [8]:
```
aprioriall.apriori(dataset, 2, verbose=False)
```

```
Result, lvl 1: [([['h']], 2), ([['i']], 2), ([['j']], 2)]
```

Out[8]:
```
[([['h']], 2),
 ([['i']], 2),
 ([['j']], 2),
 ([['h'], ['i']], 2),
 ([['h'], ['j']], 2),
 ([['i'], ['j']], 2),
 ([['h'], ['i'], ['j']], 2)]
```

Get the maximal sequential patterns

In [9]:
```
result = aprioriall.apriori(dataset, 2, verbose=False)
aprioriall.filterMaximal(result)
print(result)
```

```
Result, lvl 1: [([['h']], 2), ([['i']], 2), ([['j']], 2)]
[([['h'], ['i'], ['j']], 2)]
```

Get the closed sequential patterns

In [10]:
```
result = aprioriall.apriori(dataset, 2, verbose=False)
aprioriall.filterClosed(result)
print(result)
```

```
Result, lvl 1: [([['h']], 2), ([['i']], 2), ([['j']], 2)]
[([['h'], ['i'], ['j']], 2)]
```

The PrefixSpan algorithm

In [11]:
```
import prefixspan
from prefixspan import PrefixSpan
```

In [12]:
```
dataset =  [
    [0, 1, 2, 3, 4],
    [1, 1, 1, 3, 4],
    [2, 1, 2, 2, 0],
    [1, 1, 1, 2, 2]
]
```

In [13]:
```
# create a prefixspan object
ps = PrefixSpan(dataset)
```

Get the sequential patterns

In [14]:
```
print(ps.frequent(2))
```

```
[(2, [0]), (4, [1]), (3, [1, 2]), (2, [1, 2, 2]), (2, [1, 3]), (2,
[1, 3, 4]), (2, [1, 4]), (2, [1, 1]), (2, [1, 1, 1]), (3, [2]), (2
, [2, 2]), (2, [3]), (2, [3, 4]), (2, [4])]
```

Get the top-k

In [15]:
```
print(ps.topk(5))
```

```
[(4, [1]), (3, [1, 2]), (3, [2]), (2, [1, 3]), (2, [1, 3, 4])]
```

Get the closed sequential patterns

In [16]:
```
print(ps.frequent(2, closed=True))
```

```
[(2, [0]), (4, [1]), (3, [1, 2]), (2, [1, 2, 2]), (2, [1, 3, 4]),
(2, [1, 1, 1])]
```