In [1]:

```
# -*- coding: utf-8 -*-
import numpy as np
import pandas as pd
def connect_string(x, ms):
    x = list(map(lambda i: sorted(i.split(ms)), x))
    l = len(x[0])
    r = []
    for i in range(len(x)):
        for j in range(i, len(x)):
            if x[i][:l - 1] == x[j][:l - 1] and x[i][l - 1] != x[j][l - 1]:
                r.append(x[i][:l - 1] + sorted([x[j][l - 1], x[i][l - 1]]))
    return r
# 寻找关联规则的函数
def find_rule(d, support, confidence, ms=u'--'):
    result = pd.DataFrame(index=['support', 'confidence'])  # 定义输出结果
    support_series = 1.0 * d.sum() / len(d)  # 支持度序列
    column = list(support_series[support_series > support].index)  # 初步根据支持度筛选
    k = 0
    while len(column) > 1:
        k = k + 1
        print(u'\n正在进行第%s次搜索...' % k)
        column = connect_string(column, ms)
        print(u'数目：%s...' % len(column))
        sf = lambda i: d[i].prod(axis=1, numeric_only=True)  # 新一批支持度的计算函数

        # 创建连接数据，这一步耗时、耗内存最严重。当数据集较大时，可以考虑并行运算优化。
        d_2 = pd.DataFrame(list(map(sf, column)), index=[ms.join(i) for i in column]).T

        support_series_2 = 1.0 * d_2[[ms.join(i) for i in column]].sum() / len(d)  # 计算连接后的
        column = list(support_series_2[support_series_2 > support].index)  # 新一轮支持度筛选
        support_series = support_series.append(support_series_2)
        column2 = []

        for i in column:  # 遍历可能的推理，如{A,B,C}究竟是A+B-->C还是B+C-->A还是C+A-->B?
            i = i.split(ms)
            for j in range(len(i)):
                column2.append(i[:j] + i[j + 1:] + i[j:j + 1])

        cofidence_series = pd.Series(index=[ms.join(i) for i in column2])  # 定义置信度序列

        for i in column2:  # 计算置信度序列
            cofidence_series[ms.join(i)] = support_series[ms.join(sorted(i))] / support_series[ms

        for i in cofidence_series[cofidence_series > confidence].index:  # 置信度筛选
            result[i] = 0.0
            result[i]['confidence'] = cofidence_series[i]
            result[i]['support'] = support_series[ms.join(sorted(i.split(ms)))]
    result = result.T.sort_values(['confidence', 'support'], ascending=False)  # 结果整理，输出
    print(u'\n结果为：')
    print(result)
    return result
```

In [2]:

```
from __future__ import print_function
# 读取数据
user_goods = pd.read_excel("data/goods_new.xls", header = None)
print('\n转换原始数据至0-1矩阵')
ct = lambda x:pd.Series(1, index = x[pd. notnull (x)]) #转换0-1矩阵的过渡函数
b= map(ct, user_goods.values) #用map方式执行
data= pd.DataFrame (list(b)).fillna(0) #实现矩阵转换,空值用0填充
print('\n转换完毕')
del b # 删除中间变量b,节省内存
support = 0.2 #最小支持度
confidence = 0.5 #最小置信度
ms='---'
#连接符,用来区分不同元素
#关联规则分析并写出结果
apriori_result = find_rule(data, support, confidence, ms)
apriori_result = apriori_result.round(3)
apriori_result.to_excel('apriori_result.xls')
```

```
转换原始数据至0-1矩阵

转换完毕

正在进行第1次搜索...
数目：10...

正在进行第2次搜索...
数目：5...

正在进行第3次搜索...
数目：0...

结果为：
                       support  confidence
Beer---Diaper              0.6    1.000000
Coke---Milk                0.4    1.000000
Coke---Diaper              0.4    1.000000
Beer---Bread---Diaper      0.4    1.000000
Beer---Milk---Diaper       0.4    1.000000
Coke---Milk---Diaper       0.4    1.000000
Coke---Diaper---Milk       0.4    1.000000
Milk---Bread               0.6    0.750000
Bread---Milk               0.6    0.750000
Diaper---Bread             0.6    0.750000
Bread---Diaper             0.6    0.750000
Milk---Diaper              0.6    0.750000
Diaper---Milk              0.6    0.750000
Diaper---Beer              0.6    0.750000
Beer---Bread               0.4    0.666667
Beer---Milk                0.4    0.666667
Diaper---Milk---Bread      0.4    0.666667
Bread---Milk---Diaper      0.4    0.666667
Bread---Diaper---Milk      0.4    0.666667
Bread---Diaper---Beer      0.4    0.666667
Beer---Diaper---Bread      0.4    0.666667
Diaper---Milk---Beer       0.4    0.666667
Beer---Diaper---Milk       0.4    0.666667
Diaper---Milk---Coke       0.4    0.666667
```

```
<ipython-input-1-0ee5fa6beec3>:39: DeprecationWarning: The default dtype for empty
Series will be 'object' instead of 'float64' in a future version. Specify a dtype
explicitly to silence this warning.
  cofidence_series = pd.Series(index=[ms.join(i) for i in column2])  # 定义置信度
序列
<ipython-input-1-0ee5fa6beec3>:39: DeprecationWarning: The default dtype for empty
Series will be 'object' instead of 'float64' in a future version. Specify a dtype
explicitly to silence this warning.
  cofidence_series = pd.Series(index=[ms.join(i) for i in column2])  # 定义置信度
序列
<ipython-input-1-0ee5fa6beec3>:39: DeprecationWarning: The default dtype for empty
Series will be 'object' instead of 'float64' in a future version. Specify a dtype
explicitly to silence this warning.
  cofidence_series = pd.Series(index=[ms.join(i) for i in column2])  # 定义置信度
序列
<ipython-input-2-cccc261986e1>:17: FutureWarning: As the xlwt package is no longer
maintained, the xlwt engine will be removed in a future version of pandas. This is
the only engine in pandas that supports writing in the xls format. Install openpyx
l and write to an xlsx file instead. You can set the option io.excel.xls.writer to
'xlwt' to silence this warning. While this option is deprecated and will also rais
e a warning, it can be globally set and the warning suppressed.
  apriori_result.to_excel('apriori_result.xls')
```

In [ ]: