报告

学号:3019213043　　姓名:刘京宗　　班级:软工 5 班

1. 目标

练习使用关联规则 Apriori 算法。完成以下两个任务，并与作业里手动计算的结果进行对比分析。

2. 数据

课堂收集的选课的真实数据和 DBLP 数据集

任务 1

选择以下某一数据进行关联规则的抽取

序号	性别	囤货表（食物）
1	女	自热火锅 奶粉 燕麦片 饮用水
2	女	薯片 酸奶 椰汁 汽水 烧卖
3	女	泡面 水 牛奶 面包
4	女	方便面 火锅 饼干 鲜奶 巧克力
5	女	方便面 面包 水 巧克力
6	男	方便面 矿泉水 面包
7	男	巧克力 方便面 面包 矿泉水 肉制品 辣条
8	男	泡面 矿泉水 蛋糕 蔬菜水果
9	男	方便面 面包 饮料 银耳羹
10	男	方便面 矿泉水 面包 巧克力 牛奶

执行 Apriori 算法，记录算法设置和结果，要求：

1) 给出算法过程，记录参数设置。

2) 分析结果，找出不同频繁项数量（2-4）的关联规则结果。并给出相应的算法设置。

3) 要求给出重要的算法，过程截图，和必要的文字分析。

（1）统一表格中食物的名称，如将‘水’改为‘矿泉水’，将‘泡面’改为‘方便面’等

序号	性别	囤货表（食物）
1	女	火锅 奶粉 燕麦片 矿泉水
2	女	薯片 酸奶 椰汁 汽水 烧卖
3	女	方便面 矿泉水 牛奶 面包
4	女	方便面 火锅 饼干 鲜奶 巧克力
5	女	方便面 面包 矿泉水 巧克力
6	男	方便面 矿泉水 面包
7	男	巧克力 方便面 面包 矿泉水 肉制品 辣条
8	男	方便面 矿泉水 蛋糕 蔬菜水果
9	男	方便面 面包 饮料 银耳羹
10	男	方便面 矿泉水 面包 巧克力 牛奶

（2）编写 python 程序

```
# coding=utf-8

"""
实现 Apriori 算法，并采用所写程序提取购物篮数据中的 频繁项集 和 强关联规则
```

```
参数设置：最小支持度计数 = 2，最小置信度 = 0.6
"""
min_sup = 2
min_conf = 0.6
# 最大 K 项集
K = 4



# apriori 算法
def apriori():
    # 1.读入数据
    data_set = load_data()
    # 2.计算每项的支持数
    C1 = create_C1(data_set)
    item_count = count_itemset1(data_set, C1)
    # 3.剪枝，去掉支持数小于最小支持度数的项
    L1 = generate_L1(item_count)
    # 4.连接
    # 5.扫描前一个项集，剪枝
    # 6.计数，剪枝
    # 7.重复 4-6，直到得到最终的 K 项频繁项集
    Lk_copy = L1.copy()
    L = []
    L.append(Lk_copy)
    for i in range(2, K + 1):
        Ci = create_Ck(Lk_copy, i)
        Li = generate_Lk_by_Ck(Ci, data_set)
        Lk_copy = Li.copy()
        L.append(Lk_copy)
    # 8.输出频繁项集及其支持度数
    print('频繁项集\t 支持度计数')
    support_data = {}
    for item in L:
        for i in item:
            print(list(i), '\t', item[i])
            support_data[i] = item[i]
    # 9.对每个关联规则计算置信度，保留大于最小置信度的频繁项为 强关联规则
    strong_rules_list = generate_strong_rules(L, support_data, data_set)
    strong_rules_list.sort(key=lambda result: result[2], reverse=True)
    print("\nStrong association rule\nX\t\t\tY\t\tconf")
    for item in strong_rules_list:
        print(list(item[0]), "\t", list(item[1]), "\t %.2f" % (item[2]))
```

```
# 读入数据
def load_data():
    # 事务 ID 购买商品
    # data = {'001': '火锅 奶粉 燕麦片 矿泉水', '002': '薯片 酸奶 椰汁 汽水 烧卖',
    #         '003': '方便面 矿泉水 牛奶 面包', '004': '方便面 火锅 饼干 鲜奶 巧克力',
    #         '005': '方便面 面包 矿泉水 巧克力'}
    data = {'006': '方便面 矿泉水 面包',
            '007': '巧克力 方便面 面包 矿泉水 肉制品 辣条', '008': '方便面 矿泉水 蛋糕 蔬菜水果',
            '009': '方便面 面包 饮料 银耳羹', '010': '方便面 矿泉水 面包 巧克力 牛奶'}
    data_set = []
    for key in data:
        item = data[key].split(' ')
        data_set.append(item)
    return data_set


# 构建 1-项集
def create_C1(data_set):
    C1 = set()
    for t in data_set:
        for item in t:
            item_set = frozenset([item])
            C1.add(item_set)
    return C1


# 计算给定数据每项及其支持数，第一次
def count_itemset1(data_set, C1):
    item_count = {}
    for data in data_set:
        for item in C1:
            if item.issubset(data):
                if item in item_count:
                    item_count[item] += 1
                else:
                    item_count[item] = 1
    return item_count


# 生成剪枝后的 L1
def generate_L1(item_count):
    L1 = {}
```

```
    for i in item_count:
        if item_count[i] >= min_sup:
            L1[i] = item_count[i]
    return L1


# 判断是否该剪枝
def is_apriori(Ck_item, Lk_copy):
    for item in Ck_item:
        sub_Ck = Ck_item - frozenset([item])
        if sub_Ck not in Lk_copy:
            return False
    return True


# 生成 k 项商品集，连接操作
def create_Ck(Lk_copy, k):
    Ck = set()
    len_Lk_copy = len(Lk_copy)
    list_Lk_copy = list(Lk_copy)
    for i in range(len_Lk_copy):
        for j in range(1, len_Lk_copy):
            l1 = list(list_Lk_copy[i])
            l2 = list(list_Lk_copy[j])
            l1.sort()
            l2.sort()
            if l1[0:k - 2] == l2[0:k - 2]:
                Ck_item = list_Lk_copy[i] | list_Lk_copy[j]
                # 扫描前一个项集，剪枝
                if is_apriori(Ck_item, Lk_copy):
                    Ck.add(Ck_item)
    return Ck


# 生成剪枝后的 Lk
def generate_Lk_by_Ck(Ck, data_set):
    item_count = {}
    for data in data_set:
        for item in Ck:
            if item.issubset(data):
                if item in item_count:
                    item_count[item] += 1
                else:
                    item_count[item] = 1
```

```
    Lk2 = {}
    for i in item_count:
        if item_count[i] >= min_sup:
            Lk2[i] = item_count[i]
    return Lk2


# 产生强关联规则
def generate_strong_rules(L, support_data, data_set):
    strong_rule_list = []
    sub_set_list = []
    # print(L)
    for i in range(0, len(L)):
        for freq_set in L[i]:
            for sub_set in sub_set_list:
                if sub_set.issubset(freq_set):
                    # 计算包含 X 的交易数
                    sub_set_num = 0
                    for item in data_set:
                        if (freq_set - sub_set).issubset(item):
                            sub_set_num += 1
                    conf = support_data[freq_set] / sub_set_num
                    strong_rule = (freq_set - sub_set, sub_set, conf)
                    if conf >= min_conf and strong_rule not in strong_rule_list:
                        # print(list(freq_set-sub_set), " => ", list(sub_set), "conf: ", conf)
                        strong_rule_list.append(strong_rule)
            sub_set_list.append(freq_set)
    return strong_rule_list


if __name__ == '__main__':
    # 运行 Apriori 算法
    apriori()
```

（3）给出分析结果：

3.1 女生结果：

频繁项集	支持度计数
['矿泉水']	3
['火锅']	2
['面包']	2
['方便面']	3
['巧克力']	2

['方便面', '面包']　　2
['面包', '矿泉水']　　2
['方便面', '矿泉水']　2
['巧克力', '方便面']　2
['方便面', '面包', '矿泉水']　　2

Strong association rule
X　　Y　　conf
['面包']　　['方便面']　1.00
['面包']　　['矿泉水']　1.00
['巧克力']　['方便面']　1.00
['方便面', '面包']　　['矿泉水']　1.00
['方便面', '矿泉水']　['面包']　　1.00
['面包', '矿泉水']　　['方便面']　1.00
['面包']　　['方便面', '矿泉水']　　1.00
['方便面']　['面包']　　0.67
['矿泉水']　['面包']　　0.67
['方便面']　['矿泉水']　0.67
['矿泉水']　['方便面']　0.67
['方便面']　['巧克力']　0.67
['矿泉水']　['方便面', '面包']　　0.67
['方便面']　['面包', '矿泉水']　　0.67

3.2 男生结果：

频繁项集　支持度计数
['矿泉水']　4
['面包']　　4
['方便面']　5
['巧克力']　2
['面包', '矿泉水']　　3
['面包', '方便面']　　4
['方便面', '矿泉水']　4
['方便面', '巧克力']　2
['矿泉水', '巧克力']　2
['面包', '巧克力']　　2
['面包', '矿泉水', '方便面']　　3
['方便面', '矿泉水', '巧克力']　　2
['面包', '方便面', '巧克力']　　2
['面包', '矿泉水', '巧克力']　　2

['方便面', '矿泉水', '面包', '巧克力'] 2

Strong association rule

X	Y	conf
['面包']	['方便面']	1.00
['矿泉水']	['方便面']	1.00
['巧克力']	['方便面']	1.00
['巧克力']	['矿泉水']	1.00
['巧克力']	['面包']	1.00
['面包', '矿泉水']	['方便面']	1.00
['方便面', '巧克力']	['矿泉水']	1.00
['矿泉水', '巧克力']	['方便面']	1.00
['巧克力']	['方便面', '矿泉水']	1.00
['方便面', '巧克力']	['面包']	1.00
['面包', '巧克力']	['方便面']	1.00
['巧克力']	['面包', '方便面']	1.00
['面包', '巧克力']	['矿泉水']	1.00
['矿泉水', '巧克力']	['面包']	1.00
['巧克力']	['面包', '矿泉水']	1.00
['面包', '方便面', '巧克力']	['矿泉水']	1.00
['方便面', '矿泉水', '巧克力']	['面包']	1.00
['面包', '矿泉水', '巧克力']	['方便面']	1.00
['方便面', '巧克力']	['面包', '矿泉水']	1.00
['矿泉水', '巧克力']	['面包', '方便面']	1.00
['面包', '巧克力']	['方便面', '矿泉水']	1.00
['巧克力']	['面包', '矿泉水', '方便面']	1.00
['方便面']	['面包']	0.80
['方便面']	['矿泉水']	0.80
['面包']	['矿泉水']	0.75
['矿泉水']	['面包']	0.75
['面包', '方便面']	['矿泉水']	0.75
['方便面', '矿泉水']	['面包']	0.75
['矿泉水']	['面包', '方便面']	0.75
['面包']	['方便面', '矿泉水']	0.75
['面包', '矿泉水']	['巧克力']	0.67
['面包', '方便面', '矿泉水']	['巧克力']	0.67
['面包', '矿泉水']	['方便面', '巧克力']	0.67
['方便面']	['面包', '矿泉水']	0.60

（4）结果分析:

总体来看男生的频繁项集更多，支持度系数更高，这表明：相较于女生，男生在遇到疫情囤货时选择的食物趋于一致，相似性更高。在食物方面，矿泉水、方便面、面包在男女生中的支持度计数均较高，对食物种类的选择，男女生是相近的。

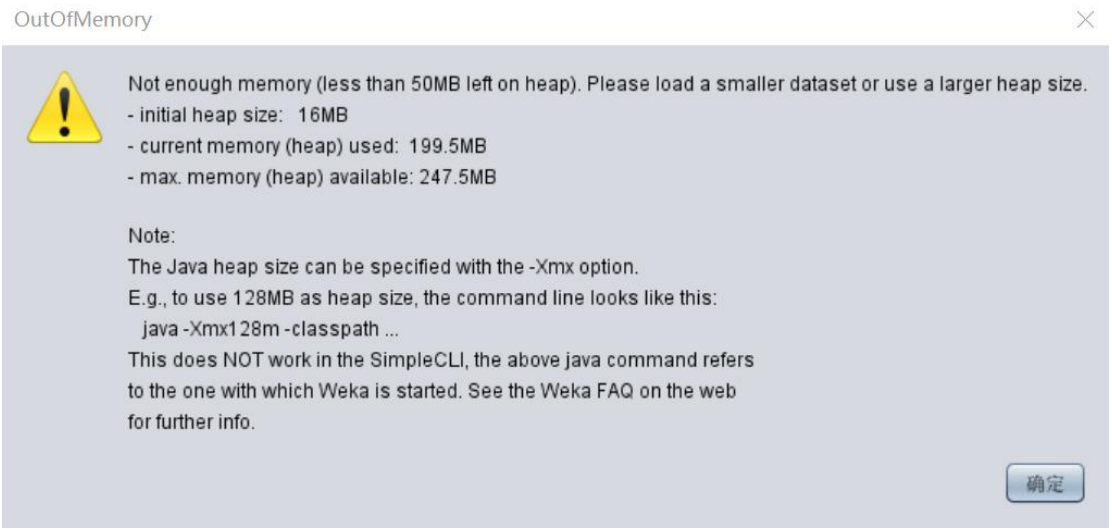
当然由于样本数较小，该结论可能不具有普遍性。

任务 2

使用 DBLP 数据集，提出一种方法，挖掘密切相关的（即经常一起合写文章）合著者关系。

（1）从智慧树上下载 dblpjson-csv.zip 文件，解压后先后运行 dblpxml-json.py，dblpjson-csv.py 文件，得到 out.csv 文件，其中存储了合著者信息。

（2）用 weka 打开 out.csv 文件，发现如下报错，在 RunWeka.ini 中修改 javaOpts=%JAVA_OPTS%为



javaOpts=%JAVA_OPTS% -Xmx2048m。问题得以解决。

（3）配置 Apriori 算法的参数并运行：

N：规则数
T：度量单位的选择，"0"值表示度量单位选为置信度
C：度量单位的最小值（此处指置信度）
D：递减迭代值
U：最小支持度上界
M：最小支持度下届
S：重要程度
c：类索引为c输出项集设为真

（4）由于数据量实在庞大，这里只选择了前 29999 行的数据，且我们发现多数数据的作者数不超过 5 人，因此同一作品只统计前 5 名作者。最终得出如下结果。

Associator output

```
=== Run information ===

Scheme:       weka.associations.Apriori -N 100 -T 0 -C 0.7 -D 0.05 -U 1.0 -M 2.5E-4 -S -1.0 -c -1
Relation:     out - 副本
Instances:    29999
Attributes:   5
              author1
              author2
              author3
              author4
              author5
=== Associator model (full training set) ===
```

```
Best rules found:

 1. author2=AlfredMenezes 22 ==> author1=DarrelHankerson 22    <conf:(1)> lift:(1363.59) lev:(0) [21] conv:(21.98)
 2. author1=DarrelHankerson 22 ==> author2=AlfredMenezes 22    <conf:(1)> lift:(1363.59) lev:(0) [21] conv:(21.98)
 3. author1=PhilippeBonnet 17 ==> author2=DennisE.Shasha 17    <conf:(1)> lift:(1666.61) lev:(0) [16] conv:(16.99)
 4. author1=XinJin0001 16 ==> author2=JiaweiHan0001 16    <conf:(1)> lift:(1249.96) lev:(0) [15] conv:(15.99)
 5. author1=JonasMellin 15 ==> author2=MikaelBerndtsson 15    <conf:(1)> lift:(1428.52) lev:(0) [14] conv:(14.99)
 6. author2=HenryLin 14 ==> author1=XiaoboZhou 14    <conf:(1)> lift:(2142.79) lev:(0) [13] conv:(13.99)
 7. author1=XiaoboZhou 14 ==> author2=HenryLin 14    <conf:(1)> lift:(2142.79) lev:(0) [13] conv:(13.99)
 8. author2=JonasMellin 13 ==> author1=MikaelBerndtsson 13    <conf:(1)> lift:(2307.62) lev:(0) [12] conv:(12.99)
 9. author1=MikaelBerndtsson 13 ==> author2=JonasMellin 13    <conf:(1)> lift:(2307.62) lev:(0) [12] conv:(12.99)
10. author2=YiZhang0001 12 ==> author1=EthanZhang 12    <conf:(1)> lift:(2499.92) lev:(0) [11] conv:(12)
11. author1=EthanZhang 12 ==> author2=YiZhang0001 12    <conf:(1)> lift:(2499.92) lev:(0) [11] conv:(12)
12. author1=AlexanderKaplan 11 ==> author2=RainerTichatschke 11    <conf:(1)> lift:(2307.62) lev:(0) [10] conv:(11)
13. author2=VictorKlee 10 ==> author1=PeterGritzmann 10    <conf:(1)> lift:(2499.92) lev:(0) [9] conv:(10)
14. author2=MarkS.Drew 10 ==> author1=RajeevRamanath 10    <conf:(1)> lift:(2999.9) lev:(0) [9] conv:(10)
15. author1=RajeevRamanath 10 ==> author2=MarkS.Drew 10    <conf:(1)> lift:(2999.9) lev:(0) [9] conv:(10)
16. author1=NikosHardavellas 10 ==> author2=IppokratisPandis 10    <conf:(1)> lift:(2307.62) lev:(0) [9] conv:(10)
17. author2=JamesB.D.Joshi 10 ==> author1=YueZhang0002 10    <conf:(1)> lift:(2999.9) lev:(0) [9] conv:(10)
18. author1=YueZhang0002 10 ==> author2=JamesB.D.Joshi 10    <conf:(1)> lift:(2999.9) lev:(0) [9] conv:(10)
19. author2=EricC.Jensen 10 ==> author1=StevenM.Beitzel 10    <conf:(1)> lift:(2999.9) lev:(0) [9] conv:(10)
20. author1=StevenM.Beitzel 10 ==> author2=EricC.Jensen 10    <conf:(1)> lift:(2999.9) lev:(0) [9] conv:(10)
21. author1=StevenM.Beitzel 10 ==> author3=OphirFrieder 10    <conf:(1)> lift:(2142.79) lev:(0) [9] conv:(10)
```