

《数据挖掘》

实验报告本

|  |  |
| --- | --- |
| 班 级： | **计金（双）200** |
| 学 号： | **20002530** |
| 姓 名： | **高明源** |
| 指导教师： | **张静** |

信息科学与工程学院

2023年 4月

**实验一 Apriori算法研究与实现**

**一、实验目的**

通过编程实现Apriori算法，并在给定数据集上进行测试，完成频繁模式的挖掘。

**二、实验设备和环境**

**Windows操作系统下的vscode和python3.9**

**三、实验内容**

编程实现Apriori算法，并在数据集T10I4D1K.dat、T10I4D10K.dat、T10I4D50K.dat、T10I4D100K.dat上完成测试，测试时各个数据集上的频繁模式，支持度计数阈值如下表所示，请按照数据集从小到大的次序找到各个支持度计数下的频繁项集，并保存在文档中。

|  |  |  |  |
| --- | --- | --- | --- |
| 数据集 | 支持度计数 | | |
| T10I4D1K | 6 | 8 | 10 |
| T10I4D10K | 60 | 80 | 100 |
| T10I4D50K | 300 | 400 | 500 |
| T10I4D100K | 600 | 800 | 1000 |

**四、实验过程、结果及分析**

1. 实验过程，关键代码，实验结果及分析。

首先观察给定的数据集特征，发现其中的值均在0-999范围内，而事务总数高达10w条，

查看书上的各种方法后，选择了使用垂直数据格式挖掘的方法，使得总复杂度有一定程度的降低，同时使用dfs剪枝来加速搜索，当当前的事务集合长度已经小于最小支持度的时候，跳过之，不再对其进行任何计算，每次开始递归时将答案加入特定列表，直至没有更多的分支，搜索完毕为止，输出结果时当遇到空集合则退出，说明不存在该长度及以上的频繁项集

代码如下：

scale=100

cfm=scale\*6

lines=scale\*1000

def dfs(inter,st,ct,items):

    vec1[8-ct].append(items)

    if ct==0:return

    for i in range(st,len(vec)):

        temp=inter.intersection(vec[i])

        if len(temp)<cfm:continue

        tmp=items.copy()

        tmp.add(i)

        dfs(temp,i+1,ct-1,tmp)

xx=open('T1014D%dK.dat'%scale,'r')

vec=[set() for \_ in range(1000)]

for item,i in zip(xx.readlines(),range(lines)):

    item=item.strip().split()

    for num in item:

        vec[int(num)].add(i)

vec1=[[] for \_ in range(8)]

dfs(set(range(lines)),0,8,set())

vec1.pop(0)

for item in vec1:

    l=len(item)

    if l==0:break

    print(l)

    for i in range(min(5,len(item))):

        print(item[i])

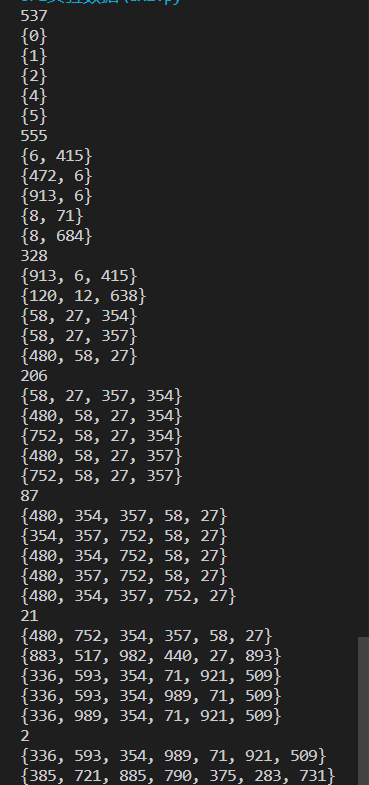
xx.close()

其中，scale变量用于选择数据集，cfm控制支持度，lines与数据集行数相等，是辅助变量，之所以使用dfs的方式进行计算，是因为对于垂直挖掘方式而言，要想找到n元的频繁项集，就需要进行n次循环枚举，而n不确定，因此将其作为变量ct放入dfs中运算，同时正好可以使用剪枝来优化搜索，由于在事先的实验中认为本数据集不会有超过8长度的频繁项集，因此设定从8开始，这个值不会影响算法效率，仅当存在比这个值更长的频繁项集时，这个值会影响正确性

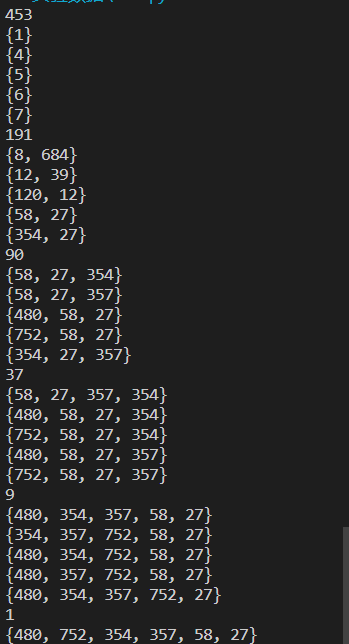
实验结果如下，依次输出的是从1开始的各项数的频繁项集，其中频繁项集过多的情况下将只输出前5个

1K

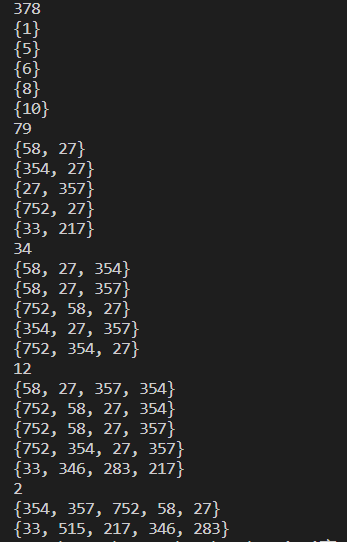
支持度6



支持度8

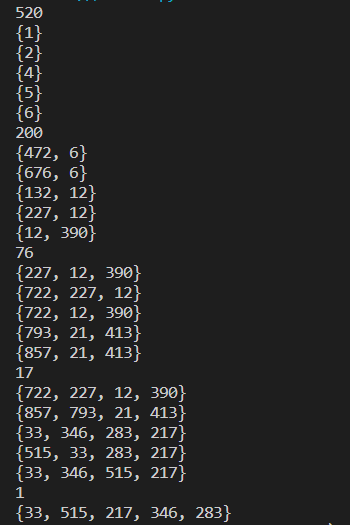


支持度10

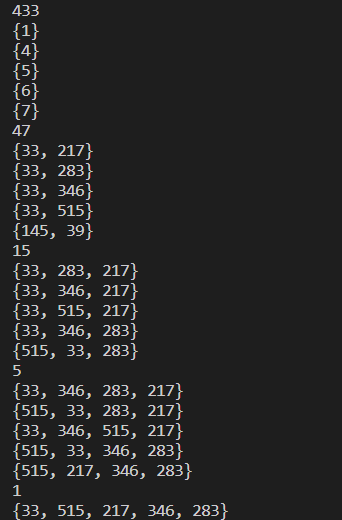


10K

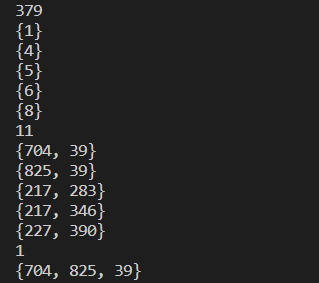
支持度60



支持度80

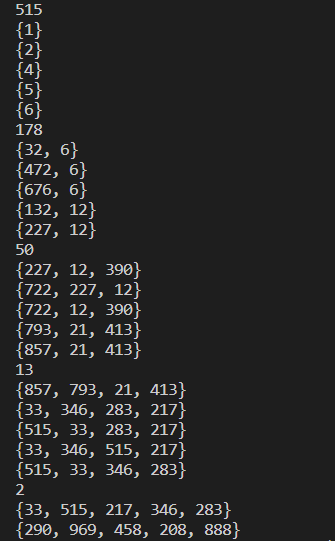


支持度100

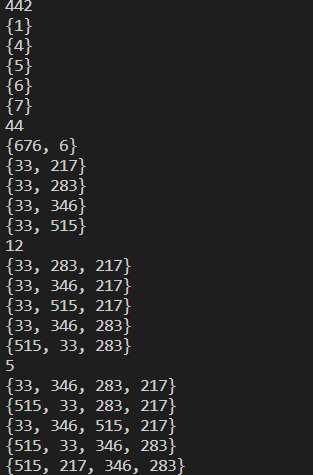


50K

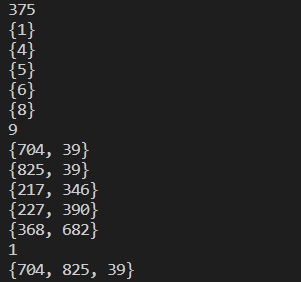
支持度300



支持度400

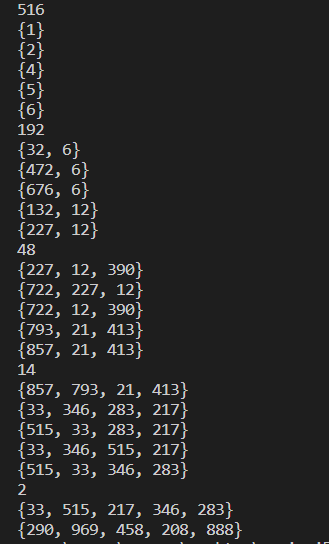


支持度500

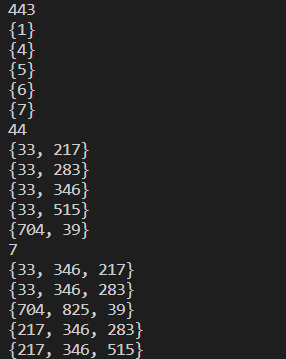


100K

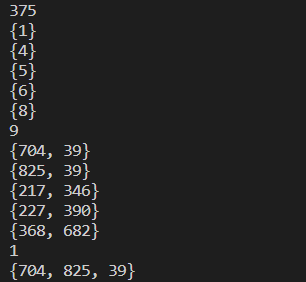
支持度600



支持度800



支持度1000



在优化后，最慢的情况，100K数据集搭配600支持度，也可以在20秒内完成搜索

从数据结果上看，可以看到在支持度更高的情况下得出的结果，是可以在支持度更低的结果中找到的，因此可以认为答案应当是正确的

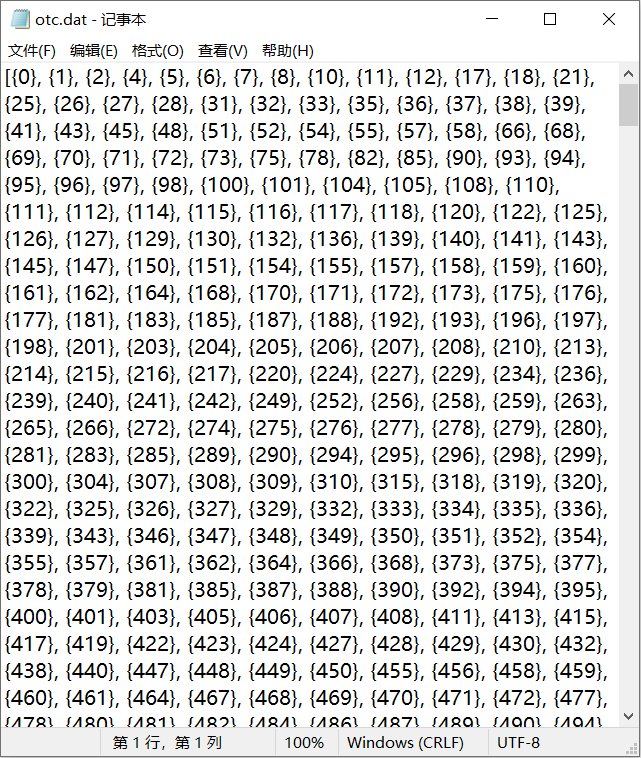
使用如下的代码段将结果写入文件中

yy=open('otc.dat','a')

yy.write(str(vec1))

yy.close()

文件中情况如下图



1. 实验中遇到的问题及解决办法。

刚开始时尝试过普通的做法运算1K数据集，发现速度过慢，预期100K数据集的复杂度

将会不可接受，因此查阅课本进行优化的学习

刚开始使用垂直挖掘法时有一段困惑的时期，因为计算结果不能前后对应，后来发现是由于事务编号范围出错，修改后结果正确

刚开始得出结果时对结果的理解有误，将行号集合错认为是频繁项集，经过思考和反复阅读课本相关内容后，添加了求取真正频繁项集的代码，结果与同学验证正确

刚开始时切换数据集和支持度需要一个个改文件名和行号范围等参数，在一次实验成功将要进入多次实验前，我使用了全局控制参数来控制这两个变量，更加便捷

1. 实验还存在哪些问题。

实验中，在计算100K数据集的情况下，仍然需要10秒以上的时间才能得出结果，课本

上展示了垂直挖掘的一种差分优化方法，我尚未研究和采用

本次实验中，由于数据集中的数据范围有限，才可以使用垂直挖掘的方法优化，假如是在实际应用中，可能范围会过大，如几亿等，这种情况下，垂直挖掘的复杂度仍然是不可接受的，需要寻找其他优化方法

1. 本实验有哪些收获和心得体会。

本次实验中深刻地学习了数据关系集挖掘的过程和目的，通过思考和查阅学习了一些优

化的方法，同时加深了对搜索剪枝的理解

1. 其它需要补充的问题。

暂无