**Python利用FP-Growth算法从新闻站点点击流中挖掘热门新闻报道**

**【实验原理】**

频繁项集挖掘算法用于挖掘经常一起出现的item集合（称为频繁项集），通过挖掘出这些频繁项集，当在一个事务中出现频繁项集的其中一个item，则可以把该频繁项集的其他item作为推荐。比如经典的购物篮分析中啤酒、尿布故事，啤酒和尿布经常在用户的购物篮中一起出现，通过挖掘出啤酒、尿布这个啤酒项集，则当一个用户买了啤酒的时候可以为他推荐尿布，这样用户购买的可能性会比较大，从而达到组合营销的目的。

常见的频繁项集挖掘算法有两类，一类是Apriori算法，另一类是FPGrowth。Apriori通过不断的构造候选集、筛选候选集挖掘出频繁项集，需要多次扫描原始数据，当原始数据较大时，磁盘I/O次数太多，效率比较低下。FPGrowth算法则只需扫描原始数据两遍，通过FP-tree数据结构对原始数据进行压缩，效率较高。

FPGrowth算法主要分为两个步骤：FP-tree构建、递归挖掘FP-tree。FP-tree构建通过两次数据扫描，将原始数据中的事务压缩到一个FP-tree树，该FP-tree类似于前缀树，相同前缀的路径可以共用，从而达到压缩数据的目的。接着通过FP-tree找出每个item的条件模式基、条件FP-tree，递归的挖掘条件FP-tree得到所有的频繁项集。算法的主要计算瓶颈在FP-tree的递归挖掘上。具体的FPGrowth算法的代码可参考http://www.cnblogs.com/datahunter/p/3903413.html网页。

**【实验内容】**

现有某新闻站点用户点击流的一个名为kosarak.dat的文件。该文件中的每一行包含某个用户浏览过的新闻报道，这是一个很大的数据集，有将近100万条记录，且新闻报道已经被编码成整数。我们可以使用FP-growth算法从新闻站点点击流中挖掘其中的频繁项集，查看哪些新闻ID被用户大量观看到。

**【实验环境】**

Linux Ubuntu 16.04

Python3.6

**【实验步骤】**

1.首先在Linux上新建/data/python17目录，并切换到该目录下。

[view plain](https://www.ipieuvre.com/e/186/311/9539) [copy](https://www.ipieuvre.com/e/186/311/9539)

1. mkdir -p /data/python17/
2. cd /data/python17/

2.使用wge命令，从网址 http://192.168.1.100:60000/allfiles/python17/目录下，将实验所需数据下载到linux本地/data/python17目录下。

[view plain](https://www.ipieuvre.com/e/186/311/9539) [copy](https://www.ipieuvre.com/e/186/311/9539)

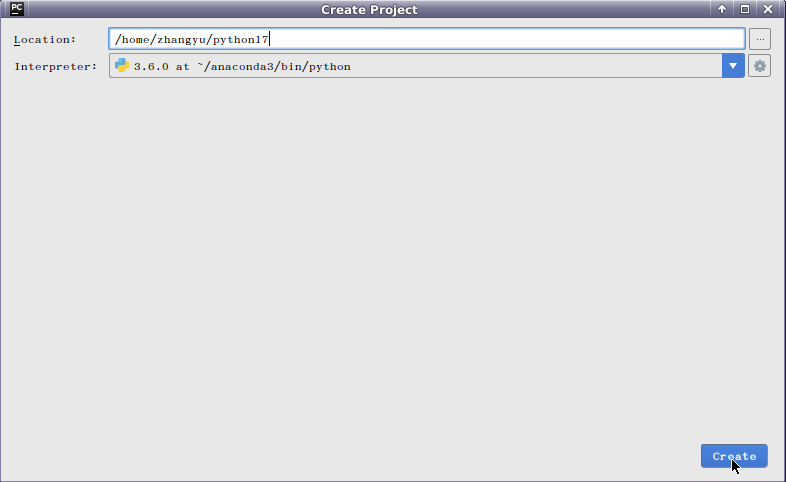
1. wget http://192.168.1.100:60000/allfiles/python17/kosarak.dat.gz

3.将压缩文件kosarak.dat.gz解压到/data/python17/目录下

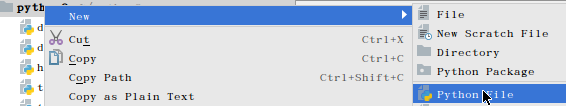
[view plain](https://www.ipieuvre.com/e/186/311/9539) [copy](https://www.ipieuvre.com/e/186/311/9539)

1. gzip -d kosarak.dat.gz

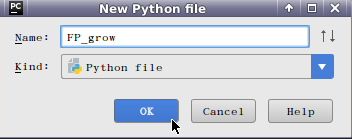
4.新建Python项目，名为python17

[](https://www.ipieuvre.com/doc/exper/28bab895-91ad-11e9-beeb-00215ec892f4/img/bfc1f52e-9a9f-4976-ad13-c6196f04755e.png)

5.在python17项目下，新建Python file

[](https://www.ipieuvre.com/doc/exper/28bab895-91ad-11e9-beeb-00215ec892f4/img/3ac2762d-15a2-4aa3-a36c-3f9a48fbdb78.png)

名为FP\_grow，用于存放FP-Growth算法的代码

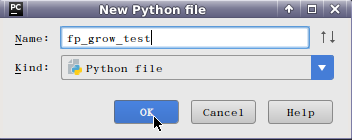
[](https://www.ipieuvre.com/doc/exper/28bab895-91ad-11e9-beeb-00215ec892f4/img/300b2428-859f-481a-8874-88c331d6693b.png)

6.编写FP-Growth算法的代码

[view plain](https://www.ipieuvre.com/e/186/311/9539) [copy](https://www.ipieuvre.com/e/186/311/9539)

1. # -\*-coding:utf-8-\*-
3. # 定义一个树，保存树的每一个结点
4. **class** treeNode:
5. def \_\_init\_\_(self, nameValue, numOccur, parentNode):
6. self.name = nameValue
7. self.count = numOccur
8. self.parent = parentNode
9. self.children = {}  # 用于存放节点的子节点
10. self.nodeLink = None  # 用于连接相似的元素项
11. # 对count变量增加给定值
12. def inc(self, numOccur):
13. self.count += numOccur
14. # 用于将树以文本形式显示，对于构建树来说并不是需要的
15. def disp(self, ind=1):
16. print("  " \* ind, self.name, "  ", self.count)
17. **for** child **in** self.children.values():
18. child.disp(ind + 1)
20. # 生成数据集
21. def loadSimpDat():
22. simpDat=[['r', 'z', 'h', 'j', 'p'],
23. ['z', 'y', 'x', 'w', 'v', 'u', 't', 's'],
24. ['z'],
25. ['r', 'x', 'n', 'o', 's'],
26. ['y', 'r', 'x', 'z', 'q', 't', 'p'],
27. ['y', 'z', 'x', 'e', 'q', 's', 't', 'm']]
28. **return** simpDat
29. def createInitSet(dataSet):
30. retDict={}
31. **for** trans **in** dataSet:
32. retDict[frozenset(trans)]=1
33. **return** retDict
35. # FP树的构建函数
36. def createTree(dataSet,minSup=1):
37. #     ''' 创建FP树 '''
38. #     # 第一次遍历数据集，创建头指针表
39. headerTable={}
40. **for** trans **in** dataSet:
41. #print(trans)
42. **for** item **in** trans:
43. #print(item)
44. headerTable[item]=headerTable.get(item,0)+dataSet[trans]
45. #print(headerTable)
46. label = []
47. #print(headerTable)
48. **for** k **in** headerTable.keys():
49. **if** headerTable[k]<minSup:
50. label.append(k)
51. **for** i **in** label:
52. del(headerTable[i])
53. freqItemSet=set(headerTable.keys())
54. **if** len(freqItemSet)==0:
55. **return** None,None
56. **for** k **in** headerTable:
57. #print(k)
58. headerTable[k]=[headerTable[k],None]
59. #print(headerTable)
60. retTree=treeNode("Null Set",1,None)
61. **for** tranSet,count **in** dataSet.items():
62. localD={}
63. #print(tranSet,count)
64. **for** item **in** tranSet:
65. **if** item **in** freqItemSet:
66. localD[item]=headerTable[item][0]
67. #print(localD.items())
68. **if** len(localD)>0:
69. orderedItems=[v[0] **for** v **in** sorted(localD.items(),key=lambda e: e[1],reverse=True)]
70. #print(orderedItems)
71. updateTree(orderedItems,retTree,headerTable,count)
72. **return** retTree,headerTable
74. def updateTree(items,inTree,headerTable,count):
75. # 判断事务中的第一个元素项是否作为子节点存在，如果存在则更新该元素项的计数
76. #print(items)
77. #inTree.disp()
78. **if** items[0] **in** inTree.children:
79. #print(items[0])
80. inTree.children[items[0]].inc(count)
81. # 如果不存在，则创建一个新的treeeNode并将其作为子节点添加到树中
82. **else**:
83. inTree.children[items[0]]=treeNode(items[0],count,inTree)
84. #print("NO",items[0])
85. **if** headerTable[items[0]][1]==None:
86. headerTable[items[0]][1]=inTree.children[items[0]]
87. **else**:
88. updateHeader(headerTable[items[0]][1],inTree.children[items[0]])
89. **if** len(items)>1:
90. #print(items[1::])
91. updateTree(items[1::],inTree.children[items[0]],headerTable,count)
93. # 获取头指针表中该元素项对应的单链表的尾节点，然后将其指向新节点targetNode
94. def updateHeader(nodeToTest,targetNode):
95. **while**(nodeToTest.nodeLink !=None):
96. nodeToTest=nodeToTest.nodeLink
97. nodeToTest.nodeLink=targetNode
99. # 辅助函数，直接修改prefixPath的值，将当前节点leafNode添加到prefixPath的末尾，然后递归添加其父节点
100. def ascendTree(leafNode,prefixPath):
101. **if** leafNode.parent !=None:
102. prefixPath.append(leafNode.name)
103. ascendTree(leafNode.parent,prefixPath)
104. # 给定元素项生成一个条件模式基（前缀路径）
105. # basePat表示输入的频繁项，treeNode为当前FP树中对应的第一个节点（可在函数外部通过headerTable[basePat][1]获取）
106. def findPrefixPath(basePat,treeNode):
107. condPats={}
108. **while** treeNode != None:
109. prefixPath=[]
110. ascendTree(treeNode,prefixPath)
111. **if** len(prefixPath)>1:
112. condPats[frozenset(prefixPath[1:])]=treeNode.count
113. treeNode=treeNode.nodeLink
114. **return** condPats
115. # 参数：inTree和headerTable是由createTree()函数生成的数据集的FP树
116. #    : minSup表示最小支持度
117. #    ：preFix请传入一个空集合（set([])），将在函数中用于保存当前前缀
118. #    ：freqItemList请传入一个空列表（[]），将用来储存生成的频繁项集
119. def mineTree(inTree,headerTable,minSup,preFix,freqItemList):
120. #print("start")
121. #print(headerTable.items())
122. bigL=[v[0] **for** v **in** sorted(headerTable.items(),key=lambda e: e[1][0])]
123. #print(bigL)
124. **for** basePat **in** bigL:
125. newFreqSet=preFix.copy()
126. newFreqSet.add(basePat)
127. freqItemList.append(newFreqSet)
128. condPattBases=findPrefixPath(basePat,headerTable[basePat][1])
129. #print(condPattBases)
130. myCondTree,myHead=createTree(condPattBases,minSup)
131. **if** myHead !=None:
132. print("conditional tree for: ", newFreqSet)
133. myCondTree.disp(1)
134. mineTree(myCondTree,myHead,minSup,newFreqSet,freqItemList)
135. # 封装算法
136. def fpGrowth(dataSet, minSup=3):
137. initSet = createInitSet(dataSet)
138. myFPtree, myHeaderTab = createTree(initSet, minSup)
139. freqItems = []
140. mineTree(myFPtree, myHeaderTab, minSup, set([]), freqItems)
141. **return** freqItems

7.在python17项目下，新建Python file，名为fp\_grow\_test.py

[](https://www.ipieuvre.com/doc/exper/28bab895-91ad-11e9-beeb-00215ec892f4/img/b009c799-cf39-400f-8497-5139bd8ea37e.png)

8.将数据集导入

[view plain](https://www.ipieuvre.com/e/186/311/9539) [copy](https://www.ipieuvre.com/e/186/311/9539)

1. parsedDat = [line.split() **for** line **in** open('/data/python17/kosarak.dat').readlines()]

9.对数据预处理

[view plain](https://www.ipieuvre.com/e/186/311/9539) [copy](https://www.ipieuvre.com/e/186/311/9539)

1. initSet = FP\_grow.createInitSet(parsedDat)

10.构建FP树，并从中寻找那些至少被10万人浏览过的新闻报道。

[view plain](https://www.ipieuvre.com/e/186/311/9539) [copy](https://www.ipieuvre.com/e/186/311/9539)

1. myFPtree, myHeaderTab = FP\_grow.createTree(initSet, 100000)

11.创建一个空列表来保存这些频繁项集

[view plain](https://www.ipieuvre.com/e/186/311/9539) [copy](https://www.ipieuvre.com/e/186/311/9539)

1. myFreqList = []
2. FP\_grow.mineTree(myFPtree, myHeaderTab, 100000, set([]), myFreqList)

12.看下有多少新闻报道或报道集合曾经被10万或者更多的人浏览过

[view plain](https://www.ipieuvre.com/e/186/311/9539) [copy](https://www.ipieuvre.com/e/186/311/9539)

1. print(len(myFreqList))

13.查看那些新闻ID被用户大量观看到

[view plain](https://www.ipieuvre.com/e/186/311/9539) [copy](https://www.ipieuvre.com/e/186/311/9539)

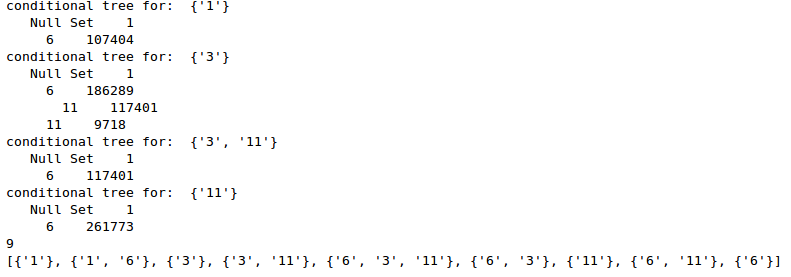
1. print(myFreqList)

14.fp\_grow\_test.py的完整代码

[view plain](https://www.ipieuvre.com/e/186/311/9539) [copy](https://www.ipieuvre.com/e/186/311/9539)

1. **import** FP\_grow
3. #将数据集加载到列表
4. parsedDat = [line.split() **for** line **in** open('/data/python17/kosarak.dat').readlines()]
5. print(parsedDat)
7. #初始集合格式化
8. initSet = FP\_grow.createInitSet(parsedDat)
10. #构建FP树
11. myFPtree, myHeaderTab = FP\_grow.createTree(initSet, 100000)
13. #创建空列表，保存频繁项集
14. myFreqList = []
15. FP\_grow.mineTree(myFPtree, myHeaderTab, 100000, set([]), myFreqList)
16. print(len(myFreqList))
17. print(myFreqList)

15.运行结果为：

[](https://www.ipieuvre.com/doc/exper/28bab895-91ad-11e9-beeb-00215ec892f4/img/bfa8143a-f84c-4b19-8be2-ec5a4e1ac134.png)