序列模式挖掘 (Sequential Pattern Mining)-- 核心算法学习

1. Suffix Tree(ST)----Ukkonen's Algorithm(UA)

$|S| = m$

- Prefix of S : $S[1, i]$
- Suffix of S : $S[i, m]$

Key: 如果 P 是 S 某个后缀的前缀，那么 P 是 S 的 substring

1 预先构造一个 Suffix trie

- Contruct
  - 每个边都有字符标签
  - 同一节点分出来的两个边不同一个标签
  - 每个 Substring 对应一个 叶子 (Leaf)
- Compress
  - 有根书 T 有 m 个叶子
  - 除去 root ；内部节点 有两个以上的 child
  - T 的每个边使用字符串进行 label
  - 同一节点分出来的边 ，以不同的 字符开头
  - 对于每一个 Leaf(i), 其中 i 根据路径上的 label 在 S 的起始位置标定 $S[i,m]$
- Problem
  - 一个后缀是另一个后缀的前缀，那么该后缀的路径不会以叶节点结束
- Solve
  - 假定最后一个字符只出现在末尾，不出现在其他的地方
  - 加入一个新的字符 $ 在 S 的末尾

PS: 叶子数 = 后缀的数目

---

2 构建隐式 ST

- 从所有的边中 移除 $
- 移除所有没有 label 的边
- 移除只有一个 child 的节点

3 UA

```
$I_{i} = S[1,i]$    ##Prefix of S
construct I_{1}

## 从 I_{i} 构建I_{i+1}
```

```
for i = 1 to m-1
    for j = 1 to i + 1
        找出P 从根部开始结束位置的label，逐一使用 S[i+1] 对P进行 extend
convert I to ST
```

- Extend 原则
  - 结束在 叶节点，直接加入
  - 不终止在叶节点，而且后面的字符不是 $S[i+1]$
    - 分割一个新的边
    - 或者可能创建一个 Internal node
  - 已经存在在，无需变动

4 减少 Runtime ---> O(m)

- Suffix link
- Count Trick
- ....
  *

---

1 Suffix link

扩展后，直接跳转到下一个需要扩展的地方，而不是从根部开始搜索查询

- 隐式 ST 每个内部节点都有一个 suffix link (在下一个扩展结束的时候)

2 Count

- Key : 每个节点只需要搜索首字符
- 计数，counter 小，skip

---

5 使用的数据结构

- An Array
- A linked list

- A balanced Tree
- A hashing scheme

6 广义后缀树

Define ： A set of Strings S.

2 Apriori-like 算法

Key： 不频繁的子序列的额超集也不频繁

1 Apriori Alg--- focus on 频繁项集，

- 缺点： 大数据集实现 慢，
- 适用场景： 购物，投票， 网站流量分析
- 适用数据类型： 数值型，或者标称型数据

- 评估标准： **支持度 (Support)-- 频繁** or 置信度 (Confidence)-- 关联
- 迭代的思想
  - 找出所有的包含 1 项的集合，小于支持度的去除
  - 将剩余的 1 项集 进行 连接（按照顺序），构成 2 项集
  - 直到出现空，或者剩 1
- 支持度计算
  - Hash 树进行支持度计算
- 项集连接（合并）
  - $F_{k-1} * F_{k-1}$
  - 按照字典序
- 遍历方式
  - 深度优先 --- a-> ab-> abc -> 非频繁节点： 便于确定 **极大频繁项集**
  - 广度优先

- 扩展
  - 处理分类属性
  - 处理连续属性
    - 基于离散化
    - 基于统计学
    - 非离散化的 --- 动态支持度 (min-Apriori)

应用： 发现毒蘑菇的相似特征

- 挖掘包含某特定元素的项集
  - 将蘑菇的每一个特征对应一个标称数据值，标称值转化为一个集合
  - 第一个特征值对应有毒没毒
  - 挖掘频繁项集，找出包含 对应有毒特征值 的项集

类比于 刘婉甜学姐 介绍的 “交通场景和犯罪发生的对应”
具有相同，相似场景特征的容易催发某类犯罪

- 挖掘包含犯罪特征值的频繁项集
  - 问题 1 ： 犯罪行为太多种，对应多个值，而且要表示犯罪数量
  - 问题 2 ： 场景除了有无，还有程度的表示
  - 问题 3 ： 倾斜支持度分布 --- 支持度阈值选取
  - solve 1 : 犯罪类型合并归类
  - solve 2 ：使用一个复合项 () 来表示某一个特征 $[1, 2, 3, 9, [10, 1]]$
    - [10,1]-- 1 对应场景包含特征程度，或者犯罪数量分为 少，中，多的数值对应
    - 使用 Apriori Alg 的时候， 如果有复合项，先只关注第一项，挖掘
    - 频繁项集挖掘第二项

---

2 GSP

- 加入时间约束： 满足 [mingap,maxgap] 之间的都符合连续
- 加入 time-windows-size, 在 size 内的 认为是同一个 itemset

- w 使用 [l,u] 来刻画： l 是 w 内事件发生的最早事件 ，u-- 最晚

---

3 FP-Tree (Frequent Pattern)

- 优点； 只需要扫描两次数据集，快
- 适用数据类型： 标称型数据

- 建立项目头表
  - 扫描数据，得到所有 1 项频繁的计数，除去小于 support 的，降序排列
  - 再次扫描，原始数据中除去 小于的，，字符降序排列
    PS： 降序排列有利于更多的使用祖先节点
- 建立 FP-tree
  - 按照顺序插入 树，靠前的是祖先节点，后是子节点
  - 插入序列
    - 共同祖先 ： 祖先计数 + 1
    - 新节点出现
- 添加指针 (link)
  - 建立项目头表 (字典) 和节点的 link
  - 建立新节点和已经存在的节点的 link
- 挖掘
  - 从项目头表的底部向上挖掘
  - 找每一项的 **条件模式基 (conditional pattern base)**
  - 条件模式基 -- 以所查找的元素项为结尾的路径集合

3 PrefixSpan 前缀投影的模式挖掘

- 输出：所有满足支持度要求的频繁序列集
- 从长度为 1 的前缀开始挖掘序列模式，搜索对应的**投影数据库**得到长度为 1 的前缀对应的

频繁序列，

- 投影数据库： 相同前缀对应的所有后缀的集合

- 然后递归的挖掘长度为 2 的前缀所对应的频繁序列，

- PrefixSpan 算法由于不用产生候选序列，且投影数据库缩小的很快，内存消耗较小
- PrefixSpan 可以优化构造投影数据库

刘闯
2018/10/30