第一次作业

凌春阳　1801213692

一、问题描述

频繁推导子树挖掘(Frequent induced subtree mining)。参考论文 Lei Zou, Yansheng Lu, Huaming Zhang, Rong Hu: PrefixTreeESpan: A Pattern Growth Algorithm for Mining Embedded Subtrees. WISE 2006: 499-505.

二、算法设计与实现

算法的主要流程如论文中所述。先找到所有一阶频繁的 label，对于每个 label，构造其投影数据库，接下来递归调用过程挖掘高阶频繁子树。

Algorithm PrefixTreeESpan
Input: A tree database D, minimum support threshold *min_sup*
Output: All frequent subtree patterns
Methods:
1) Scan D and find all frequent label *b*.
2) **For each** frequent label *b*
3) Output pattern tree <b –1>;
4) Find all **Occurrences** of b in Database D, and construct <b –1>-projected database through collecting all corresponding *Project- Instances* in *D*;
5) **call** *Fre*(<b –1>, 1 , ProDB(D,<b -1>) , *min_sup*).

Function Fre(S, *n* , ProDB(D,S), *min_sup*)
Parameters: *S*: a subtree pattern ; *n*: the length of S; *ProDB(D,S)* : the <S>-projected database; *min_sup*: the minimum support threshold.
Methods:
1) Scan *ProDB(D,S)* once to find all frequent ***GEs*** *b*.
2) **For each** GE *b*
3) extent *S* by *b* to form a subtree pattern S′, and output S′.
4) Find all **Occurrences** of *b* in ProDB(D,S), and construct < S′ > -projected database through collecting all corresponding *Project- Instances* in *ProDB(D,S)* ;
5) **call** *Fre* (S′, *n+1* , *ProDB(D,* S′),*min_sup*).

Fig. 5. Algorithm PrefixTreeESpan

在递归函数 Fre 中，首先找到所有频繁的增长因子（GE）。对于每个增长因子，扩充频繁子树，再产生其对应的投影数据库，就得到了更高一阶的频繁模式。

Induced subtree 和 embedded subtree 的不同之处就在于增长因子的选择，induced subtree 只有前缀树上的直接子节点可以作为增长因子，而

embedded subtree 可以将所有子孙结点都作为增长因子，因此搜索空间会大一些，产生的频繁模式会更多一些。

在代码实现中，主要用到的一些类包括 Record，ProjectInstance，ProjectDB。

```
class Record:
    """ 原始数据库中的一条记录
        包含原始的一行string表示，including -1 tag. 每个结点用其索引作为id
        并计算每个结点的partner，用来表示该结点范围
        每个结点的children，方便进行induced subtree扩展
    """
```

```
class ProjectDB:
    """ 投影数据库类
        pattern_tree: 对应的频繁模式
        instances: 一条ProjectInstance
        GEs: 所有 legal growth elements 及其计数
    """
```

```
class ProjectInstance:
    """ 投影数据库中的一个实例
        tid: 来自于原始数据库中的哪条记录
        pattern_tree_nodes: 记住产生该模式的所有结点，便于寻找GE
    """
```

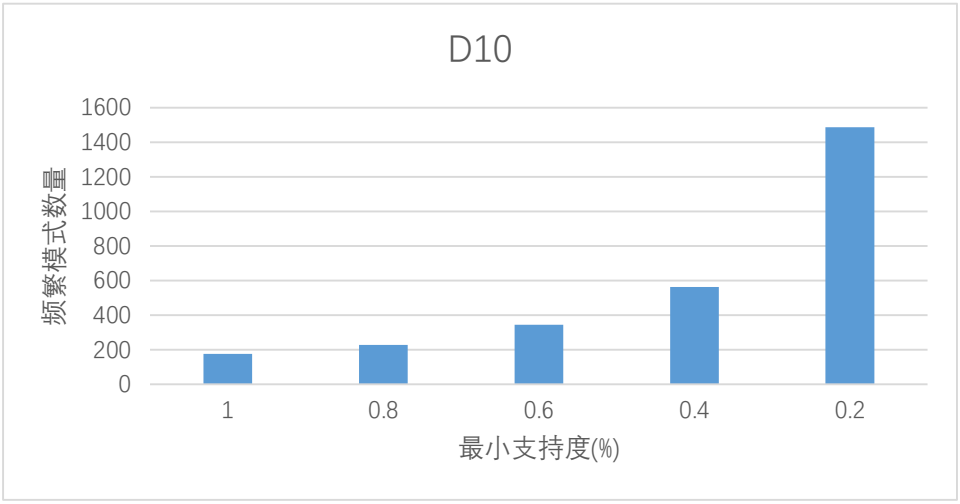
三、实验结果

首先构造了论文中简单的测试数据，验证程序的正确执行。

接下来在四个数据上进行了实验。对于 D10, T1M, F5 三个数据集，设置的五组支持度为 1%，0.8%，0.6%，0.4%， 0.2%；对于 CSlog 数据集，由于其频繁模式的支持度都比较高，参考论文中的指标，设置的五组支持度为 3.0%， 2.5%， 2.0%， 1.5%， 1.0%。

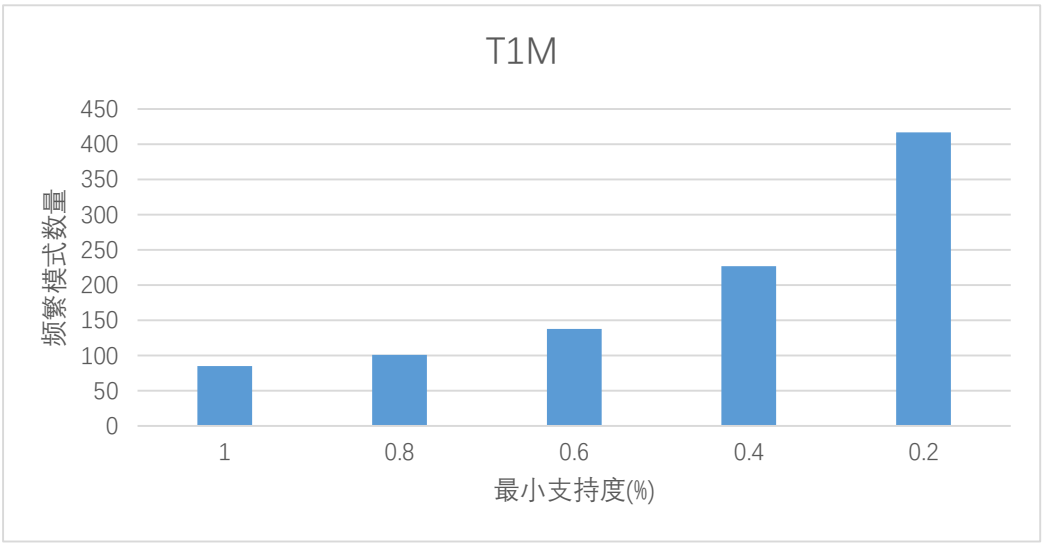
1) D10

Min_sup(%)	1	0.8	0.6	0.4	0.2
Patterns	176	229	344	563	1487
Time (s)	9.7	11.2	14.5	17.2	21.6



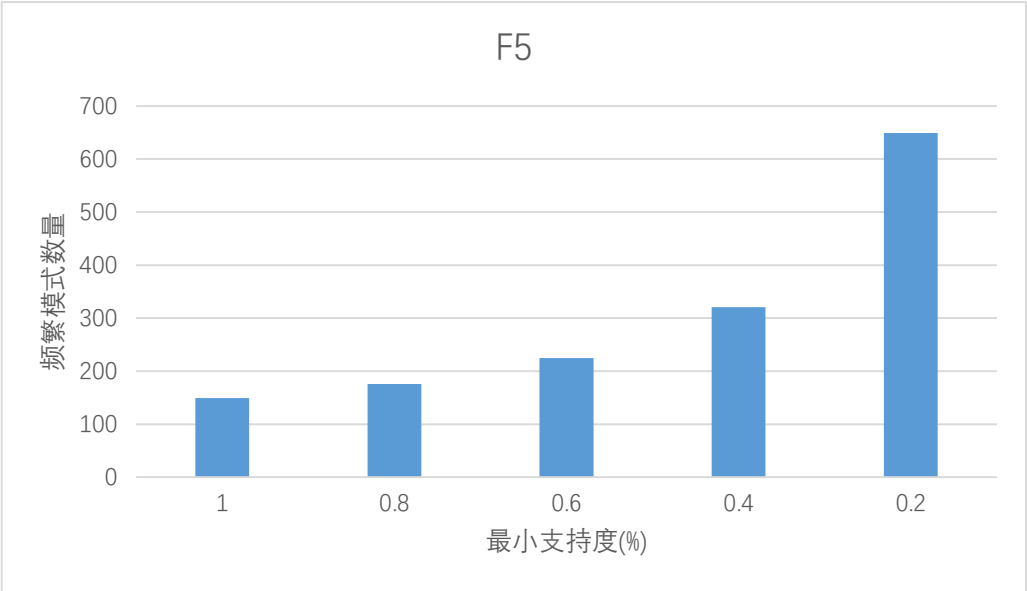
2) T1M

Min_sup(%)	1	0.8	0.6	0.4	0.2
Patterns	85	101	138	227	417
Time (s)	68.1	77.0	80.2	103.3	119.5



3) F5

Min_sup(%)	1	0.8	0.6	0.4	0.2
Patterns	149	176	225	321	649
Time (s)	8.6	9.3	10.1	11.2	13.2



4) CSlog

Min_sup(%)	3.0	2.5	2.0	1.5	1.0
Patterns	33	50	74	110	216
Time (s)	8.4	9.0	10.2	13.7	26.2