Data Mining Homework - 2

Discovery of Frequent Itemsets and Association Rules

Group – 7

Xing Zeng <xingzeng@kth.se> Sevket Melih Zenciroglu <smzen@kth.se>

1. Dataset

We use a sale transaction dataset[1], which includes generated transactions (baskets) of hashed items. The size is 4MB which includes 100,000 rows. Each row is one transaction.

```
# 1st step: read data
datContent = [i.strip().split() for i in open("./T10I4D100K.dat").readlines()]
```

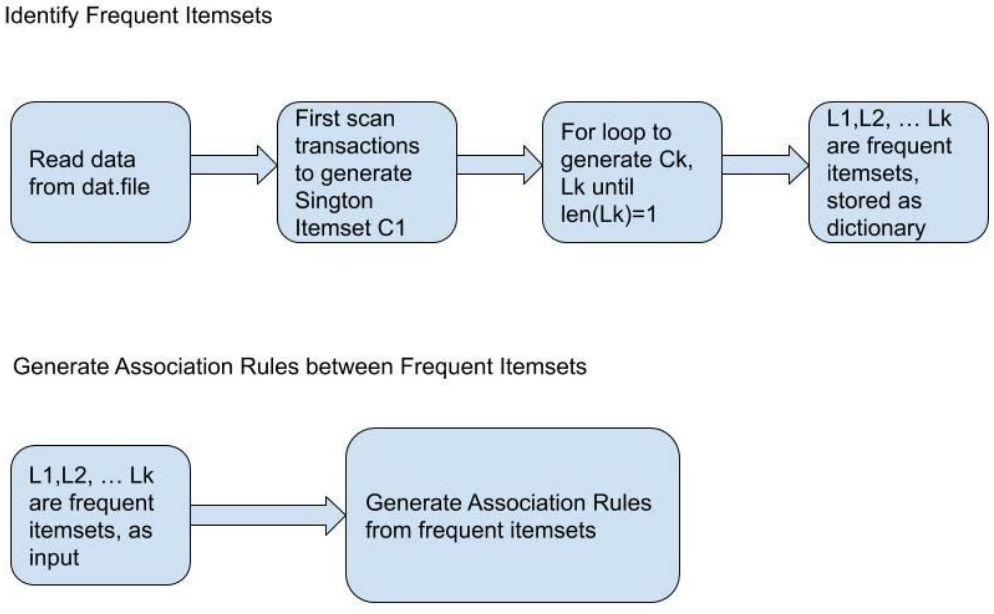
```
datContent[1:4]
```

```
[['39', '120', '124', '205', '401', '581', '704', '814', '825', '834'],
 ['35', '249', '674', '712', '733', '759', '854', '950'],
 ['39', '422', '449', '704', '825', '857', '895', '937', '954', '964']]
```

2. Workflow

The dataset is a transaction dataset, each row is one transaction. Our goal is to identify the frequent itemsets in all transactions and to generate association rules between frequent itemsets. And we use the A-Priori algorithm.

According to the A-Priori algorithm, we need to identify all the frequent itemsets based on the previous frequent itemset we already generated. So our workflow is as follows,



2.1 Generate singleton candidate itemsets C1

First scan all transactions to store each element as one set(singleton) and calculate the corresponding support, the result is stored in a dictionary as singleton candidate itemsets C1

```
len(C1)
```

```
870
```

```
C1
```

```
{'25': 1395,
 '52': 1983,
 '164': 744,
 '240': 1399,
 '274': 2628,
 '328': 663,
 '368': 7828,
 '448': 1370,
 '538': 3982,
 '561': 2783,
 '630': 1523,
 '687': 1762,
 '730': 602,
 '775': 3771,
 '825': 3085,
 '834': 1373,
 '39': 4258,
 '120': 4973,
 '124': 294,
```

.

2.2 For Loop to Generate Lk and Ck (frequent k-itemsets and candidate k-itemsets)

Define support as 1% of the number of transactions.

In the generate_Lk() function, we use Ck,support as parameters. According to candidate k-itemsets and support threshold, filter each itemset which support >= support threshold. Then return frequent k-itemsets Lk.

In the generate_Ck()function, use Transactions, L(k-1)as parameters. Return candidate k-itemset based on previous frequent k-1 itemsets L(k-1). In this function, scan each transaction, then scan each_itemset of L(k-1), if all elements in each_itemset is in this transaction, then scan again L(k-1), to check if each non-repetitive element in another itemset is also in this transaction, if yes, make a tuple, which is one generated itemset of candidate k-itemsets Ck, at the same time, calculate and update its support.

For loop these two functions generate_Lk(Ck,support and generate_Ck(datContent,Lk) until only one frequent k-frequent itemset is generated.

L1, L2… Lk are all frequent itemsets.

```
frequent_itemset #Lk
 ('85',): 1555,
 ('450',): 2082,
 ('428',): 1021,
 ('550',): 1203,
 ('769',): 1622,
 ('554',): 1114,
 ('366',): 1031,
 ('820',): 1473,
 ('207',): 1214},
2: {('39', '825'): 1187,
 ('704', '825'): 1102,
 ('39', '704'): 1107,
 ('227', '390'): 1049,
 ('789', '829'): 1194,
 ('368', '829'): 1194,
 ('217', '346'): 1336,
 ('368', '682'): 1193,
 ('390', '722'): 1042},
3: {('39', '704', '825'): 1035}}
```

2.3 Generate Association Rules

Set the confidence threshold as 0.5. We store all the frequent subsets in one list: all_frequent_itemset = []. For loop all_frequent_itemset twice to combine two disjoint frequent itemset into one new subset, if new subset is also in all_frequent_itemset and confidence (itemset1 ---> new_subset) > 0.5, then it is one of the rules.

```
rule = generate_rule(frequent_itemset,confidence)
```

```
rule
```

```
{('704',): ('39', '825'),
 ('227',): ('390',),
 ('39', '825'): ('704',),
 ('704', '825'): ('39',),
 ('39', '704'): ('825',)}
```

3.References:

[1] https://canvas.kth.se/courses/20000/files/3225774