Part 2: Association Analysis

Apriori algorithm:

The Apriori algorithm uses a “bottom-up” approach, where frequent subsets are extended one item at once and groups of candidates are tested against the data. The algorithm terminates when no further successful rules can be derived from the data.

There are three steps in our implementation:

Step 1: Generate the frequent itemsets satisfying the given support value.
Step 2: Generate all the rules from frequent itemsets which satisfy the given confidence.
Step 3: Select the qualified rules according to the given query template.

Step 1: Generating frequent itemsets:

Firstly we iterate the given dataset to get all the frequent itemset of size 1. Then we generate size 2 frequent itemset by doing this:
1, take two non-similar itemset, and check if the intersection of the parent itemset has a length = length of newly itemsets - 2(which is zero now), if satisfied then pair them to form a candidate itemset of size 2.
2, the new itemset is formed by taking the union of the parent itemsets.
3, calculate the support of newly formed itemsets and only select the qualified ones for next iteration.
4, iterate this process until we get an empty candidate itemset.

Step 2: Generating all rules satisfying given support and confidence:

We generate all the rules that satisfy the given support and confidence. The candidate rule is generated by merging two rules sharing the same prefix in the rule body.

To make it more efficient, we use the anti-monotone property of the confidence of rules generated from the same itemset. For example, because c(ABC →D) ≥ c(AB→CD)≥ c(A→BCD), if ABC →D does not satisfy, then we will exclude all the rules with head is the subsets of ABC and the whole rule including ABCD.

Step 3: Selecting qualified rules for the given query:

We generate rules according to the givens query. There are three types of queries. If the query is template 1 or template 2, we can select the rules satisfying template 1 or template 2 accordingly. If the query is template 3, the query consists of two parts each of which is either template 1 query or template 2 query. The result are combined based on the 'AND' or 'OR' operator which combines template 1 and template 2.

Results

1, Numbers of frequent itemsets with different lengths for support of 30%, 40%, 50%, 60% and 70%.

Support is set to be 30%

Number of length-1 frequent itemsets: 196

Number of length-2 frequent itemsets: 5340

Number of length-3 frequent itemsets: 5287

Number of length-4 frequent itemsets: 1518

Number of length-5 frequent itemsets: 438

Number of length-6 frequent itemsets: 88

Number of length-7 frequent itemsets: 11

Number of length-8 frequent itemsets: 1

Number of all lengths frequent itemsets: 12879

Support is set to be 40%

Number of length-1 frequent itemsets: 167

Number of length-2 frequent itemsets: 753

Number of length-3 frequent itemsets: 149

Number of length-4 frequent itemsets: 7

Number of length-5 frequent itemsets: 1

Number of all lengths frequent itemsets: 1077

Support is set to be 50%

Number of length-1 frequent itemsets: 109

Number of length-2 frequent itemsets: 63

Number of length-3 frequent itemsets: 2

Number of all lengths frequent itemsets: 174

Support is set to be 60%

Number of length-1 frequent itemsets: 34

Number of length-2 frequent itemsets: 2

Number of all lengths frequent itemsets: 36

Support is set to be 70%

Number of length-1 frequent itemsets: 7

Number of all lengths frequent itemsets: 7

2, rules counts satisfying different queries

Number of rules for support 30% and confidence 70%: 31759

Number of rules for support 40% and confidence 70%: 1137

Number of rules for support 50% and confidence 70%: 117

Number of rules for support 60% and confidence 70%: 4

Number of rules for support 70% and confidence 70%: 0

Setting support=50% and confidence= 70%.

Template1 Queries:

(result11, cnt) = asso_rule.template1("RULE", "ANY", [‘G59_UP’]): 26

(result12, cnt) = asso_rule.template1("RULE", "NONE", [‘G59_UP’]): 91

(result13, cnt) = asso_rule.template1("RULE", 1, ['G59_UP', ‘G10_Down']): 39

(result14, cnt) = asso_rule.template1("HEAD", "ANY", [‘G59_UP’]): 9

(result15, cnt) = asso_rule.template1("HEAD", "NONE", [‘G59_UP']): 108

(result16, cnt) = asso_rule.template1("HEAD", 1, ['G59_UP', ‘G10_Down’]): 17

(result17, cnt) = asso_rule.template1("BODY", "ANY", [‘G59_UP']): 17

(result18, cnt) = asso_rule.template1("BODY", "NONE", [‘G59_UP']): 100

(result19, cnt) = asso_rule.template1("BODY", 1, ['G59_UP', ‘G10_Down’]): 24

Template2 Queries:

(result21, cnt) = asso_rule.template2("RULE", 3): 9

(result22, cnt) = asso_rule.template2("HEAD", 2): 6

(result23, cnt) = asso_rule.template2("BODY", 1): 117

Template3 Queries:

(result31, cnt) = asso_rule.template3("1or1", "HEAD", "ANY", ['G10_Down'], "BODY", 1, [‘G59_UP’]): 24

(result32, cnt) = asso_rule.template3("1and1", "HEAD", "ANY", ['G10_Down'], "BODY", 1, [‘G59_UP’]): 1

(result33, cnt) = asso_rule.template3("1or2", "HEAD", "ANY", ['G10_Down'], "BODY", 2): 11

(result34, cnt) = asso_rule.template3("1and2", "HEAD", "ANY", ['G10_Down'], "BODY", 2): 0

(result35, cnt) = asso_rule.template3("2or2", "HEAD", 1, "BODY", 2): 117

(result36, cnt) = asso_rule.template3("2and2", "HEAD", 1, "BODY", 2): 3