CSE 601: Data Mining and Bio-Informatics

Association Analysis – Apriori Algorithm

Sai Kumar Aindla – 50321372

Swathi Ravindran – 50314300

Akshara Santharam - 50313950

OBJECTIVE:

Given a dataset, we need to find frequent item-sets for different support values using Apriori algorithm. We generate association rules from frequent item-sets satisfying given confidence and support requirements. Based on the template queries, we filter the generated rules.

APRIORI ALGORITHM:

The Apriori algorithm is an efficient way to extract combinations of item sets which has support requirements. In Apriori algorithm, if an item-set is frequent then all of its subsets will be also frequent. All of its supersets will also be infrequent, if an item set is infrequent which is based on anti-monotone.

IMPLEMENTATION:

The implementation is done in 3 phases:

1. **Part 1:** The frequent item-sets which qualify the given support requirements are generated
2. **Part 2:** The frequent item-sets which qualify with the given confidence requirements is generated
3. **Part 3:** Based on the given query we filter the generated rules by selecting only the rules which qualify.

PART 1: GENERATION OF FREQUENT ITEM-SETS

The item-sets of different lengths are generated from the given dataset. The support for each item-sets is compared and is stored in a dictionary and is compared with the support requirements. The item-sets which qualify the support requirements are stored in a list.

Finally all the item-sets along with their support are found and that is used to print the item-sets of different lengths which qualify the support requirement.

PART 2: ASSOCIATION RULES GENERATION

In the first phase, we generated all the frequent item-sets satisfying the threshold value. For each item-set of the frequent item-sets, we run association rule generation algorithm:

- Using binary partitioning on aN itemset, candidate rules are generated.
- Pruning of candidate rules at each level of binary partitioning of itemset is performed whose confidence falls below minimum threshold.
- If the confidence of candidate rule is less than the minimum threshold, all the rules which the subsets are can be pruned. This is according to the anti-monotone property.

When the confidence is high enough, the rule is true to justify the decision based on it.

PART 3: FILTERING RULES WHICH QUALIFY THE GIVEN QUERY

The template of the query is checked and is classified. The first template and the second template are used to fetch queries based on the given templates 1 and 2. For template 3 the query is first broken into two parts and used as input for the template 1 and 2 methods and is combined using OR or AND operators.

RESULTS:

Support = 30%, Confidence = 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
Total frequent itemsets: 12879

Support = 40%, Confidence = 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
Total frequent itemsets: 1077
2528 rules generated.

Support = 50%, Confidence = 50%

Number of length-1 frequent itemsets: 109
Number of length-2 frequent itemsets: 63
Number of length-3 frequent itemsets: 2
Total frequent itemsets: 174
117 rules generated.

Template 1:

1) Query : asso_rule.template1("RULE", "ANY", ['G59_Up'])

Result: 26 rows selected.

2) Query: asso_rule.template1("RULE", "NONE", ['G59_Up'])

Result: 91 rows selected.

3) Query: asso_rule.template1("RULE", 1, ['G59_Up', 'G10_Down'])

Result: 39 rows selected.

4) Query: asso_rule.template1("BODY", "ANY", ['G59_Up'])

Result: 9 rows selected.

5) Query: asso_rule.template1("BODY", "NONE", ['G59_Up'])

Result: 108 rows selected

6) Query: asso_rule.template1("BODY", 1, ['G59_Up', 'G10_Down'])

Result: 17 rows selected

7) Query: asso_rule.template1("HEAD", "ANY", ['G59_Up'])

Result: 17 rows selected

8) Query: asso_rule.template1("HEAD", "NONE", ['G59_Up'])

Result: 100 rows selected

9) Query: asso_rule.template1("HEAD", 1, ['G59_Up', 'G10_Down'])

Result: 24 rows selected

Template 2:

1) Query: asso_rule.template2("RULE", 3)

Result: 9 rows selected

2) Query: asso_rule.template2("BODY", 2)

Result: 6 rows selected

3) Query: asso_rule.template2("HEAD", 1)

Result: 117 rows selected

Template 3:

1) Query: asso_rule.template3("1or1", "BODY", "ANY", ['G10_Down'], "HEAD", 1, ['G59_Up'])

Result: 24 rows selected

2) Query: asso_rule.template3("1and1", "BODY", "ANY", ['G10_Down'], "HEAD", 1, ['G59_Up'])

Result: 1 rows selected
3) Query: asso_rule.template3("1or2", "BODY", "ANY", ['G10_Down'], "HEAD", 2)

Result: 11 rows selected
4) Query: asso_rule.template3("1and2", "BODY", "ANY", ['G10_Down'], "HEAD", 2)

Result: 0 rows selected
5) Query: asso_rule.template3("2or2", "BODY", 1, "HEAD", 2)

Result: 117 rows selected
6) Query: asso_rule.template3("2and2", "BODY", 1, "HEAD", 2)

Result: 3 rows selected