State University of New York at Buffalo
CSE- 601 Data Mining and Bioinformatics

Project1 Part 2: Association Analysis

Karthikeyan Ramachandriya Subramanian 50289080
Riya Hazra 50290600
Shwetasree Chowdhury 50296995

Problem Statement:

The dataset is about gene expressions (association-rule-test-data.txt) and each row stands for a patient/sample. The last column is the disease name. Implement the Apriori algorithm to find all frequent itemsets.The number of frequent itemsets for support of 30%, 40%, 50%, 60%, and 70%, respectively.

Apriori Algorithm:

Apriori is an algorithm for frequent item set mining and association rule learning. It proceeds by identifying the frequent individual items in the database and extending them to larger and larger item sets as long as those item sets appear more than a given number of times(min support count).

Apriori Implementation

Apriori algorithm is conducted in the following wide steps:
Step 1: Generate the frequent item-sets which qualify the given support threshold.
Step 2: Generate rules from the frequent item-sets which qualify the given confidence threshold.
Step 3: Filter the generated rules based on the given query by selecting only the rules which qualify the query.

Generating frequent item-sets:
Frequent item-sets are generated by first segregating individual occurrences of each item and cumulating the total number of their occurrences. The min support defines the items that can be allowed to be counted as “frequently” appearing. In every iteration, we eliminate items that do not cross the min support threshold. In the subsequent iterations, individual items are grouped with other frequently occurring

individual items, and their collective support is calculated and so on and so forth, until we reach a set of items which appear the most frequent number of times.

Generating association rules:
Association rules take into account not just support, but also confidence which is ratio of (collective appearance of X&Y/ individual appearance of X in the association) in the association X -> Y. Rules are calculated based on their confidence from the set of frequent itemsets.

Generating rules based on the template
Below are generated some rules and their count based on the following template

Results obtained:

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 length 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 length 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 length frequent itemsets: 174

Support is set to be 60%
Number of length-1 frequent itemset: 34

Number of length-2 frequent itemsets: 2
Number of all length frequent itemsets: 36

Support is set to be 70%
Number of length-1 frequent itemsets: 7
Number of all length frequent itemsets: 7

For Association Analysis of the above frequent itemsets, we set **Support=50%, Confidence=70%**. The following queries are:

For ***template 1***, we have 9 possible keywords combinations:

(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**

For ***template 2***, we have 3 keywords choices:
(result21,9) = (result21, cnt) = asso_rule.template2("RULE", 3) : **9**
(result22, cnt) = asso_rule.template2("HEAD", 2) : **6**
(result23, cnt) = asso_rule.template2("BODY", 1) : **117**

For ***template 3***, AND/OR logical operator are used to connect 2 parts i.e. either template 1 or template 2.
(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): 2