DATA MINING & BIOINFORMATICS

ASSOCIATION ANALYSIS

Himal Dwarakanath	himaldwa	50207594
MuthuPalaniappan Karuppayya	muthupal	50208484
Neeharika Nelaturu	nnelatur	50207062

Implementation of Apriori Algorithm:

The implementation is done in 3 phases:

- **Phase 1:** Generate the frequent item-sets which qualify the given support threshold.
- **Phase 2:** Generate rules from the frequent item-sets which qualify the given confidence threshold.
- **Phase 3:** Filter the generated rules based on the given query by selecting only the rules which qualify the query.

Phase 1: Generating frequent item-sets:

The item-sets of different lengths are generated from the given dataset. To start with we begin with item-sets of length 1 which consists of all the unique genes/diseases in the dataset. We check the support for each of them and store it in a dictionary (*master_gene_dict*) compare it with the support threshold. The item-sets which qualify the support threshold are stored in a list (*itemset_candidate_list*). The elements of *itemset_candidate_list* are used to form item-sets of length 2 which happens as follows:

- Two non similar item-sets from *itemset_candidate_list* are taken and paired to form a prospective 2 length item-set.
- To check if item-set going to be formed will be valid, we check if the intersection of the parent item-sets has a length = length_of_newly_formed_itemset – 2
- If valid, then the new item-set is formed by taking a union of the parent item-sets.
- The support of newly formed item-sets are calculated and the ones which qualify the support threshold are taken into the refreshed *itemset_candidate_list* for the next iteration and the elements act as parents to form the item-sets of higher length
- This process is repeated until we obtain an empty *itemset_candidate_list*.

Finally all the item-sets along with their support are found in the *master_gene_dict*. This is used to print the item-sets of different lengths which qualify the support threshold.

Phase 2: Generating frequent item-sets:

The support qualifying item-sets obtained from phase 1 are used to generate rules in phase 2. The item-set is divided into two parts – BODY and HEAD where BODY is a combination of items in the item-set whose length is in the range 1 to (length of the item-set – 1) and HEAD is obtained by taking XOR of the BODY and the item-set. The confidence for the rule is calculated and compared with the confidence threshold. The rules which qualify the confidence threshold are stored in a set (*rules_set*).

Phase 3: Fetching rules which qualify the given query:

The template of the query is checked and classified into 1, 2 or 3. The methods *get_template1_rules* and *get_template2_rules* are used to fetch the rules based on the given query for template_1 and template_2 respectively. In case of a template_3 query, the query is first broken down into two parts each of which is either a template_1 or template_2 query. These broken queries are used as input for the *get_template1_rules* and/or *get_template2_rules* methods and the results are combined based on the AND or OR operator which combines the two templates.

Results for Requirement-1:

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

Results for Requirement-2:

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

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("BODY", "ANY", ['G59_Up']): 9
(result15, cnt) = asso_rule.template1("BODY", "NONE", ['G59_Up']): 108
(result16, cnt) = asso_rule.template1("BODY", 1, ['G59_Up', 'G10_Down']): 17
(result17, cnt) = asso_rule.template1("HEAD", "ANY", ['G59_Up']): 17
(result18, cnt) = asso_rule.template1("HEAD", "NONE", ['G59_Up']): 100
(result19, cnt) = asso_rule.template1("HEAD", 1, ['G59_Up', 'G10_Down']): 24

Template2 Queries:

(result21, cnt) = asso_rule.template2("RULE", 3): 9
(result22, cnt) = asso_rule.template2("BODY", 2): 6
(result23, cnt) = asso_rule.template2("HEAD", 1): 117

Template3 Queries:

(result31, cnt) = asso_rule.template3("1or1", "BODY", "ANY", ['G10_Down'], "HEAD", 1, ['G59_Up']): 24
(result32, cnt) = asso_rule.template3("1and1", "BODY", "ANY", ['G10_Down'], "HEAD", 1, ['G59_Up']): 1
(result33, cnt) = asso_rule.template3("1or2", "BODY", "ANY", ['G10_Down'], "HEAD", 2): 11
(result34, cnt) = asso_rule.template3("1and2", "BODY", "ANY", ['G10_Down'], "HEAD", 2): 0
(result35, cnt) = asso_rule.template3("2or2", "BODY", 1, "HEAD", 2): 117
(result36, cnt) = asso_rule.template3("2and2", "BODY", 1, "HEAD", 2): 3