CSE 601 - Project 1: Dimensionality Reduction & Association Analysis

Team members:

Saketh Varma Pericherla - **sakethva**

Arvind Thirumurugan - **athirumu**

Vijay Jagannathan - **vijayjag**

Part 2: Association Analysis:

Generating frequent itemsets using Apriori:

We use the Apriori Algorithm to generate frequent itemsets. The Apriori algorithm is anti-monotonic i.e. it is based on the property that any subset of a frequent itemset must be also frequent.

In other words, this means any superset of an infrequent itemset must also be infrequent . So, we prune all the subsets of infrequent itemsets which reduces the number of candidate itemsets.

Procedure:

- To generate length-1 frequent itemsets, we simply count the number of occurences of each candidate and prune the candidates which do not meet the minimum support threshold. See `getPrunedData()`
- To generate length-2 frequent itemsets, we consider all the 2-item combinations of the length-1 frequent itemsets and prune the candidates which do not meet the minimum support threshold. See `generateTwoItemSubset()`
- To generate frequent itemsets with length > 2 we group the length-2 itemsets by their prefixes and prune the candidates which do not meet the minimum support threshold. See `generateSubsets()`
- We continue this procedure until there are no more length-k frequent itemsets.

Results:

- **Support = 30%:**

```
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
Length of all length frequent Itemsets: 12879
```

- **Support = 40%:**

```
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
Length of all length frequent Itemsets: 1077
```

- **Support = 50%:**

```
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
Length of all length frequent Itemsets: 174
```

- **Support = 60%:**

```
Support is set to be 60%
number of length-1 frequent itemsets: 34
number of length-2 frequent itemsets: 2
Length of frequent Itemsets: 36
```

- **Support = 70%:**

```
Support is set to be 70%
number of length-1 frequent itemsets: 7
number of length-2 frequent itemsets: 0
Length of frequent Itemsets: 7
```

Generating association rules based on the templates:

Once we get the frequent itemsets, we generate rules by dividing the itemset into HEAD (LHS) and BODY (RHS) as follows:

- To generate rules for level one in the lattice we simply generate all combinations of rules having only one item in the RHS. See `generateLevelOneRules()`

- If the confidence of the current level is below the threshold, prune all the subsequent rules that share the same prefix. See `checkConfidence()`
- For all levels greater than one, candidate rule is generated by merging two rules in the above level that share the same prefix. See `generateOtherLevels()`
- We classify the query specified in the command-line into either template 1, 2 or 3 and use `generateRulesForOne()`, `generateRulesForTwo()`, `generateRulesForThree()` to generate rules for 1, 2 and 3 respectively.
- For template 3 we process the split the query into two and generate rules using a combination of templates 1 and 2 and perform a set operation between them based on the operation specified in the query.

Results:

Template 1:

```
(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
```

Template 2:

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

Template 3:

```
(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
```