CSE 601

**Dimensionality Reduction &**

**Association Analysis**

Submitted by:

Sindhura

Anurag

Rimi Das (50206511)

**Table of Contents**

1. **APRIORI ALGORITHM:**

The Apriori algorithm is a very influential algorithm for mining frequent itemsets for boolean association rules. It uses a bottom-up approach where the frequent subsets are extended one at a time known as candidate generation and at every iteration the candidates so generated are tested against the data. Apriori essentially does the nipping in the bud technique i.e. pruning the infrequent itemsets at every step such that only the supersets of frequent itemsets are generated. This method is based on the **Apriori principle** which says :

**If an itemset is frequent, then all of its subsets must also be frequent.**



The Apriori Principle holds based on the following support measure rules:

* Support of an itemset never exceeds the support of its subsets
* This is known as the anti-monotone property of support

**Support Count** : The number of times a particular elements or set occurs in the given transactional database.

**Anti-Monotone Property :**

* Any subset of a *frequent* itemset must be also *frequent*
* In other words, any superset of an infrequent itemset must also be infrequent
* No superset of any infrequent itemset should be generated or tested.

**Our Implementation** :

We have implemented the Apriori Principle in generating frequent itemsets from the given list of transactions using Java. Initially , when the file the read, a HashMap is created to store the individual set elements and their ccoreesponding support count. On every iteration, the following set of tasks in the below mentioned order are executed :

* The frequent itemsets sets of length k are filtered out – depending if their support count is greater than or equal to the minimum support value(from 0.0 to 1.0) as entered by the user.
* With new list of frequent itemsets, the itemsets are merged with one another to generate sets of length k+1. In this step, an effective optimization was implemented using Apriori principle. Before merging two sets, the elements in both of the sets upto a position of k-1(starting from 0) are matched for equality. Only if all the elemnets till that position match, then they are allowed to merge and solely the newly found unique sets are added to the list of frequent items. This step is done to ensure that only those supersets are created whose all of the subsets are frequent as well.
* Now again, the support count is found by matching it with the transaction database(in our case a HashSet of Sets). And the above mentioned steps are continued until either the new merged sets are of same length as that of number of columns in a transactional row or no new unique sets are added to the list of frequent itemsets.

**Our Results** :

* For support = 0.3

Support is set to be 30.0%

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

* For support = 0.4

Support is set to be 40.0%

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

* For support = 0.5

Support is set to be 50.0%

Number of length-1 frequent itemsets:109

Number of length-2 frequent itemsets:63

Number of length-3 frequent itemsets:2

* For support = 0.6

Support is set to be 60.0%

Number of length-1 frequent itemsets:34

Number of length-2 frequent itemsets:2

* For support = 0.7

Support is set to be 30.0%

Number of length-1 frequent itemsets:7

1. **ASSOCIATION RULE GENERATION ALGORITHM:**

Given a frequent itemset, generating all of its all non-empty subsets f ⊂ L such that f → L – f satisfies the minimum confidence requirement is called Rule Generation. Essentially, if an itemset L contains k number of elements in it, i.e. |L| = k, then there are 2k – 2 candidate association rules (ignoring L → ∅ and ∅ → L).

**Confidence** : It is the ratio of the support of the superset of elements in a rule to the support count of the body of a rule. Confidence does not have an anti-monotone property.

c(ABC →D) can be larger or smaller than c(AB →D)

But confidence of rules generated from the same itemset has an anti-monotone property

For example : if L = {A,B,C,D} , then c(ABC → D) ≥ c(AB → CD) ≥ c(A → BCD)

Therefore, Confidence is anti-monotone w.r.t. number of items on the RHS of the rule

**Our Implementation**:

We have implemented the Association Rule Generation Algorithm on the frequent itemsets that we got as the output of the Apriori implementation. Then the following tasks are executed in the below mentioned order:

* Each of the set in the list of frequent itemsets is processed to form its subsets. (in other words, its power set is generated)
* Now for every subset of a frequent itemset, the confidence is checked taking the subset as the body and the difference of the set & its subset as the head of a rule.
* If the confidence calculated of a rule is greater than or equal to the minimum confidence value (from 0.0 to 1.0) as entered by the user, then that rule is added to a list of plausible rules.
* After all possible rules are generated with the given confidence threshold, the string query as entered by user (in one of three template formats as mentioned in README.txt), that query is parsed and a list of all the plausible rules that qualify as per the entered query is displayed. Finally, the total number of rules that are screened as valid as per the query is also displayed.

**Our Results** :

* For support = 0.5 & Confidence = 0.7

Support is set to be 50.0%

Number of length-1 frequent itemsets:109

Number of length-2 frequent itemsets:63

Number of length-3 frequent itemsets:2

Total Number of rules initially generated = 117

**Query Results** :

* "RULE ANY ['G59\_Up']” = 26
* "RULE NONE ['G59\_Up']” = 91
* "RULE 1 ['G59\_Up', 'G10\_Down']” = 39
* "BODY ANY ['G59\_Up']” = 9
* "BODY NONE ['G59\_Up']” = 108
* "BODY 1 ['G59\_Up', 'G10\_Down']” = 17
* "HEAD ANY ['G59\_Up']” = 17
* "HEAD NONE ['G59\_Up']” = 100
* "HEAD", 1, ['G59\_Up', 'G10\_Down']” = 24
* "SIZEOF RULE 3” = 9
* "SIZEOF BODY 2” = 6
* "SIZEOF HEAD 1” = 117
* "1or1 BODY ANY ['G10\_Down'] HEAD 1 ['G59\_Up']” = 24
* "1and1 BODY ANY ['G10\_Down'] HEAD 1 ['G59\_Up']” = 1
* "1or2 BODY ANY ['G10\_Down'] HEAD 2” = 11
* "1and2 BODY ANY ['G10\_Down'] HEAD 2” = 0
* "2or2 BODY 1 HEAD 2” = 117
* "2and2 BODY 1 HEAD 2” = 3

**REFERENCES**

[1] https://piazza.com/class\_profile/get\_resource/j27qtla8x536j0/j7dxq1l8hk12pa

[2] https://www.slideshare.net/INSOFE/apriori-algorithm-36054672