**Programming Assignment 1**

2014005014

Jimin Lee

**1. Environment**

* OS: Windows
* Language: Python 3.7

**2. Goal** : Find association rules using the **Apriori** algorithm

**3. Implementation of Apriori algorithm**

* **Step 1**. **Get frequent itemset from TDB.**

1. Initially, scan TDB once to get frequent 1-itemset.
2. k = 1
3. Generate candidate itemsets of length (k+1) from frequent itemsets of length k.
4. Test the candidate against DB and generate frequent itemsets of length (k+1).
5. Terminate when no frequent set can be generated.
6. If not terminate, return to 4.

* **Step 2**. **Generate association rules from the frequent itemset.**

1. For itemset {A, B} in frequent 2-itemset, generate association rules of {A}→{B} and {B}→{A}.
2. For itemsets in frequent k-itemset with k≥3, generate association rules of {j}→{k-j}.
3. {j} means the itemset of length j, and j is from 1 to k-1.
4. Terminate when no association rule can be generated.

**4. Description of code**

* **Apriori class**
* Member
* tran\_idata

- store learned transaction data using 'dict' data type

- key is each item of transaction data

- value is list of transaction ID that contains the item

- ex) { item : [tid, tid….] }

* tran\_num

- the number of transactions

* items

- set of item

- use ‘set’ data type to exclude duplicate

* minSup

- minimum support

* Method
* train(tran\_data, minSup)

- receive transaction data and minimum support value as an argument

- set each member variable

* get\_association\_rules()

- get frequent itemsets using \_\_mine\_frequent\_itemsets() function

- generate association rules from the frequent itemsets

- return list that is sets of association rule

- the format is [x, y, support, confidence], if x → y

* \_\_mine\_frequent\_itemsets()

- search frequent item by comparing each support value and minSup

- generate candidate and test

- return list that is sets of frequent patterns

* \_\_support(x)

- search the set of intersection of the transaction lists includes in each item

- length of the set the means count of x in transaction data

- calculate support value and return

* \_\_confidence(x, y)

- calculate confidence value using support value

- confidence(x, y) =

* \_\_generate\_candidates(f, k)

- receive frequent pattern as f and the length of candidate as k+1

- firstly, self-join f

- the itemset is pruned if the itemset is not included in subset of frequent k-itemset

- return the candidate

* **read\_file(path)**

**-** receive path of input file as an argument

**-** open the file and read transaction data

**-** return transaction data

* **write\_file(path, rules)**

- receive path and association rules as an argument

- write the file association rules and each support value and confidence

* **main()**

**-** receive minimum support, input path and output path as an argument

**-** call read\_file() and store transaction data

**-** create object of Apriori class and call train() method

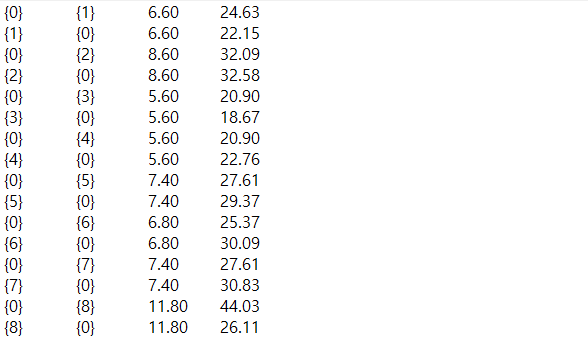
**-** get association rules by get\_association\_rules() method and write

**5. Execution**

**[command line]**

****

**[a part of output.txt]**

****