Assignment B10

import sys

from itertools import chain, combinations

from collections import defaultdict

def subsets(arr):

""" Returns non empty subsets of arr"""

return chain(\*[combinations(arr, i + 1) for i, a in enumerate(arr)])

def returnItemsWithMinSupport(itemSet, transactionList, minSupport, freqSet):

"""calculates the support for items in the itemSet and returns a subset

of the itemSet each of whose elements satisfies the minimum support"""

\_itemSet = set()

localSet = defaultdict(int)

for item in itemSet:

for transaction in transactionList:

if item.issubset(transaction):

freqSet[item] += 1

localSet[item] += 1

for item, count in localSet.items():

support = float(count)/len(transactionList)

if support >= minSupport:

\_itemSet.add(item)

return \_itemSet

def joinSet(itemSet, length):

"""Join a set with itself and returns the n-element itemsets"""

return set([i.union(j) for i in itemSet for j in itemSet if len(i.union(j)) == length])

def getItemSetTransactionList(data\_iterator):

transactionList = list()

itemSet = set()

for record in data\_iterator:

transaction = frozenset(record)

transactionList.append(transaction)

for item in transaction:

itemSet.add(frozenset([item])) # Generate 1-itemSets

return itemSet, transactionList

def runApriori(data\_iter, minSupport, minConfidence):

"""

run the apriori algorithm. data\_iter is a record iterator

Return both:

- items (tuple, support)

- rules ((pretuple, posttuple), confidence)

"""

itemSet, transactionList = getItemSetTransactionList(data\_iter)

freqSet = defaultdict(int)

largeSet = dict()

# Global dictionary which stores (key=n-itemSets,value=support)

# which satisfy minSupport

assocRules = dict()

# Dictionary which stores Association Rules

oneCSet = returnItemsWithMinSupport(itemSet,

transactionList,

minSupport,

freqSet)

print "L1"

currentLSet = oneCSet

print currentLSet

k = 2

while(currentLSet != set([])):

largeSet[k-1] = currentLSet

currentLSet = joinSet(currentLSet, k)

print "\n"

print "L",k

print currentLSet

currentCSet = returnItemsWithMinSupport(currentLSet,

transactionList,

minSupport,

freqSet)

currentLSet = currentCSet

k = k + 1

def getSupport(item):

"""local function which Returns the support of an item"""

return float(freqSet[item])/len(transactionList)

toRetItems = []

for key, value in largeSet.items():

toRetItems.extend([(tuple(item), getSupport(item))

for item in value])

return toRetItems

def printResults(items):

max\_length=0

"""prints the generated itemsets sorted by support and the confidence rules sorted by confidence"""

print "\n"

for item, support in sorted(items, key=lambda (item, support): support):

if len(item)>max\_length:

max\_length=len(item)

print "The frequent itemset extracted with Apriori Algorithm with min support count as 3 (relative support as 0.6) is:\n"

for item, support in sorted(items, key=lambda (item, support): support):

if len(item)==max\_length:

print "item: %s " % str(item)

def dataFromFile(fname):

"""Function which reads from the file and yields a generator"""

file\_iter = open(fname, 'rU')

for line in file\_iter:

line = line.strip().rstrip(',') # Remove trailing comma

record = frozenset(line.split(','))

yield record

if \_\_name\_\_ == "\_\_main\_\_":

inFile = None

inFile = dataFromFile("dataset.csv")

minSupport = 0.6 #3/5 Relative Support

minConfidence =1

items= runApriori(inFile, minSupport, minConfidence)

printResults(items)

OUTPUT

abhishek@Lenovo:~/workspace$ python Apriori.py

L1

set([frozenset(['Onion']), frozenset(['Chocolate']), frozenset(['Keychain']), frozenset(['Eggs']), frozenset(['Mango'])])

L 2

set([frozenset(['Mango', 'Keychain']), frozenset(['Eggs', 'Onion']), frozenset(['Onion', 'Chocolate']), frozenset(['Mango', 'Chocolate']), frozenset(['Keychain', 'Chocolate']), frozenset(['Eggs', 'Chocolate']), frozenset(['Keychain', 'Onion']), frozenset(['Mango', 'Onion']), frozenset(['Eggs', 'Keychain']), frozenset(['Eggs', 'Mango'])])

L 3

set([frozenset(['Eggs', 'Keychain', 'Onion']), frozenset(['Eggs', 'Keychain', 'Chocolate']), frozenset(['Mango', 'Keychain', 'Chocolate']), frozenset(['Mango', 'Keychain', 'Onion']), frozenset(['Eggs', 'Mango', 'Keychain']), frozenset(['Keychain', 'Onion', 'Chocolate'])])

L 4

set([])

The frequent itemset extracted with Apriori Algorithm with min support count as 3 (relative support as 0.6) is:

item: ('Eggs', 'Keychain', 'Onion')

abhishek@Lenovo:~/workspace$