Data Science

Assignment1 : Apriori Algorithm

2016025532 컴퓨터전공 심수정

1. Summary of Algorithm

Apriori 알고리즘은 기본적으로는 frequent 한 pattern 을 생성하는 알고리즘입니다. 이 알고리즘은 downward closure 를 이용합니다. 우선 downward closure 은 frequent itemset 의 subset 은 반드시 frequent 해야 한다는 것을 의미합니다. 이를 활용하여 apriori 알고리즘은 frequent 하지 않은 itemset 의 supserset 을 생성, test 하지 않는 방식으로 진행됩니다. 이를 통해서 많은 candidate 를 줄입니다.

2. Detailed Description of Code

```
# get minimum support, input file name, output file name
import sys
import itertools
minimun_support = float(sys.argv[1])
input_filename = sys.argv[2]
output_filename = sys.argv[3]

# read input.txt file and parsing it to data.
with open(input_filename) as input_file_:
    input_data = input_file.readlines()

input_file.close()
```

실행할 때 받은 argument 를 각각 min_support, input_filename, output_filename 에 넣어주고, input file 을 열어 데이터를 읽어들인 후 input file 을 닫아줍니다.

```
# for rounding at second point exactly
def RoundAtSecondPoint(num) :
    number_string = str(num)
    floating_point = number_string.split(".")
    if len(floating_point[1]) > 2 :
        # in python, 5 is sometimes not ceiled.
        if int(floating_point[1][2]) == 5 :
            num += 0.001
        return round(num, 2)
    else :
        return num
```

python 에서는 5 를 올리지 않는 경우가 있기에 셋째자리가 5 인 경우 셋째자리를 6 으로 만들어 준 후 반올림해줍니다.

```
# get support of item_set
def GetSupport(item_set, tranx, total):
    count = 0  # for check how many times in transaction
    item_set = set(item_set)
    for i in tranx:
        if item_set.issubset(set(i)):
            count = count + 1

    return RoundAtSecondPoint((count / total) * 100)
```

item set, transaction, transaction length 를 받아 support 를 구합니다.

```
# get confidence of item_set
def GetConfidence(item_set_x, item_set_y, tranx):
    count = 0
    total_count = 0
    item_set_x = set(item_set_x)
    item_set_y = set(item_set_y)

    # if item_set_x is in transaction, increase total count and check item_set_y is also in transaction
    # if so, increase count ]
    for i in tranx:
        if item_set_x.issubset(i) :
            total_count = total_count + 1
            if item_set_y.issubset(i) :
                count = count + 1

    # check whether devied 0, and return rounded number or 0
    if total_count == 0 :
        print("no item set")
        return 0
    return RoundAtSecondPoint((count / total_count) * 100)
```

item_set_x, item_set_y, transaction list 를 받아 confidence 를 구합니다.

```
# for removing duplicated element of list
def RemoveDuplicated(list) :

    # if find not duplicated element, add to removed list
    removed_list = []
    for i in list :
        if i not in removed_list :
            removed_list.append(i)
    return removed_list
```

중복되지 않은 항목만 새로운 list 에 추가하여 return 시킵니다.

```
# for generating candidates of apriori algorithm
def GenerateAprioriCandidate(previous_candidate, current_length) : # current length : previous item set C(k-1)'s length
    new_candidate = []

    # union two previous candidate with minimum support and remove duplicated
    for i in previous_candidate :
        for j in previous_candidate :
            new = list(set().union(i, j))
            new_candidate.append(new)
    candidate = list(map(set, new_candidate))
    candidate = RemoveDuplicated(candidate)

    # maintain only candidate with k length
    filtered_candidate = []
    for i in candidate :
        if len(i) == (current_length + 1) :
            filtered_candidate.append(i)
    candidate = filtered_candidate

    return candidate
```

이전 candidate 들을 합집합하고, 해당 개수에 해당하는 candidate 들만 list 로 return 시킵니다.

```
# filter candidates with minimum support
def GetAprioriCandidate(previous_candidate, current_length, min_support, tranx, total) :
    candidate = GenerateAprioriCandidate(previous_candidate, current_length)
    filtered_candidate = []
    for i in candidate :
        if GetSupport(i, tranx, total) >= min_support :
            filtered_candidate.append(i)
    candidate = filtered_candidate
    candidate.sort()
    return  candidate
```

생성된 candidate 들의 support 를 확인하고 frequent pattern 이 아닌 candidate 들을 제거합니다.

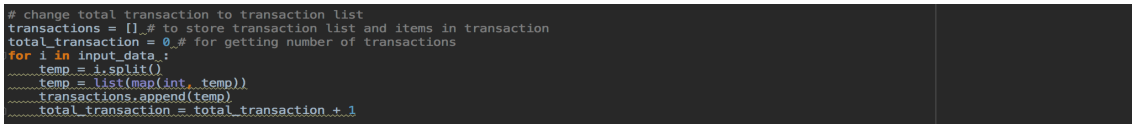
```
# for getting item set with length 1
def GetOneItemSet(tranx, min_support, total) :
    # get variable item in transaction
    candidate = []
    for i in tranx :
        for j in i :
            if not j in candidate :
                candidate.append(j)
    candidate.sort()

    # change element to set
    candidate_list = []
    for i in candidate :
        tmp = []
        tmp.append(i)
        tmp = set(tmp)
        candidate_list.append(tmp)

    # filter candidates with minimum support
    candidate = []
    for i in candidate_list :
        if GetSupport(i, tranx, total) >= min_support :
            candidate.append(i)
    return candidate

# apriori master algorithm
```

1 개짜리 item 을 set 으로 저장하고, 빈발패턴만 return 합니다.

length 별로 frequent 한 item set 을 생성하고, item set 의 subset 을 구해서 support, confidence 를 구해 list 에 넣어주고, 마지막에 생성된 pattern 들, item set, support, confidence 를 담은 list, 큰 length 를 return 해줍니다.

item set, support, confidence 를 담은 list 를 set ₩t set ₩t support ₩t confidence ₩n 의 형태로 output file 로 써줍니다.

list 를 {element1, element2, ...}의 string 형태로 변환하여 return 해줍니다.



input file 의 transaction 을 list 의 list 형태로 바꿔줍니다.

apriori algorithm 을 통해 frequent pattern, support, confidence 를 구하고 이를 output file 에 보내줍니다.

3. Instruction for Compiling Source Code

이 코드는 python3 을 기반으로 작성되었습니다. 따라서 python3 가 설치되

어 있어야 합니다.
“python3 apriori.py [minimum_support] [input_file] [output_file]”의 형태로 complie, 실행하면 됩니다.

4. Other Sepcification of Implementation and Testing

프로그램 종료 시 no more item set 을 출력을 한 후 apriori algorithm 을 종료하게 되니 해당 message 가 뜨기 전에 종료하면 apriori 알고리즘이 제대로 수행되지 않고, output file 에 존재하지 않을 수 있습니다. 또한 “no more item set”이 출력된 후 파일에 저장하는 별도의 시간이 들 수도 있습니다. 그 외에 과제 명세서에 나온 기본적인 format 외의 다른 특별한 특이사항은 존재하지 않습니다.