

# Data Science

## Assignment1 : Apriori Algorithm

2016025532 컴퓨터전공 심수정

### 1. Summary of Algorithm

Apriori 알고리즘은 기본적으로는 frequent한 pattern을 생성하는 알고리즘입니다. 이 알고리즘은 downward closure를 이용합니다. 우선 downward closure은 frequent itemset의 subset은 반드시 frequent해야 한다는 것을 의미합니다. 이를 활용하여 apriori 알고리즘은 frequent하지 않은 itemset의 superset을 생성, test하지 않는 방식으로 진행됩니다. 이를 통해서 많은 candidate를 줄입니다.

### 2. Detailed Description of Code

```
# get minimum support, input file name, output file name
import sys
import itertools
minimum_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 argument를 각각 min\_support, input\_filename, output\_filename에 넣어주고, input file을 열어 데이터를 읽어들이고 후 input file을 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
```

python에서는 5를 올리지 않는 경우가 있기에 셋째자리가 5인 경우 셋째자리를 6으로 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 length를 받아 support를 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
```

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

```
# apriori master algorithm
def GenerateApriori(tranx, min_support, total):
    # set current length 1 and create item set with length 1
    item_set = {}
    current_length = 1
    patterns = []
    item_set[current_length] = GetOneItemSet(tranx, min_support, total)

    # repeat creating candidates until no candidates generated.
    while True:
        # if no candidates generated, terminate function
        if (not current_length in item_set) or (len(item_set[current_length]) == 0):
            print("no more item set")
            break

        # generate candidate with kth length
        item_set[current_length + 1] = GetAprioriCandidate(item_set[current_length], current_length, min_support, tranx, total)

        # generate subsets and get support and confidence
        for i in item_set[current_length]:
            subset = []
            for j in range(current_length):
                subset += list(itertools.combinations(i, j))

            for j in subset:
                difference = set(i) - set(j)
                if (len(difference) < 1) or (len(j) < 1):
                    continue

                tmp1 = [set(i), difference, GetSupport(i, tranx, total), GetConfidence(j, difference, tranx)]
                patterns.append(tmp1)

            current_length += 1

    return patterns, item_set, current_length
```

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.

```
# send fixed candidates to output file with output format
def sendToOutputFile(association_list, output_file):
    output_file.write(item_list_to_str(association_list[0]))
    output_file.write("\t")
    output_file.write(item_list_to_str(association_list[1]))
    output_file.write("\t")
    output_file.write(str(association_list[2]))
    output_file.write("\t")
    output_file.write(str(association_list[3]))
    output_file.write("\n")
```

item set, support, confidence 를 담은 list 를 set Wt set Wt support Wt confidence Wn 의 형태로 output file 로 써줍니다.

```
# change item set list to string with output format in sorted form
def item_list_to_str(item_list):
    count = 0
    string = "("
    item_list = list(item_list)
    item_list.sort()
    for i in item_list:
        if count != 0 and count != len(item_list):
            string += ","
        string += str(i)
        count += 1
    string += ")"
    return string
```

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

```
# change total transaction to transaction list
transactions = [] # to store transaction list and items in transaction
total_transaction = 0 # for getting number of transactions
for i in input_data:
    temp = i.split()
    temp = list(map(int, temp))
    transactions.append(temp)
    total_transaction = total_transaction + 1
```

input file 의 transaction 을 list 의 list 형태로 바꿔줍니다.

```
# apriori algorithm result to file
association_list, item_set, length = GenerateApriori(transactions, minimum_support, total_transaction)
with open(output_filename, "w") as output_file:
    for i in association_list:
        sendToOutputFile(i, output_file)
```

apriori algorithm 을 통해 frequent pattern, support, confidence 를 구하고 이 를 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 compile, 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.