

Data Science (ITE4005)

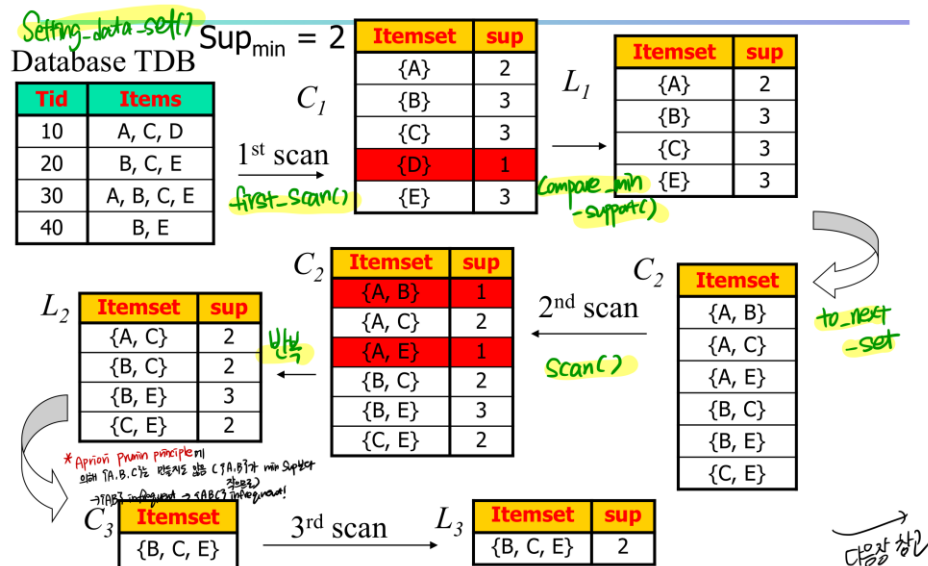
< Programming Assignment #1 >

2016025732 이영석

2021-03-22

1. Summary

The Apriori Algorithm—An Example



March 9, 2021

Data Mining: Concepts and Techniques

10

```
if __name__ == '__main__':
    t1 = time.time()

    handle_input()
    setting_data_set() #TDB
    first_scan() #L1, C1
    length = 2
    while(item_set != {}):
        to_next_set(length) #Ln -> C(n+1)
        scan() #C(n+1) -> L(n+1)
        length+=1

    apply_association_rule(result_set)

    t2 = time.time()
    print("\nTime")
    print(t2-t1) # Time check
```

각각의 function들은 위 그림 형광펜에 대응 (compare_min_support는 scan함수들 내에 존재)

```

import itertools, time, sys

global input_file, output_file

data_set=[] #TBD

check_set=set([]) #Cn
item_set={} # Ln / Hashmap <-> set 기능 번갈아가며 사용
result_set={} # as a dictionary to make association rule, support, confidence

min_support = 0 #default

```

<전역변수> 주석 참고

input.txt내 트랜잭션 데이터들 data_set array(메인 메모리) 저장 후 진행하여 접근 시간 단축

item_set은 performance에 초점

dictionary 기능 : hashmap 형식으로 scan할 때 count

set 기능 : join시 set의 union함수 사용

→ `python Apriori_2016025732.py 2 input.txt output1.txt` Time
1.76088547706604

→ input.txt에서 min_support = 2 로도 1.76초 가량의 performance를 보여줌

2. Detailed Description

A. handle_input(), setting_data_set()

```

def handle_input():
    global min_support, input_file, output_file
    min_support = int(sys.argv[1]) # initialize min_support
    input_file = sys.argv[2]
    output_file = sys.argv[3]

# 'input.txt to DataSet'
def setting_data_set():
    global data_set, min_support
    with open(input_file, 'r') as f:
        for line in f:
            data_set.append(list(map(int, line.replace('\n', '').split('\t'))))
    min_support = min_support * len(data_set) / 100

```

B. first_scan() – 주석참고

```
'''
Put and Count every element using Dictionary (format : frozenset)
Input : TDB (data_set)
Output : L1 (item_set)
'''
def first_scan():
    for i in range(0, len(data_set)):
        for data in data_set[i]:
            temp = frozenset([data])
            if temp in item_set:
                item_set[temp] += 1
            else :
                item_set[temp] = 1
    # Compare every elements if it's bigger than the min_support
    compare_min_support(item_set)
    print('first item set')
    print(item_set)

    result_set.update(item_set)
```

data_set의 element들을 일일이 scan하며 개수 count, item_set에 저장

C. compare_min_support(_dict) – 주석참고

```
'''
Filtering the elements by comparing with min_support
'''
def compare_min_support(_dict):
    temp = []
    for item in _dict:
        if _dict[item] < min_support:
            temp.append(item)
    for i in range(0, len(temp)):
        del(_dict[temp[i]])
```

scan(), first_scan() 내부 모듈. dict의 모든 element들 중 min_support를 넘는 것만 남기고 나머지는 없앴

D. to_next_set(length) – 주석 참고

(length : 구하고자 하는 frequent set 길이)

```
'''
Join the sets and filter bt length and subsets
Input : Ln (item_set)
Output : C(n+1) (check_set)
'''
def to_next_set(length):
    global item_set, check_set
    keys = list(item_set.keys())
    print('keys')
    print(len(keys))

    temp_set = set([])

    num = 0
    for i in range(0, len(keys)-1):
        for j in range(i+1, len(keys)):
            num+=1
            temp = keys[i].union(keys[j])
            # Check if the length is right and its subsets all exist
            if (len(temp) == length and get_subsets(temp, length-1) <= frozenset(keys)):
                temp_set.add(temp)

    check_set = temp_set
```

Ln의 element들끼리 join하여 길이 늘리는 역할. get_subsets로 모든 subsets를 구함.

이후 join후의 길이가 맞고, subsets들이 모두 Ln에 해당하는 elements들만 check_set에 저장

E. get_subsets()

```
'''
Get all the subsets from the set
'''
def get_subsets(_set, length):
    temp = frozenset(frozenset(item) for item in itertools.combinations(_set,length))
    return temp
```

to_next_set과 apply_association_rule의 내부 모듈.

itertools의 combination 메소드를 통해 _set의 length만큼의 모든 subsets을 return

F. scan() – 주석참고

```
'''
Count Cn elements and compare
Input : Cn (check_set)
Output : Ln (new item_set, result_set)
'''

def scan():
    global item_set, check_set
    item_set.clear()

    for data in data_set :
        for check in check_set :
            if check <= frozenset(data):
                if check in item_set:
                    item_set[check] += 1
                else :
                    item_set[check] = 1
        compare_min_support(item_set)

    print("\nnext item set")
    print(item_set)
    result_set.update(item_set)
```

Cn의 element들을 data_set과 비교하고 compare_min_support를 통해 Ln 얻어냄.

이후 result_set에 해당 set들 추가

G. apply_association_rule(_set)

```
'''
Get and print all the association rule using result_set
'''

def apply_association_rule(_set):
    with open(output_file, 'w') as f:

        for items in _set:
            if len(items) != 1:
                for i in range(1, len(items)):
                    temp_set = get_subsets(items, i)
                    for item in temp_set:
                        counterpart = items - item
                        support = '%0.2f' % round(float(_set[items])/float(len(data_set))*100, 2)
                        confidence = '%0.2f' % round(float(_set[items])/float(_set[item])*100, 2)
                        line = str(set(item))+ '\t' + str(set(counterpart)) + '\t' + str(support)+ '\t' + str(confidence) + '\n'
                        f.write(line)
        f.close()
```

result set의 모든 subset와 그 counterpart를 구하고, 각각이 result_set에 몇 개씩 저장되어있는지 확인하여 support와 confidence 확인 후 output file에 저장

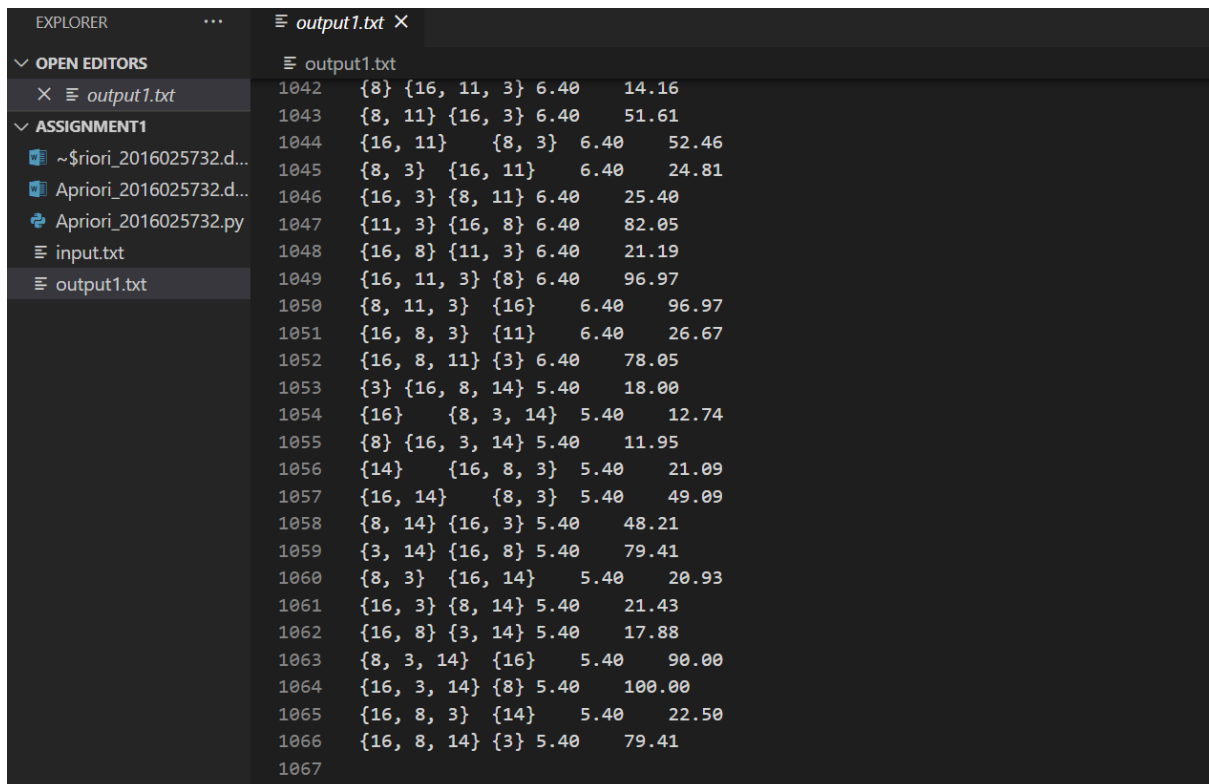
3. Instruction for compiling

- A. python 설치
- B. input file 같은 디렉토리 내 위치
- C. Execute the program with three arguments: minimum support, input file name, output file name

```
PS C:\Users\이영석\Desktop\Assignment1> python Apriori_2016025732.py 5 input.txt output1.txt
```

4. Testing

- A. 3번 과 같이 실행
- B. Result



EXPLORED	...	output1.txt X
OPEN EDITORS		output1.txt
output1.txt		1042 {8} {16, 11, 3} 6.40 14.16
ASSIGNMENT1		1043 {8, 11} {16, 3} 6.40 51.61
~\$riori_2016025732.d...		1044 {16, 11} {8, 3} 6.40 52.46
Apriori_2016025732.d...		1045 {8, 3} {16, 11} 6.40 24.81
Apriori_2016025732.py		1046 {16, 3} {8, 11} 6.40 25.40
input.txt		1047 {11, 3} {16, 8} 6.40 82.05
output1.txt		1048 {16, 8} {11, 3} 6.40 21.19
		1049 {16, 11, 3} {8} 6.40 96.97
		1050 {8, 11, 3} {16} 6.40 96.97
		1051 {16, 8, 3} {11} 6.40 26.67
		1052 {16, 8, 11} {3} 6.40 78.05
		1053 {3} {16, 8, 14} 5.40 18.00
		1054 {16} {8, 3, 14} 5.40 12.74
		1055 {8} {16, 3, 14} 5.40 11.95
		1056 {14} {16, 8, 3} 5.40 21.09
		1057 {16, 14} {8, 3} 5.40 49.09
		1058 {8, 14} {16, 3} 5.40 48.21
		1059 {3, 14} {16, 8} 5.40 79.41
		1060 {8, 3} {16, 14} 5.40 20.93
		1061 {16, 3} {8, 14} 5.40 21.43
		1062 {16, 8} {3, 14} 5.40 17.88
		1063 {8, 3, 14} {16} 5.40 90.00
		1064 {16, 3, 14} {8} 5.40 100.00
		1065 {16, 8, 3} {14} 5.40 22.50
		1066 {16, 8, 14} {3} 5.40 79.41
		1067

minimum support 5로 진행 시 다음과 같이 output1.txt.에서 1066개의 association rule 확인 가능