Assignment #1

Apriori algorithm implementation

컴퓨터소프트웨어학부 2019055078 신채영

Apriori Algorithm

Apriori 알고리즘은 데이터 마이닝 분야에서 사용되는 알고리즘으로 빈번하게 발생하는 항목 집합을 찾아내는 데 사용된다.

- Support(지지도)

- 전체 거래 중 항목 집합이 포함된 거래의 비율
- Minimum Support(최소 지지도)데이터 집합에서 빈발 항목 집합으로 판단하기 위한 최소한의 지지도를 나타내는 임계값

- Confidence(신뢰도)

- 항목 A가 포함된 거래 중에서 항목 B가 포함된 거래의 비율
- Minimum Confidence(최소 신뢰도): 항목 A가 포함된 거래 중에서 항목 B가 포함된 거래의 비율이 최소한의 값 이상이어야 해당 항목 집합이 신뢰도를 갖는 것으로 판단한다.

Environment

- Language: `Python 3.9.13`
- OS: `macOS Vertura 13.2.1`

Conpiling and Running

- 실행 방법

- `python apriori.py min_sup (최소 지지도) input.txt output.txt`의 형태로 실행한다.
- ex. `python apriori.py 5 input.txt output.txt`

- Input

- 과제 명시에 제공된 input 사용

```
7   14
9
18  2   4   5   1
...
```

- Output

- 1066 lines printed

```
{14}    {7} 7.60    29.69
{7} {14}    7.60    31.67
...
{10,3,8}    {16}    7.00    92.11
{16,3,8}    {10}    7.00    29.17
```

Code Description

Main문 및 Apriori 알고리즘

argv에서 input file, output file 그리고 minimum support를 입력받는다.

```
    min_sup = float(sys.argv[1])
    input = sys.argv[2]
    output = sys.argv[3]

    read(input)
```

Frequent items를 담을 배열을 선언하고 k=1 인 Ck, Lk를 구한다.

```
    freq_itemsets = []
    support = defaultdict(int)

    k = 1
    supC1 = c1(datasets)
    Lk, supLk = CktoLk(supC1)
```

k = k인 Ck, Lk를 구한다. k를 증가해가며 실행하다가 더 이상 그룹을 만들 수 없을 때 종료한다.

```
    while True:
        freq_itemsets.append(Lk)
        support.update(supLk)
        Ck, supCk = ck(Lk, supLk, k)
        Lk, supLk = CktoLk(supCk)

        k += 1
        if not Lk or not Ck:
            break
```

구한 모든 frequent items를 파일에 출력한다.

```
    write(output, freq_itemsets)
```

Read input file

`read`: datasets에 데이터들을 입력한 후 minimum support에 개수로 환산해 저장한다.

```
def read(input_file):
    global datasets, min_sup

    f = open(input_file, 'r')
    lines = f.readlines()
    for line in lines:
        items = line.split()
        items = set(items)
        datasets.append(items)

    size = len(datasets)
    min_sup = size * (min_sup/100)
    f.close()
```

Create C1

`c1` : 1개짜리 item과 각 item의 support count를 반환한다.

```
    def C1(db):
        C1_sup = defaultdict(int)
        for items in db:
            for item in items:
                C1_sup[tuple([item])] += 1
        return C1_sup
```

Generate Ck from L(k-1) (self-joining + pruning)

`ck`: L(k-1)에서 Ck를 조합한다. `pruning`함수를 이용해 min_sup을 만족하는지 확인하고 제거한다.

```
def ck(Lk, supLk, k):
    Ck = []
    supCk = defaultdict(int)

    for i in range(len(Lk)):
        for j in range(i+1, len(Lk)):
            a = Lk[i][:-1]
            b = Lk[j][:-1]
            if a == b:
                if Lk[i][-1] < Lk[j][-1]:
```

```
                    candi_itemsets = Lk[i] + tuple([Lk[j][-1]])
                else:
                    candi_itemsets = Lk[j] + tuple([Lk[i][-1]])
                Ck, supCk = pruning(Ck, supCk, supLk, k, candi_itemsets,)
    return Ck, supCk
```

Pruning

`pruning`: candidate itemset이 min_sup을 만족하면 Ck에 추가하고, 아니라면 제거한다.

```
def pruning(Ck, supCk, supLk, k, candi_itemsets):
    for i in range(k+1):
        subset = candi_itemsets[:i] + candi_itemsets[i+1:]
        sup = supLk[subset]

        if sup < min_sup:
            del(candi_itemsets)
            return Ck, supCk

    Ck.append(candi_itemsets)
    supCk[candi_itemsets] = cal_sup(candi_itemsets)

    return Ck, supCk
```

Calculate min_sup

`cal_sup`: 매 단계 min_sup을 계산한다.

```
    cnt = 0
    candidate = set(candidate)
    for i in datasets:
        if candidate.issubset(i):
            cnt += 1
    return cnt
```

Generate Lk from Ck

`CktoLk`: Lk에서부터 frequent itemsets를 만든다. min_sup 조건을 만족하는 itemsets를 Ck에 추가한다.

```
    def CktoLk(supCk):
        Lk = []
        supLk = defaultdict(int)

        for itemset, sup in supCk.items():
            if sup >= min_sup:
                Lk.append(itemset)
                supLk[itemset] = sup
```

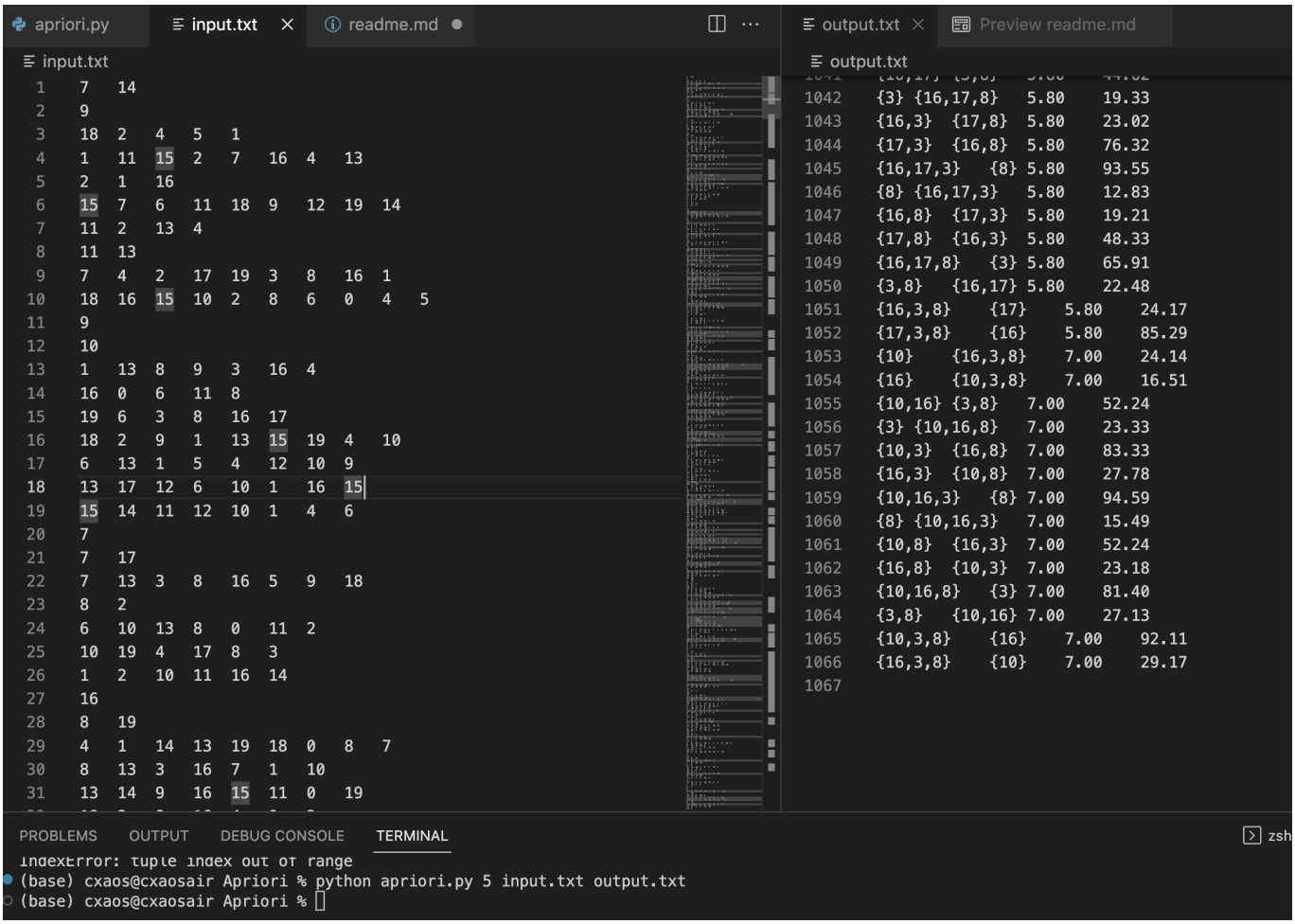
```
        return Lk, supLk
```

Write

`write`: freq_sets에서 itemset들을 추출하고 associative itemset의 support와 confidence를 계산하고 저장한다.

```
    def write(output_file, freq_sets):
        f = open(output_file, 'w')
        for k in range(len(freq_sets)):

            for itemset in freq_sets[k]:
                length = len(itemset)
                subsets = []

                for i in range(1 << length):
                    s = set(itemset[j] for j in range(i) if (i & (1 <<
j)))
                    if s != set() and s != set(itemset):
                        subsets.append(s)

                ##s1, s2의 confidence와 support를 구한다.
                for s1 in subsets:
                    s2 = set(itemset) - s1

                    ## S1 U S2 와 S1을 정렬하여 튜플을 만들고 support를 구한다.
                    sorted_12 = tuple(sorted(s1.union(s2)))
                    sorted_1 = tuple(sorted(s1))
                    sup_12 = support[sorted_12]
                    sup_1 = support[sorted_1]

                    ##소수점 두 자리까지의 support, confidence 값을 계산한다.
                    sup = format(sup_12 / len(datasets)*100, ".2f")
                    conf = format(sup_12 / sup_1*100, ".2f")

                    ## 출력 문자열 만들기
                    string1 = ",".join(str(s) for s in sorted(s1))
                    string2 = ",".join(str(s) for s in sorted(s2))
                    string = "{" + string1 + "}"
                    string += "\t" + "{" + string2 + "}"
                    string += "\t" + str(sup) + "\t" + str(conf) + "\n"

                    f.write(string)
        f.close()
```

Results


apriori.py
input.txt
readme.md
input.txt
7 14
9
18 2 4 5 1
1 11 15 2 7 16 4 13
2 1 16
15 7 6 11 18 9 12 19 14
11 2 13 4
11 13
7 4 2 17 19 3 8 16 1
18 16 15 10 2 8 6 0 4 5
9
10
1 13 8 9 3 16 4
16 0 6 11 8
19 6 3 8 16 17
18 2 9 1 13 15 19 4 10
6 13 1 5 4 12 10 9
13 17 12 6 10 1 16 15
15 14 11 12 10 1 4 6
7
7 17
7 13 3 8 16 5 9 18
8 2
6 10 13 8 0 11 2
10 19 4 17 8 3
1 2 10 11 16 14
16
8 19
4 1 14 13 19 18 0 8 7
8 13 3 16 7 1 10
13 14 9 16 15 11 0 19
output.txt
Preview readme.md
output.txt
1042 {3} {16,17,8} 5.80 19.33
1043 {16,3} {17,8} 5.80 23.02
1044 {17,3} {16,8} 5.80 76.32
1045 {16,17,3} {8} 5.80 93.55
1046 {8} {16,17,3} 5.80 12.83
1047 {16,8} {17,3} 5.80 19.21
1048 {17,8} {16,3} 5.80 48.33
1049 {16,17,8} {3} 5.80 65.91
1050 {3,8} {16,17} 5.80 22.48
1051 {16,3,8} {17} 5.80 24.17
1052 {17,3,8} {16} 5.80 85.29
1053 {10} {16,3,8} 7.00 24.14
1054 {16} {10,3,8} 7.00 16.51
1055 {10,16} {3,8} 7.00 52.24
1056 {3} {10,16,8} 7.00 23.33
1057 {10,3} {16,8} 7.00 83.33
1058 {16,3} {10,8} 7.00 27.78
1059 {10,16,3} {8} 7.00 94.59
1060 {8} {10,16,3} 7.00 15.49
1061 {10,8} {16,3} 7.00 52.24
1062 {16,8} {10,3} 7.00 23.18
1063 {10,16,8} {3} 7.00 81.40
1064 {3,8} {10,16} 7.00 27.13
1065 {10,3,8} {16} 7.00 92.11
1066 {16,3,8} {10} 7.00 29.17
1067
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
zsh
IndexError: tuple index out of range
(base) cxaos@cxaosair Apriori % python apriori.py 5 input.txt output.txt
(base) cxaos@cxaosair Apriori %