[데이터 사이언스 2023-1]

컴퓨터소프트웨어학부

2019016735 현수빈

Assignment 1

1. 컴파일 환경 및 실행 방법

A. 컴파일 환경

- Python3.7.8
- Visual studio code로 개발(v.1.58.2)

B. 실행 환경

- Mac OS 12.6에서 실행하고 개발하였다.

C. 실행방법

- 해당 apriori 파일이 존재하는 곳까지 terminal로 이동한다.
- 아래의 형식에 맞춰 명령어를 입력한다.

python3 apriori.py {minimum_support} {input_file_name} {output_file_name}
Ex) python3 apriori.py 5 input.txt output.txt

```
hyeonsubin-ui-MacBookPro-2:assignment1 hyunsubin$ python3 apriori.py 5 input.txt output.txt
hyeonsubin-ui-MacBookPro-2:assignment1 hyunsubin$
```

2. 구현 방법

A. 전체 구조

```
# input file을 읽고 리스트에 해당 값을 저장. 이 때 dbscan의 수를 줄이기 위해 c1 계산
def read_input_file_and_get_c1(file_name):…


# lk_1 -> ck구할 때 self joining하는 함수
def self_joining(l,k):…

# lk_1 -> ck구할 때 self joining 후 Pruning하는 함수
def pruning_before_testing(c, p):…

# ck와 lk의 차로 pk 구하는 함수: pk는 self joining 후 pruning할 때 pruning하기 위해 저장하는 추가적인 변수
def difference(c,l):…

# ck -> lk, min sup를 넘는 lk를 구하는 함수
def create_lk(ck, support_list, min_sup):
    return [item_set for item_set in ck if support_list[frozenset(item_set)] >= min_sup]

#apriori알고리즘 함수, l1을 구한 후 반복적으로 수행
def apriori(transaction_list, total, min_sup, support_list, c1):…

#association rule을 찾고 output 출력값 생성하는 함수
def find_association_rule(frequent_list, support_list, total):…
```

이번 과제는 apriori.py 단일 소스코드로 작성하였다. 함수를 크게 위의 그림과 같이 총 8개의 함수로 구성되어 있다. (k+1라는 변수를 쓰기 어려워 k=2부터 시작하고 K_1, K로 사용한 점 참고)

I. Read_input_file_and_get_c1

I.1 input file이름을 인자로 받아 해당 파일의 텍스트를 읽어 transaction_list라는 변수에 각 줄마다 텍스트를 set으로 변환하여 저장한다. 이 때 db scan의 수를 줄이기 위하여 읽으면서 item size 1의 candidate인 c1과 해당 c1의 support를 저장한다.

II. Self_joining

II.1 lk-1 리스트를 조합하여 k size의 ck를 만드는 함수이다.

III. Pruning_before_testing

III.1 Lk-1 리스트를 이용하여 ck 리스트를 만들 때 self joining 후 pruning을 하는 함수이다.

IV. Difference

IV.1 pruning을 하기 위해 lk를 만든 후 lk – ck하여 pruning된 set의 집합을 계산하는 함수이다.

V. Create_lk

V.1 ck 리스트 중 min_sup를 넘는 set을 찾아 lk를 생성하는 함수이다.

VI. Apriori

VI.1 input 파일을 처음 스캔하여 얻은 c1과 support list를 이용하여 l1을 계산한 후 lk_1가 없을 때까지 반복문을 돌면서 frequent pattern을 계산하는 함수이다.

VII. Find_association_rule

VII.1 apriori 알고리즘을 이용하여 계산한 support list와 frequent pattern을 이용하여 association rule과 각각의 support, confidence를 계산하여 Output을 만드는 함수이다.

B. 메인 함수 설명

```
if __name__ == "__main__":
    if len(sys.argv) != 4:
        print("argv is not correct! please check argv one more time")
        quit()

    min_sup = int(sys.argv[1])
    min_sup = min_sup/100

    input_file_name = sys.argv[2]
    output_file_name = sys.argv[3]

    #read file and return transaction list and total transaction
    transaction_list, total, support_list, c1 = read_input_file_and_get_c1(input_file_name)

    # find freqeunt pattern
    frequent_pattern_set, support_list = apriori(transaction_list, total, min_sup, support_list, c1)
    result = find_association_rule(frequent_pattern_set, support_list, total)

    output_file = open(output_file_name, 'w')
    output_file.write(result)
    output_file.close()
```

가장 중요한 우선적으로 3개의 인자를 받는 것이 조건이기에 sys.argv 의 수 가 4개가 아니면 경고 문구를 출력하고 Quit하였다.

4개일 경우에 min_sup, input_file_name, output_file_name 변수에 각각의 argv값을 할당하였다. 이 때 min_sup은 퍼센트로 받기에 100으로 나눠 저장하였다. 그 후 **read_input_file_and_get_c1**함수를 호출하여 input_file을 읽어 해당 파일을 스캔하면서 transaction_list와 size 1인 canddiate인 c1과 스캔하면서 support를 계산하였다. 그 후 **apriori** 함수를 이용하여 frequent pattern을 찾고 난 후 **find_association_rule**함수를 이용하여 association rule을 찾으면서 해당 Outptut_file에 저장할 string을 생성하였다. 그 후 output_file에 해당 결과값을 쓰고 종료한다.

C. 주요 함수 설명

i. Apriori

1) 선행조건

우선 해당 함수를 호출하기 전에 Input file에 있는 텍스트를 읽으면서 transaction list에 해당 transaction을 저장했고 database를 스캔하면서 c1과 c1이 나온 Support list를 생성한다.

2) 인자

apriori를 호출 시 인자로는 input_file값을 저장한 transaction_list와 총 거래의 수인 total, min_sup, c1을 만들면서 support를 저장했던 support_list, c1을 받는다.

3) 과정

L은 frequent pattern을 모두 저장하는 리스트이고 p는 pruning한 set의 리스트를 의미한다.

C1과 support list를 이용하여 l1을 생성한다. 그 후 나중에 Pruning을 위해 생성한 lk와 ck의 차를 저장한다.

K=2로 지정한 후 반복문을 돌면서 lk_1가 빈 리스트일 때까지 반복한다. Lk_1가 빈 리스트가 아니라면 self_joining함수를 호출하여 lk_1을 이용하여 size k인 ck를 생성한다. 그 후 pruning_before_testing함수를 호출하여 test하기 전 생성한 ck에서 이미 pruning된 집합을 제거한다. 그 후 transaction_list 리스트를 읽으면서 ck의 support를 count하고 그 값을 support_list에 저장한다. 그 후 create_lk함수를 통해 min_sup

ii. **Find_association_rule**

1) 선행조건

Apriori함수를 호출하였기 때문에 frequent pattern 리스트와 모든 ck에 대한 support_list를 값을 가지고 있는 상태이다.

2) 인자

인자로 frequent_list와 support_list와 total을 받는다.

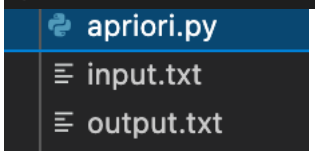
3) 과정

Frequent_list 중에서 set의 길이가 2개 이상인 set만 뽑아 candidate_list를 생성한다. 후 candidate_list의 set에 대하여 가능한 조합으로 item_set과 association_item_set을 만들고 support_list를 이용하여 support와 confidence를 구한다. 그 후 output 형식에 맞게 string을 생성하여 append한다.

D. 실행 결과

주어진 input.txt를 이용하여 min_sup을 5로 실행했을 경우 아래와 같이 output.txt가 생성된다.

```
hyeonsubin-ui-MacBookPro-2:assignment1 hyunsubin$ python3 apriori.py 5 input.txt output.txt
hyeonsubin-ui-MacBookPro-2:assignment1 hyunsubin$
```



해당 output.txt에 들어가면 보이는 결과인데 너무 길어 가장 위의 10개만 사진을 첨부하였다. 결과적으로는 총 1066개의 Association rule이 생성되었다.