데이터 사이언스

Programming Assignment #1 : Apriori algorithm

학생: 최윤석
학번: 2015005169
작성일자: 2022.03.25
교수님: 김상욱 교수님

실행 환경

- Window
- Python 3.10.1

Summary of my program (+algorithm)

- **global 변수**

transaction_list: input file로 받은 transaction을 저장할 변수

min_support_count: 파일 실행 시 입력 받은 min_support persent를 계산을 쉽게 하기 위해 갯수로 변경

- **initialize_item_set**

L1(frequent itemset of size 1)을 만드는 함수.

transaction_list에 저장되어 있는 transaction들에 하나씩 접근하면서 item이 item_set_1이라는 dictionary에 존재하면 값을 늘리고 없으면 새롭게 추가하며 item의 수를 counting 한다.

counting이 끝나면 min_support_count보다 작은 item들을 삭제해 L1을 완성한다.

- **make_candidate_set_list(prev_keys_list, item_length)**

item_length와 만들고자 하는 C{item_length}보다 item_length가 하나 작은 L{item_length-1}의 key값만 모은 prev_keys_list를 사용한다.

길이가 2인 경우에는 combination으로 바로 길이 2짜리 candidate들을 뽑아내면 되지만 2보다 클 경우, 해당 요소들로 만들 수 있는 다양한 조합을 모두 뽑아내야 하기 때문에 모든 prev_key에 하나씩 접근하며 list(single_key_list)에 추가하고

이 list에 combination을 적용시켜 candidates를 얻어낸다.

- **make_frequent_set(prev_keys_list, item_length, candidate_set_list)**

make_candidate_set_list와 동일하게 prev_keys_list와 item_length를 사용하며 make_candidate_set_list에서 얻은 candidate_set_list를 사용한다.

이 부분이 Apriori pruning principle이 적용되는 부분으로 자신보다 하나 길이가 작은 모든 조합에 대하여 그 중 하나라도 frequent하지 않을 경우 frequent_set에서 제거한다.

combination으로 얻어낸 모든 조합에 대하여 prev_keys_list에 존재하는지 확인하는 과정을 거친다.

위 과정이 끝나면 모든 transaction을 순회하며 frequent_set에 대해 counting을 진행하고 이 과정이 끝나면 min_support_count보다 작은 요소는 모두 제거한다.

```
for candidate_set in candidate_set_list:
    count = 0
    for key in list(combinations(candidate_set, item_length - 1)):
        key = set(key)

        if key not in prev_keys_list:
            break
        count = count + 1
    # nCn-1 = n 임을 이용해 조건 체크
    if count == item_length:
        # unhashable type error로 key값을 tuple로 변경
        frequent_set_tmp[tuple(candidate_set)] = 0

for key in frequent_set_tmp.keys():
    for transaction in transaction_list:
        if set(key).issubset(set(transaction)):
            frequent_set_tmp[key] += 1

frequent_set = {key: frequent_set_tmp[key] for key in frequent_set_tmp.keys() if
                frequent_set_tmp[key] >= min_support_count}
return frequent_set
```

- **apriori()**

위에서 설명한 함수들을 활용해 candidate_set이 만들어지지 않거나 frequent_set이 만들어지지 않을 때까지 과정을 반복하며, 얻어낸 frequent_set을 item_set_total_list에 추가한다.

- **from_set_to_form(target_set)**

처음 그냥 set자체를 출력하였을 때 itemset이 {1, 2, 3}이런 식으로 띄어쓰기가 포함되어 출력되어서 띄어쓰기가 없는 상태 {1,2,3}과 같은 형태로 출력하기 위해 사용한 함수이다.

- **make_output_text_by_association(item_set_total_list)**

support값과 confidence를 구하기 위한 함수이다.
전체 list에서 길이가 2이상인 것만 가져와서 모든
item_set association_item_set에 대한 support와 confidence를
요구조건에 맞는 형식으로 output_file_text에 저장한다.

```
for list_index in range(1, len(item_set_total_list)):
    for item_set, item_set_count in item_set_total_list[list_index].items():
        sub_itemset_length = list_index
        while sub_itemset_length > 0:
            sub_itemset = list(combinations(item_set, sub_itemset_length))
            for item in sub_itemset:
                associative_item = set(item_set).difference(item)
                support = (int(item_set_count) / num_of_transaction) * 100

                item_count = 0
                for transaction in transaction_list:
                    if set(item).issubset(transaction):
                        item_count += 1
                confidence = (int(item_set_count) / item_count) * 100

                output_file_text.append(
                    '{}\t{}\t{:.2f}\t{:.2f}\n'.format(from_set_to_form(item), from_set_to_form(associative_item),
                                                      support, confidence))

            sub_itemset_length -= 1
```

- **main()**

파일 실행 시 입력받은 매개변수를 바탕으로 실행된다
apriori.py minimum_support input.txt output.txt 형태
association정보까지 얻어낸 후
output file에 출력한 뒤 종료된다.

 # <u>실행 결과</u>

- 컴파일이 필요하지 않습니다.
동일 폴더에 apriori.py와 input.txt가 존재하기만 하고
바른 실행 방법 (ex window에서 git bash사용했을 때 : python ./apriori.py 5 input.txt output.txt와 같이 실행)으로 실행하면 됩니다.
실행 결과는 LMS에 첨부된 PA1.exe 결과와 min_support 5에 대한 txt파일로 첨부하겠습니다.

```
최윤석@DESKTOP-ANR4SCC MINGW64 /d/programming/Github/2022_ite4005_2015005169/assignment1 (main)
$ python apriori.py 5 input.txt output.txt

최윤석@DESKTOP-ANR4SCC MINGW64 /d/programming/Github/2022_ite4005_2015005169/assignment1 (main)
$ python apriori.py 4 input.txt output.txt

최윤석@DESKTOP-ANR4SCC MINGW64 /d/programming/Github/2022_ite4005_2015005169/assignment1 (main)
$
```

output.txt - Windows 메모장

파일(F) 편집(E) 서식(O) 보기(V) 도움말(H)

{17,3}	{8,16}	5.80	76.32
{17,16}	{8,3}	5.80	44.62
{8,3}	{17,16}	5.80	22.48
{8,16}	{17,3}	5.80	19.21
{3,16}	{8,17}	5.80	23.02
{17}	{3,8,16}	5.80	24.37
{8}	{3,17,16}	5.80	12.83
{3}	{8,17,16}	5.80	19.33
{16}	{3,8,17}	5.80	13.68
{8,3,16}	{6}	5.40	22.50
{8,3,6}	{16}	5.40	96.43
{8,16,6}	{3}	5.40	71.05
{3,16,6}	{8}	5.40	96.43
{8,3}	{16,6}	5.40	20.93
{8,16}	{6,3}	5.40	17.88
{8,6}	{16,3}	5.40	42.86
{3,16}	{8,6}	5.40	21.43
{3,6}	{8,16}	5.40	87.10
{16,6}	{8,3}	5.40	50.00
{8}	{3,16,6}	5.40	11.95
{3}	{8,16,6}	5.40	18.00
{16}	{8,3,6}	5.40	12.74
{6}	{8,3,16}	5.40	23.89

```
최 윤 석 @DESKTOP-ANR4SCC MINGW64 ~/Downloads/Project1/새  폴 더
$ ./PA1.exe output.txt outputRsupport4.txt
1804    < The exact number of correct answers you need to print >
1800    < The number of correct answers out of the rules you printed >
1804    < The number of total rules you printed >
```

```
최 윤 석 @DESKTOP-ANR4SCC MINGW64 ~/Downloads/Project1/새  폴 더
$ ./PA1.exe output.txt outputRsupport5.txt
1066    < The exact number of correct answers you need to print >
1064    < The number of correct answers out of the rules you printed >
1066    < The number of total rules you printed >
```

 # FeedBack

- python의 기본 정도만 할 줄 알았는데 이번 기회에 set, dictionary, list, tuple에 대한 개념적인 이해가 확실하게 잡혀서 좋은 기회였다.

아쉬운 점은 테스트 결과 몇 개 답이 이상한 것 같은데 어느 부분에서 잘못된 것인지 찾지 못했다는 점이다.

시간적 여유가 없어서 다양한 시도를 해보지 못했지만

여유가 있을 때 실습시간에 배운 numpy를 사용하여 한 번 더 만들어보고자 한다.