이름: 강응찬

Apriori 알고리즘 리포트

목차

1. Summary of the algorithm

A. Minimum_support와 input text, output text를 입력받기

B. Input텍스트에 있는 new line(₩n)과 tab(₩t)을 제거

C. 매 transacion을 하나의 리스트로 하고 전체 데이터를 더블 리스트로 만들기

D. 데이터를 한번 스캔하면서 각 item의 frequency를 찾아서 item_set과 함께 딕셔너리로 저장하여 candidate_item_set을 리턴

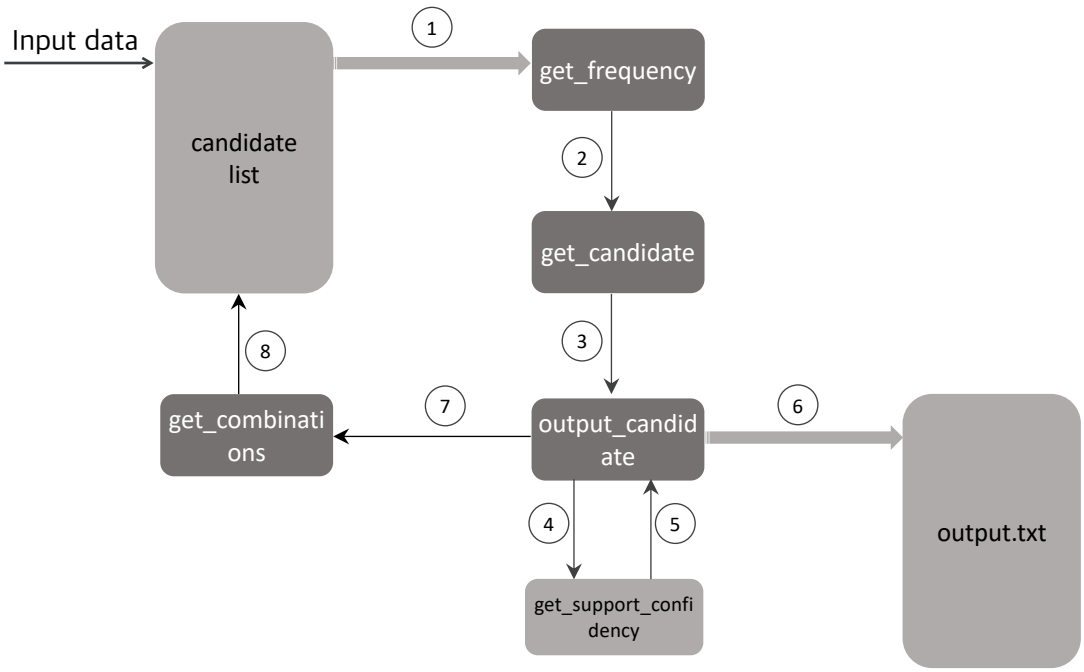
E. 리턴받은 딕셔너리(item_set : frequency)를 가지고 minimum support을 기준으로 그 이상에 해당하는 item만 추출하여 frequent item_set을 리턴

F. 리턴받은 item_set을 가지고 support과 confidence값을 구하여 output파일에 저장

G. frequent_item_set을 k 를 늘려(k++) k 값을 사이즈로 하는 조합을 만들어 각 그룹을 하나의 item_set으로 하는 데이터를 리턴

H. candidate_set의 개수가 0이 될 때까지 D ~ G까지를 반복

2. Diagram for whole design



3. Detailed description of codes

A. 각 item들의 frequency를 계산하여 리턴하는 함수

```
def get_frequency(candidate_list):
# item_id를 [key], frequency를 value로 하는 딕셔너리 만들기
    frequent_set = {}
# k가 1이면 item_set을 생성하기 전, 즉 첫번째 DB스캔
    if k <= 1:
        for items in candidate_list:
            for item in items:
                if item in frequent_set:
                    frequent_set[item] += 1
                else:
                    frequent_set[item] = 1
    # k 값이 2 이상이 될 때
    else:
        for items in candidate_list:
            # 처음에 만든 더블 리스트를 사용하여 비교
            for transaction in original_list:
                # 각 candidate_set과 각 transaction의 합집합을 구한 결과가
                # 각 candidate_set과 같으면 해당 transaction은 item set을 포함
                temp_set = set(items) & set(transaction)
                if len(temp_set) == len(items) and tuple(items) in frequent_set:
                    frequent_set[tuple(items)] += 1
                elif len(temp_set) == len(items) and tuple(items) not in frequent_set:
                    frequent_set[tuple(items)] = 1
                else:
                    continue
```

B. Frequent_list를 가지고 candidate_set을 찾는 함수

```
def generate_candidate(frequent_list):
    candidate_list = {}
    for key in frequent_list.keys():
        # 각 항목의 frequency가 min_sup보다 작으면 패스
        if frequent_list[key] < min_sup:
            continue
        else:
            candidate_list[key] = frequent_list[key]
        return candidate_list
```

C. Candidate_set에서 support과 confidence를 찾아서 ouput파일에 입력

```
def output_candidate(candidate_set):
    output_file = ''
```

```
    # 각 후보 item_set을 가지고 1부터 하나씩 사이즈를 늘리면서 조합을 만든 후
    # 조합을 만들어 진 각 그룹들 간에 confidece를 전부 구하는 함수
    support_items = list(candidate_set.keys())
    for items in support_items:
        size = 1 # size는 조합의 묶음 단위
        # 사이즈를 늘리면서 조합을 구하기
        while size < len(items):
            temp_item_set = list(combinations(items, size))
            for item_set in temp_item_set:
                # 중복을 피하기 위해 차집합 사용
                # 가능한 모든 조합들의 confidence를 구하기
                associative_item_set = set(items) - set(item_set)
                # 처음 하나를 선택한 item_set과 차집합으로 구한 associative_set을
                # 문자열로 변환하여 outputfile 에 입력
                output_file += '{}₩t'.format(set(item_set))
                output_file += '{}₩t'.format(set(associative_item_set))
                # 각 항목들의 support과 confidence를 구하는 함수 호출
                item_set_frequency = get_item_set_frequency(item_set)
                support, confidence = get_support_confidence(candidate_set[items],
item_set_frequency)
                output_file += '{}₩t{}₩n'.format(support, confidence)
            size += 1
        return output_file
```

D. Item_set 사이즈를 늘리기 위해 후보 집합을 가지고 조합을 만들어 리턴하는 함수

```
def get_combinations(candidate_list, k):
    combination_list = []
    if k <= 2: # 각 후보조합 사이즈가 2 이하이면 combination함수 호출
        candidate = list(candidate_list.keys())
        combination_list = list(combinations(candidate, k))
    else: # 각 후보 조합이 2 이상이면 중복되는 값을 빼고 하나의 리스트로 만든 후
        # combination함수 호출
        for key in candidate_list.keys():
            for i in key:
                if i not in combination_list: # to avoid duplication item
                    combination_list.append(i)
        combination_list = list(combinations(combination_list,k))
        return combination_list
```

4. Instructions for compiling source codes

A. 과제에 명시된 것 처럼 python apriori.py 5 input.txt output.txt를 입력하여 실행

B. set() 사용으로 생긴 아이템들 간 공백은 없앴음