**Apriori algorithm**

**for association rule mining in transactional databases**

김상욱 교수님

2013002277 김재균

2017.03.28

**1. 개요**

Apriori 알고리즘은 빈발항목집합(frequent itemsets) 및 연관 규칙 탐색을 위한 알고리즘이다. Apriori 알고리즘은 트랜젝션 데이터베이스에서 빈번한 개별 항목을 식별하고 해당 항목 세트를 기준으로 더 큰 항목으로 확장함으로써 진행된다.

**2. 목적**

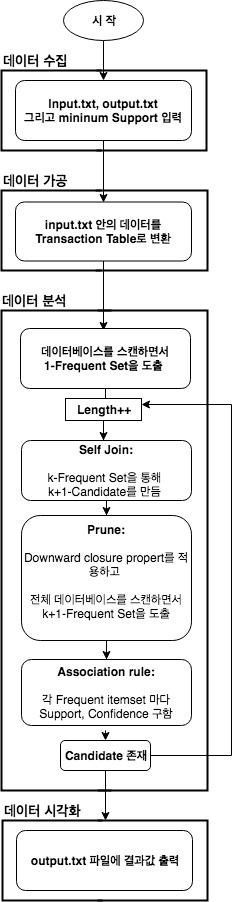
이 프로젝트의 목적은 주어진 트랜젝션 데이터와 minimum Support를 가지고 Apriori 알고리즘을 통해 연관 규칙 탐색을 구현한다. 그리고 각각의 빈번항목집합마다 Association Rule을 적용하여 지지도(Support) 및 신뢰도(Confidence)를 구한다.

**3. 개발환경**

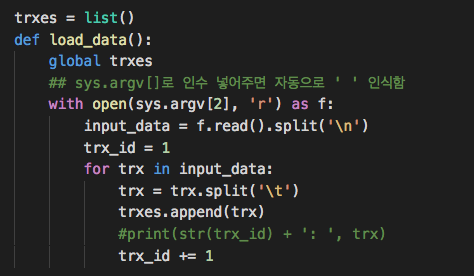
* Operating System: OS X El Captain 10.11.6 64-bit
* Language: Python 3.6.4
* Source code editor: Visual Studio Code

**4. 프로그램 구조 및 알고리즘**

프로그램 순서도는 아래와 같다.

****

**5. 코드 설명**



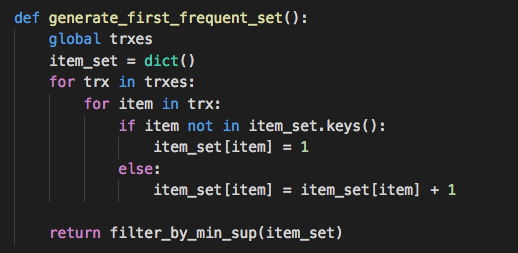
**전역변수**

- trxes는 모든 transaction을 리스트로 담는 변수

**데이터를 로드하는 함수**

- sys 모듈을 사용해서 input으로 apriori.py, minSup, input.txt, output.txt를 받는다

- input.txt의 데이터가 transaction마다 ‘\n’으로 구분되어 있고 각 transaction의 item들이 ‘\t’로 구분되어 있으므로 이를 기준으로 item\_set을 나누었다

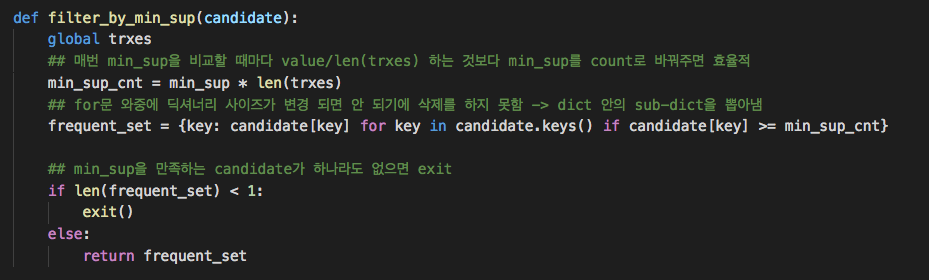


**1-Frequentset을 생성하는 함수**

- 1-Frequent set을 담을 dictionary 함수를 item\_set이라는 이름으로 만든다

- 전역변수 trxes에 있는 모든 transaction들을 Scan 하면서 고유의 item들을 key로 넣어주면서 동시에 count 해준다

- 마지막으로 minimum support의 기준에 맞지 않은 item set을 제거한다

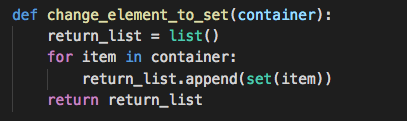


**Minimum Support를 비교하고 기준에 맞지 않은 item set을 제거해주는 함수**

- 반복문에서 매번 item set의 빈번도를 나누는 비효율성을 방지하기 위해 minimum support(확률)에 전체 transactions의 길이를 곱해주었다.

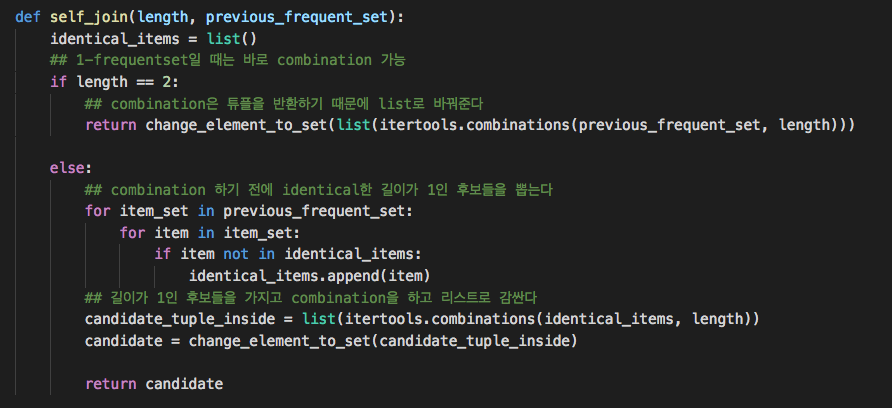
- 반복문을 돌며 minimum support의 기준에 맞는 item set들만 subset으로 뽑아준다

- minimum support를 만족하는 candidate가 하나라도 없으면 exit 한다



**List 안의 item set들을 “set” 자료형으로 바꾸는 함수**

- Prune 과정에서 하나의 item과 transaction들을 비교할 때 set 자료형이 유용해서 리스트 안의 리스트 요소들을 모두 set으로 바꾸는 역할

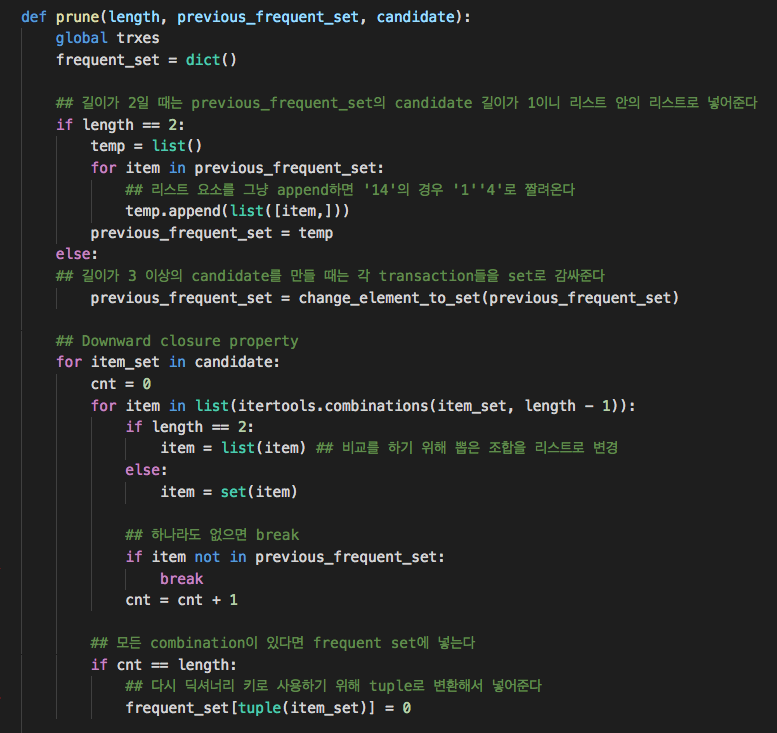
****

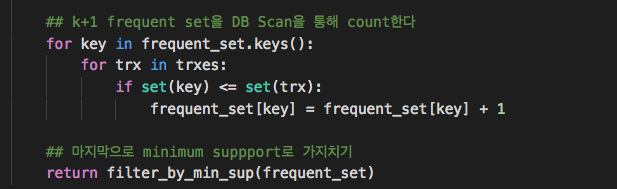
**Self Joining을 하는 함수**

- 2-candidate를 만들 때는 1-frequentset의 key들을 바로 조합한다

- 길이가 2 이상의 candidate를 만들 때는 이전 frequentset의 길이가 1인 후부들을 중복되지 않게 뽑은 후 이를 가지고 조합을 한다

- combination 함수의 경우, 반환하는 자료형이 튜플이므로 이를 모두 set으로 바꿔준다



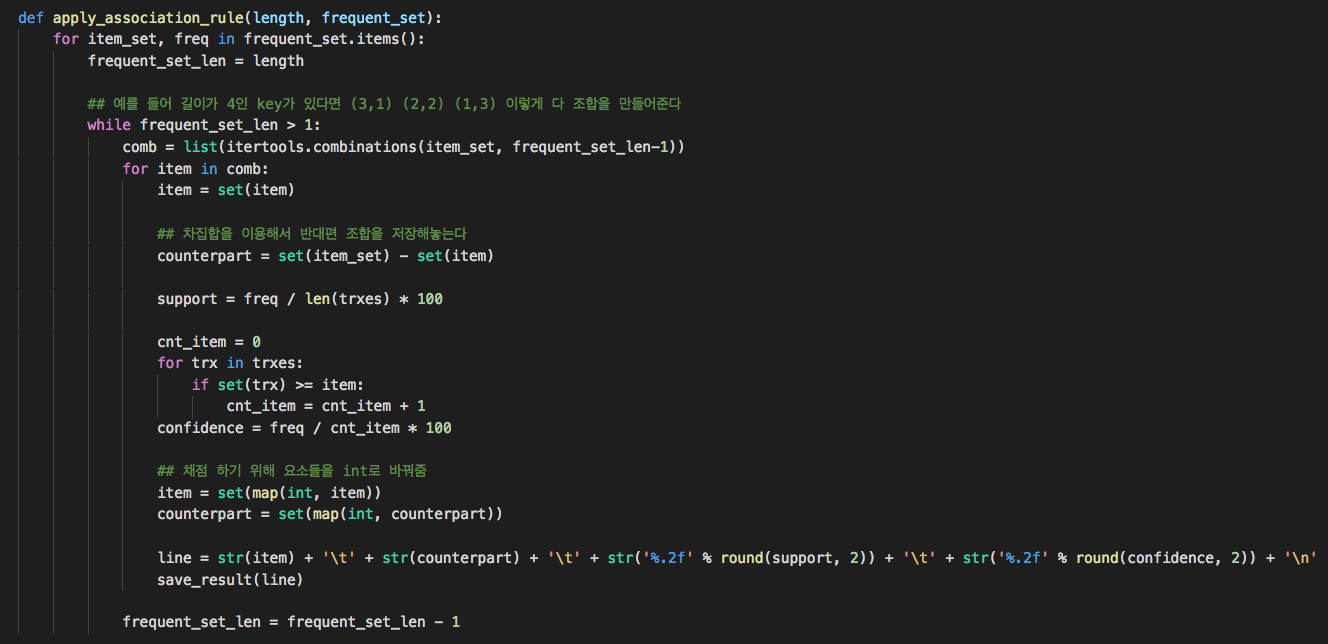
****

**Pruning을 하는 함수**

- Self join으로 만들어진 k+1-candidate와 이전 k-frequentset이랑 비교하여 downward closure property의 기준에 충족하지 않는 candidate를 가지치기 한다

- 그 후, k+1-frequentset의 item set들을 count 하기 위해 transaction을 다시 돌며 count 한다

- 마지막으로, filter\_by\_min\_sup() 함수를 통해서 minimum support의 기준에 맞지 않는 candidate를 가지치기 한다

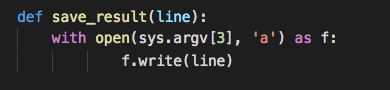
****

**Association rule을 구하는 함수**

- 각 frequentset의 item\_set들을 대상으로 지지도(Support)와 신뢰도(Confidence)를 구해준다

- 예를 들어 지지도는 A, B 가 전체 transaction 중에 같이 있을 확률

- 신뢰도는 A가 있을 때 B가 같이 있을 확률



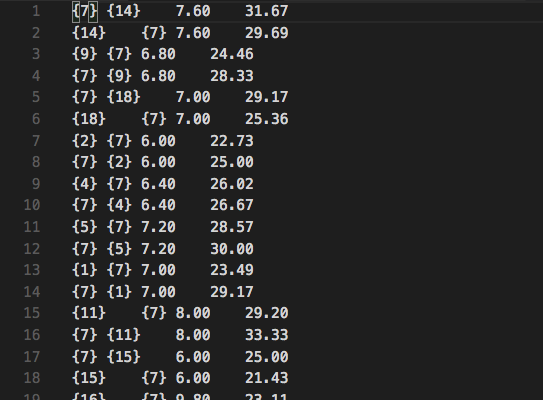
**출력 결과를 output.txt로 쓰는 함수**

**6. 가이드 라인 및 실행 결과**

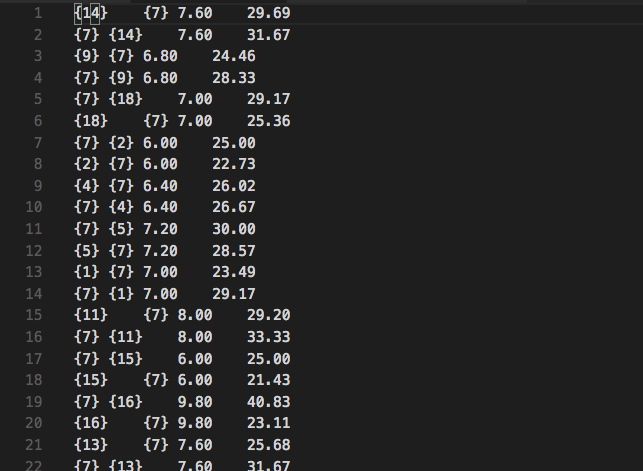
사용한 module: sys, itertools

terminal(command) 명령어: python3 apriori.py 4 input.txt output.txt

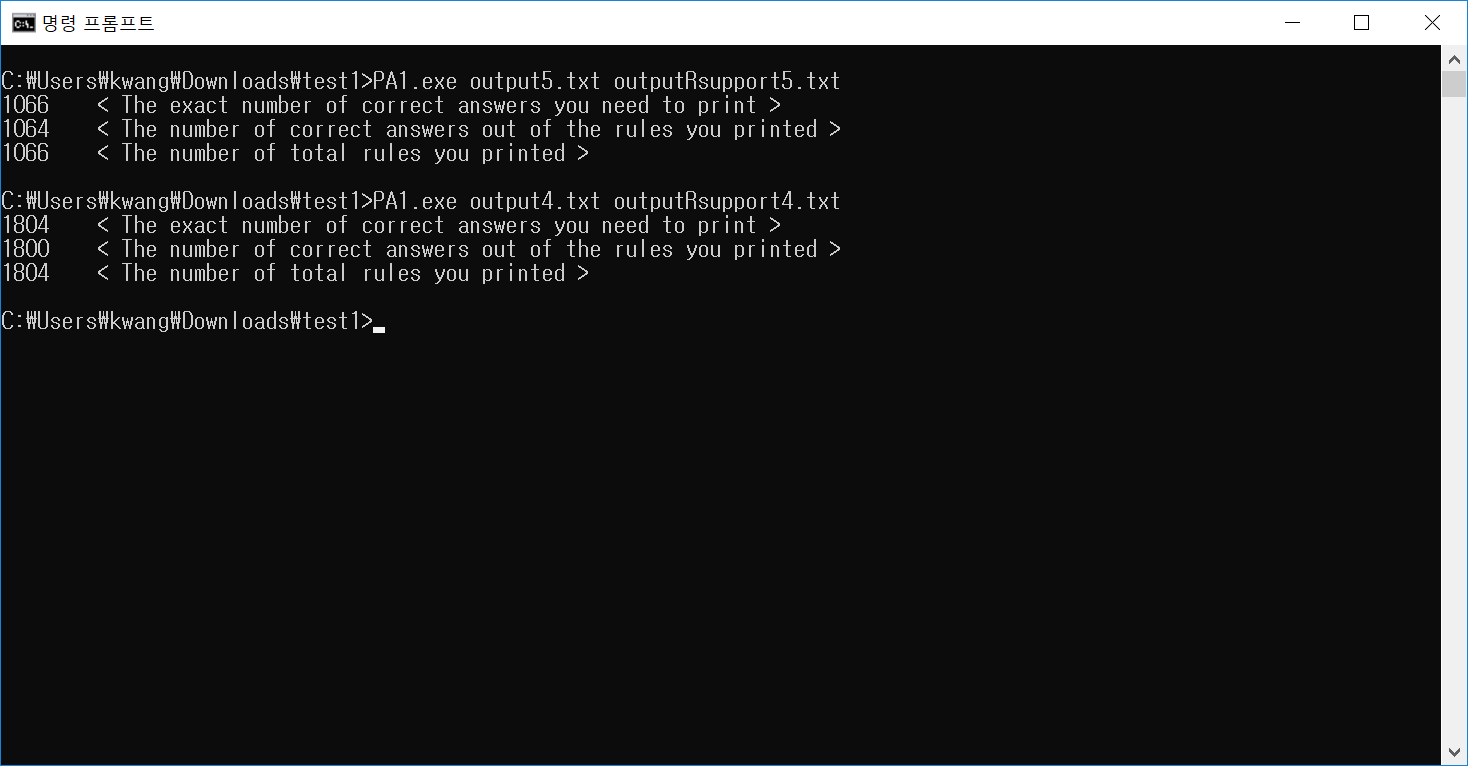
1. python3 apriori.py 4 input.txt output.txt



2. python3 apriori.py 5 input.txt output.txt



3. Test file 비교 결과



**7. 에필로그**

이번 프로젝트를 통해 빈번항목집합을 만드는 방법 중 하나인 Apriori 알고리즘을 배울 수 있

었다. Candidate를 Pruning 하는 단계에서 자료형을 여러번 조정해야 했는데, 이 과정에서 Python의 각 자료형의 장단점을 알 수 있었다.

**8. 참조**

Data mining: concepts and techniques, Morgan Kaufmann  
https://en.wikipedia.org/wiki/Apriori\_algorithm