Assignment 1 Report

박준영

1. Summary

프로그램이 시작하면 우선 database를 읽어 transactions를 메모리에 로드한다. 그 이후 길이가 1인 frequent pattern의 후보군을 뽑는다. 그런 후 전체 transaction 순회하며 각 후보군이 몇 번 나왔는지를 체크한 후, support가 threshold에 못 미치는 후보군은 제거하는 방식으로 길이 1인 frequent pattern을 뽑는다. 그 이후부터 바로 직전에 뽑은 frequent pattern 두 개를 결합하여 길이가 하나 긴 frequent pattern 후보군을 뽑는다. 예컨대 길이 4인 frequent pattern 후보는 길이 3인(직전에 뽑힌) frequent pattern을 결합하여 그중 길이가 4인 것을 모아 만든다. 후보군이 만들어지면 다시 전체 transaction을 순회하며 각 후보군이 몇 번 나왔는지를 체크한 후, support가 threshold에 못 미치는 후보군을 제거하는 방식으로 frequent pattern을 추가해 나간다. frequent pattern 후보군에서 frequent pattern을 추려내는 과정에서 각 frequent pattern이 몇 번 등장하는지를 세며, 전체 frequent pattern의 support를 dictionary에 저장한다. 이는 association rules를 구하기 위함이다.

Apriori 연산이 모두 완료되면 산출된 전체 frequent pattern에 대한 support를 바탕으로 association rule을 구한다. 길이가 2 이상인 frequent pattern을 찾아 해당 frequent pattern의 power set 중 자기 자신이 아닌 것을 구해 별도의 작업 없이 support와 confidence를 구해 output 파일에 모두 적는다. 이래도 되는 사유는 다음과 같다. 위 방식으로 구해진 $X \to Y$인 association rule에 대해 $X \cup Y$이 frequent pattern이므로 $P(X \cup Y) \geq \sup_{min}$이다. 그러므로 $X \cup Y$의 power set인 X와 Y에 대해 $\sup_{min} \leq P(X)$, $P(Y) \leq 1$을 만족한다. confidence의 정의에 따라 $P(Y|X) = \frac{P(X \cup Y)}{P(X)}$이고,

$$\frac{P(X \cup Y)}{P(X)} - \sup_{min}$$

$$P(X \cup Y) - P(X) \cdot \sup_{min}$$

위 식에서 $P(X) \leq 1$ 이므로 $P(X \cup Y) \geq \sup_{min}$, $P(X) \cdot \sup_{min} \leq \sup_{min}$ 이므로 최종적으로 $\frac{P(X \cup Y)}{P(X)} - \sup_{min} \geq 0$이므로 본 과제의 조건인 $\sup_{min} = \text{conf}_{min}$에 따라 위 과정으로 뽑힌 것은 모두 association rule이다.

2. Detailed description

2-1. if __name__ == '__main__'

프로그램의 시작 지점이다. 해당 코드 블록에선 다음의 순서대로 작업을 수행한다.

1. 명령줄 파싱
2. database 파싱
3. apriori 연산 계산
4. association rule 구한 뒤 결과 파일에 쓰기

2-2. pase_database(filename)

본 함수는 database 파일의 위치를 받아 database 파일을 파싱한다. 이후 pattern 포함관계 등을 set 관련 연산을 활용해 효과적으로 구현하기 위해 database의 모든 transaction을 set 자료구조로 저장하게 된다.

2-3. generate_candidates(data, length, not_freq_patterns)

전체 database 혹은 직전에 뽑힌 frequent pattern을 받아 length 길이의 frequent pattern 후보군을 찾는 함수이다. 이 함수는 length가 1일 때와 2일 때, 그리고 3 이상일 때 동작이 달라진다. length가 1일 때는 data는 database이며, 모든 transaction에 등장하는 itemset을 구해 반환한다. length가 2일 때는 data는 length가 1인 frequent patterns이며 단순히 모든 frequent pattern의 조합을 구해 반환한다. 마지막으로 length가 3이상일 경우엔 직전에 frequent pattern에서 배제된 pattern을 추가로 받아 어떤 조합이 not_freq_pattern에 속한 어떤 pattern의 superset인 경우 후보군에서 배제하는 방식으로 동작한다.

2-4. filter_count_candidates(data, candidates, min_support)

전체 database와 후보군, 그리고 support의 threshold값을 받아 후보군 중 frequent pattern을 찾고 각 frequent pattern의 support를 계산하는 함수이다. 이때 추가적으로 후보군 중 frequent pattern이 아닌 것의 리스트를 별도로 반환한다.

2-5. apriori(data, min_support)

database와 support의 threshold를 받는다. 다음의 순서대로 관련된 연산을 호출한다.

1. 전체 database에서 길이가 1인 frequent pattern candidates를 뽑는다.
2. 1번에서 뽑은 후보군 중 frequent pattern과 support. frequent pattern이 아닌 리스트를 구한다.
3. 더 이상 candidate가 나오지 않을 때까지 다음을 작업을 반복한다.
    A. 직전의 frequent pattern을 통해 length를 1 늘린 frequent pattern candidates를 뽑는다.
    B. 후보군 중 실제 frequent pattern을 찾는다.
    C. 전체 frequent pattern dictionary에 B에서 찾은 frequent pattern을 업데이트 한다.
    D. 만약 candidate가 1개였다면 반복을 종료한다.

3. Instructions for executing the program

우선 다음 명령을 통해 본 프로젝트를 받는다.

```
git clone https://github.com/frechele/ITE4005
```

이후 다음 명령을 통해 본 프로그램의 위치로 이동한다.

```
cd ITE4005/assignment1
```

마지막으로, 다음의 명령줄을 통해 프로그램을 실행시킨다.

```
python main.py <min_support> <input file path> <output file path>
ex) python main.py 5 input.txt output.txt
```

**** 이때, min_support의 단위는 퍼센트이다.***

4. Additional information

본 프로젝트는 다음의 환경에서 테스트되었다.

- CPU: Intel(R) Core(TM) i9-10940X
- RAM: DDR4 32GB×4 (128GB)
- OS: Ubuntu Server 20.04 LTS
- Python 3.8.8