데이터 사이언스

Apriori 알고리즘 구현.

2015004475 김태훈

1. 구동 환경

OS: Windows 10 64비트

Language: Python 3.8

2. 프로그램 구조

메인 알고리즘 전:

Argument로 minimum support, input, output file path를 받아 파싱.

Output 파일이 이미 존재하면 삭제.

Input을 읽어 리스트 형태로 저장.

Apriori 알고리즘:

C_1을 생성한 후 미니멈서포트로 필터링하여 L_1 생성. -> L_1 = L[0]

이하 L_n = L[n-1]로 표기.

While True:

L[0]을 이용해 self-join 하여 C_2 (candidate) 생성.

➔ L[0]의 각 원소의 길이는 1, 따라서 길이가 2인 후보(superset) 생성.

Candidate를 prune.

➔ Pruning principle의 대우: set의 subset 중 infrequent한 것이 하나라도 있으면, set은 infrequent하다.

이를 이용해 candidate의 subset을 모두 검사하여 최종적으로 후보가 되는 집합을 구함.

Association rule을 적용해 support와 confidence와 함께 결과 출력.

3. 코드 설명

```
read_file(path)
```

path 위치에 있는 파일을 읽어 리스트 형태로 리턴. 파일 관련 에러가 발생하면 프로그램 종료.

```
write_file(data, path=sys.argv[3])
```

data와 path를 받아 path에 data를 작성하는 함수. path의 기본값은 argv[3], 본 과제 아웃풋 파일 자리.

```
filter_by_min_sup(candidate: dict, min_sup_count: float) -> dict
```

dictionary 형태의 candidate와 minimum support를 확률이 아닌 전체 개수 대비 산정한 카운트 형태로 전달받아 dictionary를 리턴.

candidate 딕셔너리에는 아이템 집합과 그 출현 빈도가 저장되어 각 아이템 집합에 대해 카운트와 비교해 필터링 후 리턴.

```
generate_C1(transactions: list) -> dict
```

첫 candidate가 될 C1 을 생성하는 함수. 전체 transaction (이하 db) 을 인자로 받아 각 인자의 빈도를 검사해 딕셔너리 형태로 리턴.

```
self_join(itemset, k)
```

candidate 생성 과정 step 1. 아이템 집합 L_k-1(길이가 k-1인 frequent set)을 인자로 받아 길이가 k인 필터링 전 C_k를 생성한다. 즉, itemset의 superset을 생성하여 리스트 형태로 리턴.

```
prune(k, itemset, superset, min_sup_count)
```

step 2. 길이 파라미터 k, L_k-1인 itemset, 이전 단계인 셀프조인에서 받은 itemset의 superset, 미니멈서포트 카운트를 파라미터로 받는다.

superset의 subset들이 모두 frequent 해야하므로, superset으로부터 길이가 k-1인 subset을 모두 구한 후 해당 집합이 실제 k-1 frequent set인 L_k-1, 즉 itemset와 비교하여 pruning 과정을 거친다. 이후 구해낸 아이템 집합들이 실제 db에 들어있는지 비교하며 frequency count를 구해 미니멈 서포트 필터링을 거쳐 리턴한다.

```
get_support_and_confidence_with_associative_item(itemset: dict, k, db)
```

self join과 prune을 거쳐 구한 아이템 집합을 이용해 association rule을 적용한다. itemset와 길이 k, 전체 transaction data를 인자로 받는다.

confidence를 구할 때, 예를 들어 길이가 4인 아이템 집합이라면, 1->3, 2->2, 3->1 의 경우를 모두 계산해야 하므로 길이 카운트를 변동시키기 위해 별도의 변수에 저장한다. 각 아이템 집합에 대해 위 예시와 같은 방식을 적용하기 위해 길이 별 subset을 구한 후 그 차집합을 저장해 대응 관계를 구현.

요소들을 db와 비교하며 빈도를 카운트하고, support와 confidence를 구해 아웃풋 파일에 출력한다. 별도의 리턴값은 존재하지 않는다.

main 함수:

파일을 읽고 길이 k를 무한 루프에서 1씩 올리며 L_0부터 차례로 구해나가다 더 이상 candidate를 구할 수 없거나 미니멈 서포트를 만족하는 candidate가 없다면 종료한다.

본 문서를 통해 설명된 사항 이외의 코드 블록들은 아이템 집합들의 비교와 카운트 등을 위해 튜플, 리스트로 일부러 한 겹 더 감싸거나 자료형을 변환하는 내부적인 기능을 수행하므로 별도의 설명은 생략한다.

4. 소스 컴파일 가이드

본 프로그램은 windows 운영체제에서 python 3.8 가상환경을 구축한 후 개발 되었다. 기본 python 패키지만 사용되었으며 별도의 requirement는 존재하지 않는다.