

ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN
KHOA HỌC MÁY TÍNH



KHAI THÁC DỮ LIỆU VÀ ỨNG DỤNG – CS313.N22

Bài tập 3: Sử dụng ngôn ngữ Python triển khai bài toán Khai Thác Tập Phổ Biến Và Luật Kết Hợp với các thuật toán cơ bản

Nhóm 3:

Bùi Nguyễn Anh Trung (NT)	20520332
Hồ Thanh Tịnh	20520813
Nguyễn Trần Minh Anh	20520394
Lê Nguyễn Bảo Hân	20520174
Nguyễn Văn Đức Ngọc	20521666

Hồ Chí Minh, 13 tháng 03 năm 2023

Nội dung

1. Các thuật toán khai thác tập phổ biến	3
1.1. Thuật toán Apriori.....	3
1.1.1. Nội dung thuật toán.....	3
1.1.2. Các bước thực hiện.....	4
1.1.3. Mã giả.....	4
1.1.4. Ví dụ minh họa	6
1.1.5. Áp dụng Apriori khai thác luật kết hợp bằng Python	9
1.2. Thuật toán AprioriTID	11
1.2.1. Nội dung thuật toán.....	11
1.2.2. Các bước thực hiện.....	12
1.2.3. Mã giả.....	12
1.2.4. Ví dụ minh họa	13
1.2.5. Cấu trúc lưu trữ	16
1.2.6. Nhận xét.....	17
1.3. Thuật toán FP-Growth.....	17
1.3.1. Giới thiệu thuật toán.....	17
1.3.2. Các bước thực hiện.....	18
1.3.3. Mã giả.....	18
1.3.4. Ví dụ minh họa	22
1.4. Thuật toán EClat	25
1.4.1. Nội dung thuật toán.....	26
1.4.2. Các bước thực hiện.....	27
1.4.3. Mã giả.....	28
1.4.4. Nhận xét.....	29
1.4.5. Áp dụng EClat khai thác luật kết hợp bằng Python.....	29
2. Độ đo đánh giá.....	33
2.1. Support	33
2.2. Confidence	34
2.3. Lift.....	34
2.4. Conviction.....	34
3. Tham khảo.....	35

Bảng phân công

<div>Thành viên</div> <div>Công Việc</div>	Bùi Nguyễn Anh Trung	Hồ Thanh Tịnh	Nguyễn Trần Minh Anh	Lê Nguyễn Bảo Hân	Nguyễn Văn Đức Ngọc
Phân công, quản lý công việc chung	✓		✓		
Tìm hiểu thuật toán Apriori		✓			✓
Tìm hiểu thuật toán AprioriTID				✓	✓
Tìm hiểu thuật toán FP-Growth	✓	✓			
Tìm hiểu thuật toán EClat			✓	✓	
Tổng hợp nội dung và định dạng báo cáo	✓	✓	✓	✓	✓
Chuẩn bị demo Thuyết trình	✓	✓	✓	✓	✓
Làm slide	✓	✓	✓	✓	✓

Mức độ hoàn thiện (%)	100%	100%	100%	100%	100%
-----------------------	------	------	------	------	------

1. Các thuật toán khai thác tập phổ biến

Có 2 dạng khai thác tập phổ biến chính là theo chiều ngang và theo chiều dọc. Đa số các thuật toán khai thác tập phổ biến đều dựa trên cách khai thác theo chiều ngang: mỗi tập phổ biến sẽ có một mã định danh (TID) và tập phổ biến {TID: Itemset}.

Bảng 1. Khai thác theo chiều ngang

TID	List of item_IDs
T10	Coke, Slice, Kurkure
T20	Coke, Kurkure
T30	Slice, Slice

Ngược lại, cách khai thác theo chiều dọc sẽ triển khai dữ liệu theo một tập phổ biến và tập định danh của nó {Item: Tidset}

Bảng 2. Khai thác theo chiều dọc

Itemsets	TID_set
Coke	T10, T20
Slice	T10, T30
Kurkure	T10, T20
Pizza	T30

1.1. Thuật toán Apriori

1.1.1. Nội dung thuật toán

Thuật toán Apriori là gì?

Thuật toán Apriori [1] được công bố bởi R. Agrawal và R. Srikant vào năm 1994 để tìm các tập phổ biến trong một bộ dữ liệu lớn. Tên của thuật toán là Apriori vì nó sử dụng kiến thức đã có từ trước (prior) về các thuộc tính,

vật phẩm thường xuyên xuất hiện trong cơ sở dữ liệu. Để cải thiện hiệu quả của việc lọc các mục thường xuyên theo cấp độ, một thuộc tính quan trọng được sử dụng gọi là thuộc tính Apriori giúp giảm phạm vi tìm kiếm của thuật toán.

Ứng dụng của thuật toán Apriori

Thuật toán Apriori có thể nhìn vào quá khứ và đưa ra xác suất nếu một việc xảy ra kéo theo sự việc tiếp theo xảy ra. Điều này rất có ích cho các nhà kinh doanh. Ví dụ, một siêu thị muốn nghĩ cách sắp xếp các gian hàng một cách hợp lý nhất, họ có thể nhìn vào lịch sử mua hàng và sắp xếp các tập sản phẩm thường được mua cùng nhau vào một chỗ. Hoặc một trang web tin tức muốn giới thiệu cho người dùng các bài viết liên quan đến nhau nhất, cũng có thể áp dụng quy luật tương tự.

1.1.2. Các bước thực hiện

Ý tưởng: Nếu không phải là tập phổ biến thì tập bao nó cũng không phổ biến.

Phương pháp bao gồm các bước:

- **Bước 1:** Tìm tất cả các tập phổ biến 1- hạng mục.
- **Bước 2:** Tạo các tập ứng viên kích thước k-hạng mục C_k (k - candidate itemset) từ các tập phổ biến có kích thước $L_{(k-1)}$ hạng mục.
- **Bước 3:** Kiểm tra độ phổ biến của các ứng viên trên CSDL và loại các ứng viên không phổ biến.
- **Bước 4:** Dừng khi không tạo được tập phổ biến hay tập ứng viên

1.1.3. Mã giả

C_k : Tập ứng viên có kích thước là k

L_k : Tập phổ biến có kích thước là k

D : Cơ sở dữ liệu gốc

L_1 = {Tập ứng viên có kích thước là 1}

```

for (k = 1; Lk != ∅; k++) do begin    //Đến khi nào không tìm
được tập phổ biến nữa thì dừng

    Ck+1 = Tập ứng viên kích thước k+1 sinh ra từ tập phổ
biến trước Lk;

    (Ck+1 = apriori_gen(Lk))

    for (c ∈ Ck+1)

        c.count = 0;

        for (t ∈ D)    //Quét các tập giao dịch trong D, với mỗi phần
tử t trong D thì ta quét qua tất cả các phần tử c trong Ck+1 nếu
là con thì tăng biến đếm của c
        {
            for (c ∈ Ck+1 and c ⊂ t )

                c.count++;

        }

    Lk+1 = {c ∈ Ck+1 | c.count ≥ minsup}    //Tìm được Lk dựa trên
minsup cung cấp từ đầu và các c.count tính bước trên

return ∪k Lk;    //Thêm Lk vào L là tập chứa tất cả các tập phổ
biến thỏa mãn

```

Lưu ý: Thật ra để tìm được L_{k+1} từ C_{k+1} thì có 2 bước loại bỏ là ở bước

$C_{k+1} = \text{apriori_gen}(L_k)$: bước này ta tìm tập ứng viên C_{k+1} từ L_k , ta kết hợp các phần tử L_k lại để tìm ra các ứng viên có $k + 1$ phần tử sau đó xem xét các tập con ứng cử viên gồm $k + 1$ phần tử mới được tạo ra này có trong tập phổ biến L_k hay không nếu tồn tại 1 con mà không thuộc L_k thì ta loại bỏ phần tử đó ra khỏi C_{k+1}

$L_{k+1} = \{c \in C_{k+1} \mid c.\text{count} \geq \text{minsup}\}$: bước này loại bỏ các phần tử c trong C_{k+1} dựa trên *minsup* cung cấp ban đầu.

1.1.4. Ví dụ minh họa

Cho tập giao dịch D với $minsup=2/9$ (support count=2) và $mincof=70\%$

TID	List of Items
T100	I1, I2, I5
T100	I2, I4
T100	I2, I3
T100	I1, I2, I4
T100	I1, I3
T100	I2, I3
T100	I1, I3
T100	I1, I2, I3, I5
T100	I1, I2, I3

Bước 1: Tạo tập phổ biến gồm các phần tử có 1 item

Itemset	Sup.Count
{I1}	6
{I2}	7
{I3}	6
{I4}	2
{I5}	2

Ta tìm được C_1 như hình bên dựa trên phép đếm số lần mà các item xuất hiện trong các phần tử của tập giao dịch D (quét qua cơ sở dữ liệu D).

Itemset	Sup.Count
{I1}	6
{I2}	7
{I3}	6
{I4}	2
{I5}	2

Tìm được L_1 dựa trên C_1 trên và thông qua việc duyệt qua các phần tử C_1 thỏa mãn $minsup=2$ mà đề bài ban đầu yêu cầu.

Bước 2: Tạo tập phổ biến gồm các phần tử có 2 item

Itemset	Sup. Count
{11, 12}	4
{11, 13}	4
{11, 14}	1
{11, 15}	2
{12, 13}	4
{12, 14}	2
{12, 15}	2
{13, 14}	0
{13, 15}	1
{14, 15}	0

Ta sinh ra C_2 từ L_1 và duyệt qua cơ sở D đếm số lần xuất hiện của các phần tử trong C_2 . Đối chiếu với $minsup=2$, loại bỏ các phần tử có $supcount < 2$ từ C_2 ta được tập L_2 như bên dưới

Itemset	Sup Count
{11, 12}	4
{11, 13}	4
{11, 15}	2
{12, 13}	4
{12, 14}	2
{12, 15}	2

Bước 3: Tạo tập phổ biến gồm các phần tử có 3 item. Tương tự bước 1, 2 ta thu được $C_3=L_3$ như bên dưới

Itemset	Sup Count
{I1, I2, I3}	2
{I1, I2, I5}	2

Bước 4: Tạo tập phổ biến gồm các phần tử có 4 item

C_4 lúc đầu được sinh ra từ L_3 là $\{\{I1, I2, I3, I5\}\}$ nhưng sau đó tập mục này bị lược bớt vì tập con của nó $\{\{I2, I3, I5\}\}$ không là tập phổ biến có trong L_3 .

Do đó, $C_4 = \varnothing$ và thuật toán kết thúc.

$$L = \bigcup_k L_k$$

Bước 5: Tạo tập phổ biến gồm các phần tử có 3 item

Ta có $L = \{\{I_1\}, \{I_2\}, \{I_3\}, \{I_4\}, \{I_5\}, \{I_1, I_2\}, \{I_1, I_3\}, \{I_1, I_5\}, \{I_2, I_3\}, \{I_2, I_4\}, \{I_2, I_5\}, \{I_1, I_2, I_3\}, \{I_1, I_2, I_5\}\}$

- Giả sử $I = \{I_1, I_2, I_5\}$.
- Tất cả các tập con khác rỗng của nó là $\{I_1, I_2\}, \{I_1, I_5\}, \{I_2, I_5\}, \{I_1\}, \{I_2\}, \{I_5\}$.
- $\text{minconf} = 70\%$

Các luật kết hợp sinh ra từ tập I được tính với độ tin cậy như dưới đây:

$$R_1: I_1 \wedge I_2 \rightarrow I_5$$

- $\text{Confidence} = \text{suppc}\{I_1, I_2, I_5\} / \text{suppc}\{I_1, I_2\} = 2/4 = 50\%$
 $\rightarrow R_1$ bị loại do $< 70\%$

$$R_2: I_1 \wedge I_5 \rightarrow I_2$$

- $\text{Confidence} = \text{suppc}\{I_1, I_2, I_5\} / \text{suppc}\{I_1, I_5\} = 2/2 = 100\%$
 $\rightarrow R_2$ được chọn

$$R_3: I_2 \wedge I_5 \rightarrow I_1$$

- $\text{Confidence} = \text{suppc}\{I_1, I_2, I_5\} / \text{suppc}\{I_2, I_5\} = 2/2 = 100\%$
 $\rightarrow R_3$ được chọn ($> 70\%$)

$$R_4: I_1 \rightarrow I_2 \wedge I_5$$

- Confidence = $\text{suppc}\{I_1, I_2, I_5\} / \text{suppc}\{I_1\} = 2/6 = 33\%$
→ R₄ bị loại

R₅: $I_2 \rightarrow I_1 \wedge I_5$

- Confidence = $\text{suppc}\{I_1, I_2, I_5\} / \text{suppc}\{I_2\} = 2/7 = 29\%$
→ R₅ bị loại

R₆: $I_5 \rightarrow I_1 \wedge I_2$

- Confidence = $\text{suppc}\{I_1, I_2, I_5\} / \text{suppc}\{I_5\} = 2/2 = 100\%$
→ R₆ được chọn

Như vậy ta đã tìm ra được các luật kết hợp của I (I thuộc L)

Tương tự như trên ta cũng sẽ tìm được các luật kết hợp các phần tử của L và cuối cùng ta sẽ được tất cả các luật kết hợp từ tập D với *minsupp* và *minconf* ta cung cấp ban đầu.

1.1.5. Áp dụng Apriori khai thác luật kết hợp bằng Python

Mô tả tập dữ liệu

Các sản phẩm khác nhau được cung cấp 7500 giao dịch trong vòng một tuần tại một cửa hàng bán lẻ ở Pháp.

Ta dùng thư viện **apyori** để tính quy tắc kết hợp bằng Apriori.

Khai báo các thư viện sẽ dùng:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from apyori import apriori
```

Đọc dữ liệu từ file csv, ta thấy có 20 cột chính là những item và có 7501 dòng dữ liệu:

```
store_data = pd.read_csv("store_data.csv", header=None)
display(store_data.head())
print(store_data.shape)
```

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
0	shrimp	almonds	avocado	vegetables mix	green grapes	whole wheat flour	yams	cottage cheese	energy drink	tomato juice	low fat yogurt	green tea	honey	salad	mineral water	salmon	antioxydant juice	frozen smoothie	spinach	olive oil
1	burgers	meatballs	eggs	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2	chutney	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
3	turkey	avocado	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
4	mineral water	milk	energy bar	whole wheat rice	green tea	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

(7501, 20)

Tiền xử lý dữ liệu: chuyển dữ liệu ban đầu thành 1 list

```
records = []
for i in range(1, 7501):
    records.append([str(store_data.values[i, j]) for j in range(0, 20)])
```

Sử dụng thư viện apriori: Dùng dữ liệu đã được xử lý trên và cung cấp các tham số cho hàm: *min_support*, *min_confidence*, *min_lift*, *min_length*, *min_length* để tìm ra luật kết hợp cho bộ dữ liệu đầu

- *min_support (minsupp)*: Độ phổ biến ban đầu ta đặt ra để tìm ra các tập phổ biến hay còn gọi là độ phổ biến tối thiểu
- *min_confidence (min_conf)*: Độ phổ biến ban đầu ta đặt ra để tìm ra các luật kết hợp từ các tập phổ biến thông qua thuật toán apriori hay còn gọi là độ tin cậy tối thiểu
- *min_length, min_length*: giới hạn nhỏ nhất ,lớn nhất của độ lớn của tập phổ biến sinh ra luật kết hợp ta cần tìm
- *min_lift*: Sau khi tìm ra các luật kết hợp ta sử dụng giá trị này để tìm ra các luật kết hợp có giá trị. $Lift(a,b) = \frac{supp(a,b)}{(supp(a)*supp(b))}$

Ta tính được các luật kết hợp thông qua hàm apriori ở thư viện đã khai báo:

```
association_rules = apriori(records, min_support=0.0045, min_confidence=0.2, min_lift=3, min_length=2)
association_results = list(association_rules)
```

In ra các tập phổ biến cần tìm:

```
for i in range(0, len(association_results)):
    print(association_results[i][0])
```

```
frozenset({'light cream', 'chicken'})
frozenset({'escalope', 'mushroom cream sauce'})
frozenset({'escalope', 'pasta'})
frozenset({'herb & pepper', 'ground beef'})
frozenset({'tomato sauce', 'ground beef'})
frozenset({'olive oil', 'whole wheat pasta'})
frozenset({'shrimp', 'pasta'})
frozenset({'nan', 'light cream', 'chicken'})
frozenset({'shrimp', 'chocolate', 'frozen vegetables'})
frozenset({'cooking oil', 'spaghetti', 'ground beef'})
frozenset({'escalope', 'mushroom cream sauce', 'nan'})
frozenset({'escalope', 'pasta', 'nan'})
frozenset({'spaghetti', 'ground beef', 'frozen vegetables'})
frozenset({'milk', 'olive oil', 'frozen vegetables'})
frozenset({'shrimp', 'mineral water', 'frozen vegetables'})
frozenset({'spaghetti', 'olive oil', 'frozen vegetables'})
frozenset({'shrimp', 'spaghetti', 'frozen vegetables'})
frozenset({'spaghetti', 'frozen vegetables', 'tomatoes'})
frozenset({'spaghetti', 'ground beef', 'grated cheese'})
frozenset({'herb & pepper', 'ground beef', 'mineral water'})
frozenset({'herb & pepper', 'nan', 'ground beef'})
frozenset({'herb & pepper', 'spaghetti', 'ground beef'})
frozenset({'milk', 'ground beef', 'olive oil'})
frozenset({'nan', 'tomato sauce', 'ground beef'})
frozenset({'shrimp', 'spaghetti', 'ground beef'})
frozenset({'milk', 'spaghetti', 'olive oil'})
frozenset({'soup', 'mineral water', 'olive oil'})
frozenset({'nan', 'olive oil', 'whole wheat pasta'})
frozenset({'shrimp', 'nan', 'pasta'})
frozenset({'spaghetti', 'pancakes', 'olive oil'})
frozenset({'shrimp', 'chocolate', 'frozen vegetables', 'nan'})
frozenset({'cooking oil', 'nan', 'spaghetti', 'ground beef'}) ...
```

In ra các luật kết hợp cùng các giá trị *supp*, *conf*, *lift* của luật kết hợp đó:

```
for item in association_results:
    pair = item[0]
    items = [x for x in pair]
    print("Rule: " + items[0] + " -> " + items[1])
    print("Support: " + str(item[1]))
    print("Confidence: " + str(item[2][0][2]))
    print("Lift: " + str(item[2][0][3]))
    print("=====")
```

```
Confidence: 0.20574162679425836
Lift: 3.1299436124887174
=====
Rule: spaghetti -> shrimp
Support: 0.006
Confidence: 0.21531100478468898
Lift: 3.0183785717479763
=====
Rule: spaghetti -> tomatoes
Support: 0.006666666666666667
Confidence: 0.23923444976076555
Lift: 3.497579674864993
=====
Rule: nan -> grated cheese
Support: 0.005333333333333333
Confidence: 0.3225806451612903
Lift: 3.282706701098612
=====
Rule: mineral water -> herb & pepper
Support: 0.006666666666666667
Confidence: 0.390625
Lift: 3.975152645861601
=====
Rule: nan -> herb & pepper
Support: 0.0064
Confidence: 0.3934426229508197
Lift: 4.003825878061259
=====
Rule: nan -> ground beef
Support: 0.004933333333333333
Confidence: 0.22424242424242424
Lift: 3.411395906324912
```

1.2. Thuật toán AprioriTID

1.2.1. Nội dung thuật toán

Thuật toán AprioriTID [1]:

- Có cùng phương thức tạo tập ứng viên như Apriori (*apriori_gen*)
- Sau lần duyệt đầu tiên, thực hiện kiểm tra độ phổ biến (đếm *support*) của các ứng viên trên tập $\overline{C_k}$ thay vì cơ sở dữ liệu D
- Mỗi ứng viên của tập $\overline{C_k}$ có dạng $\langle TID, \{X_k\} \rangle$ với X_k là tập k -hạng mục tiềm năng có trong giao dịch t có mã là TID, hay có thể viết $\langle t.TID, \{c \in \overline{C_k} \mid c \text{ có trong } t\} \rangle$
- Giao dịch không chứa bất kì ứng viên nào từ tập k -hạng mục sẽ không được đưa vào $\overline{C_k}$. Do đó, số lượng ứng viên được đưa vào $\overline{C_k}$ có thể nhỏ hơn số lượng giao dịch trong cơ sở dữ liệu, đặc biệt là khi k có giá trị lớn

1.2.2. Các bước thực hiện

Bước 1: Quét qua các giao dịch để tìm tất cả item có độ *Support* lớn hơn *MinSupport* và đưa tập *Large 1-Itemsets* vào L_1

Bước 2: Đưa toàn bộ các TID của giao dịch cùng các items vào $\overline{C_1}$ dưới dạng $\langle TID, \{X_1\} \rangle$

Bước 3: Xây dựng các cặp 2-items từ L_1 đưa vào tập ứng viên C_2 . Quét tất cả các giao dịch trong $\overline{C_1}$ để tìm tất cả các tập *Large 2-Itemsets* từ C_2 đưa vào $\overline{C_2}$ dưới dạng $\langle TID, \{X_2\} \rangle$, đồng thời đưa các tập *Large 2-Itemsets* ứng viên vào L_2 .

Bước 4: Xây dựng các cặp k items từ L_{k-1} đưa vào tập ứng viên C_k . Quét tất cả các giao dịch trong $\overline{C_{k-1}}$ để tìm tất cả các tập *Large k-Itemsets* từ C_k và đưa vào $\overline{C_k}$ dưới dạng $\langle TID, \{X_k\} \rangle$, đồng thời đưa các tập *Large k-Itemsets* vào L_k . Lặp lại *Bước 4* cho đến khi hết ứng viên mới. Phát sinh Luật.

1.2.3. Mã giả

- 1) $L_1 = \text{Large1_Itemsets}();$
- 2) $\overline{C_1} = \text{Database } D;$
- 3) **for** ($k = 2; L_{k-1} \neq \emptyset; k++$) **do begin**

```

4)       $C_k = \text{Apriori\_Gen}(L_{k-1});$  //tạo tập ứng viên
5)       $\overline{C_k} = \emptyset$ 
6)      for all  $t \in \overline{C_{k-1}}$  do begin
7)          //xác định tập ứng viên  $C_k$  có chứa trong giao dịch t.TID
           $C_t = \{c \in C_k \mid (c - c[k]) \in t.\text{set\_of\_itemsets} \wedge$ 
               $(c - c[k-1] \in t.\text{set\_of\_itemsets})\}$ 
8)          for all candidates  $c \in C_t$  do
9)               $c.\text{count}++;$ 
10)         if  $(C_t \neq \emptyset)$  then  $\overline{C_k} += \langle t.TID, C_t \rangle$ 
11)         end
12)          $L_k = \{c \in C_k \mid c.\text{count} \geq \text{MinSupport}\}$ 
13)     end
14)      $\text{Answer} = \bigcup_k L_k$ 

```

1.2.4. Ví dụ minh họa

Cho một tập giao dịch TID với các Items như sau (với $\text{MinSupport}=50\%$, $\text{MinConfidence}=60\%$):

TID	Items
100	{1, 3, 4}
200	{2, 3, 5}
300	{1, 2, 3, 5}
400	{2, 5}

Lấy toàn bộ $\langle TID, \{X_1\} \rangle$ đưa vào $\overline{C_1}$

TID	Tập 1-item
100	{ {1}, {3}, {4} }
200	{ {2}, {3}, {5} }

300	$\{\{1\}, \{2\}, \{3\}, \{5\}\}$
400	$\{\{2\}, \{5\}\}$

Tính tập *Large1_Itemsets*, ta có L_1 :

Tập 1-item	Support
$\{1\}$	2
$\{2\}$	3
$\{3\}$	3
$\{5\}$	3

Ở bước kết từ L_1 trên ta có tập C_2 gồm các cặp 2-item:

Tập 2-item
$\{1,2\}$
$\{1,3\}$
$\{1,5\}$
$\{2,3\}$
$\{2,5\}$
$\{3,5\}$

Xác định ứng viên từ C_2 khi duyệt TID trong $\overline{C_1}$ và đưa vào $\overline{C_2}$

TID	Tập 2-item
100	$\{\{1,3\}\}$
200	$\{\{2,3\}, \{2,5\}, \{3,5\}\}$
300	$\{\{1,2\}, \{1,3\}, \{1,5\}, \{2,3\}, \{2,5\}, \{3,5\}\}$
400	$\{\{2,5\}\}$

Tính tập *Large2_Itemsets*, ta có L_2 :

Tập 2-item	Support
$\{1,3\}$	2
$\{2,3\}$	2
$\{2,5\}$	3

{3,5}	2
-------	---

Ở bước kết từ L_2 ta có tập C_3 gồm các cặp 3-item:

Tập 3-item
{2,3,5}

Xác định ứng viên từ C_3 khi duyệt TID trong $\overline{C_2}$ và đưa vào $\overline{C_3}$

TID	Tập 3-item
200	{ {2,3,5} }
300	{ {2,3,5} }

Tính tập *Large3_Itemsets*, ta có L_3 :

Tập 3-item	Support
{2,3,5}	2

Phát sinh luật:

- $2,3 \rightarrow 5$ có độ Confidence $2/2 = 100\%$
- $2,5 \rightarrow 3$ có độ Confidence $2/3 = 66,66\%$
- $3,5 \rightarrow 2$ có độ Confidence $2/2 = 100\%$

Ở bước kết từ L_3 ta có tập C_4 gồm các cặp 4-item là $\{\emptyset\}$.

Thuật toán kết thúc.

Database		$\overline{C_1}$		L_1	
TID	Items	TID	Set-of-Itemsets	Itemset	Support
100	1 3 4	100	{ {1}, {3}, {4} }	{1}	2
200	2 3 5	200	{ {2}, {3}, {5} }	{2}	3
300	1 2 3 5	300	{ {1}, {2}, {3}, {5} }	{3}	3
400	2 5	400	{ {2}, {5} }	{5}	3

C_2		$\overline{C_2}$		L_2	
Itemset		TID	Set-of-Itemsets	Itemset	Support
{1 2}		100	{ {1 3} }	{1 3}	2
{1 3}		200	{ {2 3}, {2 5}, {3 5} }	{2 3}	2
{1 5}		300	{ {1 2}, {1 3}, {1 5}, {2 3}, {2 5}, {3 5} }	{2 5}	3
{2 3}		400	{ {2 5} }	{3 5}	2
{2 5}					
{3 5}					

C_3		$\overline{C_3}$		L_3	
Itemset		TID	Set-of-Itemsets	Itemset	Support
{2 3 5}		200	{ {2 3 5} }	{2 3 5}	2
		300	{ {2 3 5} }		

Hình 1. Ví dụ minh họa thuật toán AprioriTID

1.2.5. Cấu trúc lưu trữ

- Mỗi ứng viên trong *Itemsets* sẽ được gán cho một mã số **ID** duy nhất. Mỗi tập *Itemsets* C_k được lưu trong một mảng. Mỗi ứng viên của $\overline{C_k}$ bây giờ có dạng $\langle TID, ID \rangle$
- Hàm **Apriori_Gen** phát sinh một tập các ứng viên k -ItemSet C_k bằng cách kết hợp hai tập $Large(k-1)$ -Itemsets. Mỗi ứng viên thêm hai trường:
 - Generators*
 - Extensions*
- Trường *generators* của C_k lưu các ID của hai tập $Large(k-1)$ -Itemsets kết với nhau để phát sinh C_k
- Trường *extensions* của C_k lưu các ID của các tập $Large(k+1)$ -Itemsets kết với nhau để phát sinh C_k

- Khi một ứng viên *Itemsets* C_k được phát sinh bằng cách kết hợp L_{k-1}^1 và L_{k-1}^2 thì các ID sẽ được lưu vào trường *generators* của C_k , đồng thời ID của C_k được lưu vào trường *extensions* của L_{k-1}^1

1.2.6. Nhận xét

- **Apriori:** Để xác định độ phổ biến của các tập ứng viên, Apriori luôn phải quét lại toàn bộ các giao dịch trong cơ sở dữ liệu. Điều này gây tốn rất nhiều thời gian khi giá trị của k-items tăng (số lần xét duyệt các giao dịch tăng).
- **AprioriTID:** Trong quá trình xét duyệt khởi tạo, kích thước $\overline{C_k}$ thường rất lớn và hầu hết tương đương với CSDL gốc. Ngoài ra, thuật toán AprioriTID còn phải gánh chịu thêm chi phí phát sinh nếu $\overline{C_k}$ vượt quá bộ nhớ trong mà phải sử dụng kèm bộ nhớ ngoài.

1.3. Thuật toán FP-Growth

1.3.1. Giới thiệu thuật toán

- Thuật toán FP-Growth được giới thiệu lần đầu tiên vào năm 1996 bởi Jiawei Han, Jian Pei và Yiwen Yin với tên gọi "Mining Frequent Patterns without Candidate Generation". Tuy nhiên, thuật toán này chưa được phổ biến cho đến khi phiên bản của nó được tối ưu hóa lại vào năm 2000. Tại đó, Jiawei Han, Jian Pei và Yongjian Fu đã đề xuất một phiên bản cải tiến của thuật toán mang tên "FP-Growth", giúp tăng tốc độ xử lý và giảm độ phức tạp tính toán so với phiên bản ban đầu.
- Kể từ đó, FP-Growth đã trở thành một trong những thuật toán phổ biến nhất trong khai phá dữ liệu, đặc biệt là trong việc tìm kiếm các mẫu kết hợp phổ biến. Hơn nữa, nó cũng được sử dụng rộng rãi trong các ứng dụng thực tế như phân tích dữ liệu, quản lý kho sản phẩm, đề xuất sản phẩm, phân loại tin nhắn thư rác và nhiều lĩnh vực khác.

1.3.2. Các bước thực hiện

Bước 1 : Đếm tần suất xuất hiện của các phần tử trong tập dữ liệu và xác định tập các phần tử phổ biến.

Bước 2 : Xây dựng FP-Tree từ tập dữ liệu và tập các phần tử phổ biến.

Bước 3 : Duyệt qua FP-Tree để tìm các itemset phổ biến (được biểu diễn trên FP-Tree dưới dạng các đường đi từ gốc đến lá).

Bước 4 : Dùng các itemset phổ biến để tạo các luật kết hợp.

1.3.3. Mã giả

Input: tập dữ liệu D, ngưỡng hỗ trợ tối thiểu minsup

Output: tập phổ biến

Bước 1: Đếm tần số và lọc các mục dưới minsup

```
freq = {}  
for transaction in D:  
    for item in transaction:  
        freq[item] = freq.get(item, 0) + 1  
freq = {item: count for item, count in freq.items() if count  
    >= minsup}  
freq_items = set(freq.keys())
```

Bước 2: Xây dựng cây FP-Tree

```
root = Node(None)  
for transaction in D:  
    transaction = [item for item in transaction if item in  
        freq_items]
```

```

transaction = sorted(transaction, key=lambda item:
freq[item], reverse=True)

current_node = root

for item in transaction:

    if item in current_node.children:

        current_node.children[item].count += 1

    else:

        new_node = Node(item)
        new_node.parent = current_node
        current_node.children[item] = new_node
        if freq[item] > freq[current_node.item]:
            current_node.item = item

        if item not in freq_items:
            freq_items.add(item)
            freq[item] = 0

        new_node.link = freq[item]
        freq[item] += 1

        current_node = current_node.children[item]

```

Bước 3: Khai thác tập mục phổ biến đệ quy

```

freq_itemsets = []

prefix = []

mine_fptree(root, freq, minsup, prefix, freq_itemsets)

def mine_fptree(node, freq, minsup, prefix, freq_itemsets):

```

```

# Tạo cơ sở mẫu điều kiện
cond_items = {}

while node:

    prefix_path = get_prefix_path(node)

    if len(prefix_path) > 1:

        cond_items[frozenset(prefix_path[1:])] =
node.count

    node = node.link

# Tạo Conditional FP-Tree
cond_tree, cond_freq = construct_fptree(cond_items,
minsup)

if cond_tree:

    if prefix:

        freq_set = prefix.copy()

    else:

        freq_set = set()

    freq_set.add(node.item)

    freq_itemsets.append(freq_set)

    mine_fptree(cond_tree, cond_freq, minsup,
freq_set, freq_itemsets)

def get_prefix_path(node):

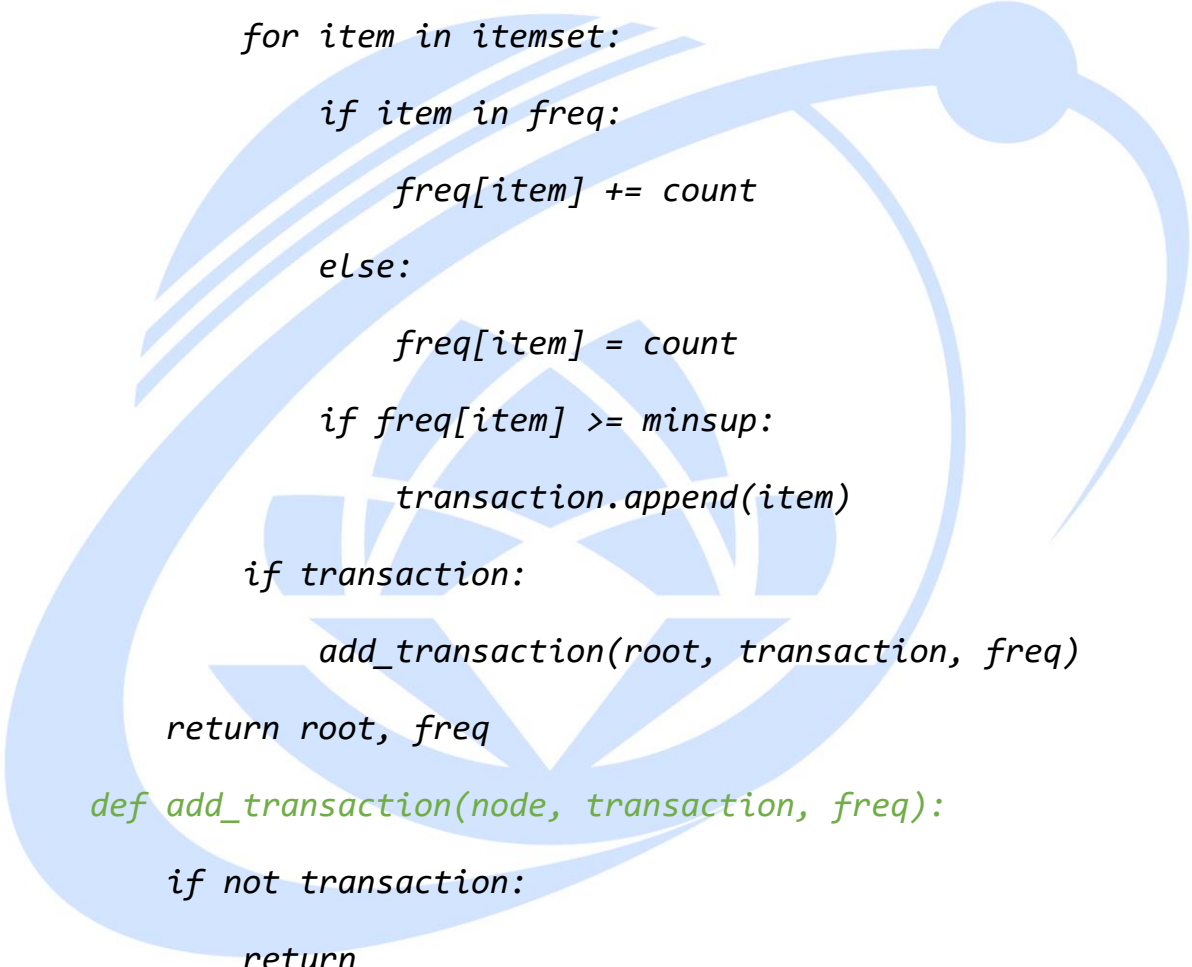
    path = []

    while node and node.item is not None:

        path.append(node.item)

        node = node.parent

```

```

    return path[::-1]

def construct_fptree(items, minsup):
    freq = {}
    root = Node(None)
    for itemset, count in items.items():
        transaction = []
        for item in itemset:
            if item in freq:
                freq[item] += count
            else:
                freq[item] = count
            if freq[item] >= minsup:
                transaction.append(item)
        if transaction:
            add_transaction(root, transaction, freq)
    return root, freq

def add_transaction(node, transaction, freq):
    if not transaction:
        return

    item = transaction[0]
    if item in node.children:
        child = node.children[item]
    else:
        child = Node(item)

```



```

    child.parent = node
    node.children[item] = child
    if freq[item] > freq[node.item]:
        node.item = item
    child.increment()
    add_transaction(child, transaction[1:], freq)

```

1.3.4. Ví dụ minh họa

TID	Item
T100	A, B, C, D, F, H, K
T200	C, D, E, K, G, E
T300	A, B, F, G, H
T400	B, C, E, F, G
T500	A, C, D, E, F

Cho minsupp=40%, minconf=70%.

Hãy tìm tập phổ biến với thuật toán FP Growth.

[Bài làm](#)

- Tập phổ biến 1 phần tử và tần suất xuất hiện của nó

Item	Supp
A	3
B	3
C	4
D	3
E	3
F	4
G	3
H	2
K	2

- Tập phổ biến 1 phần tử thỏa $\geq \text{minsupp} = 2$ và được sắp xếp theo thứ tự giảm dần.

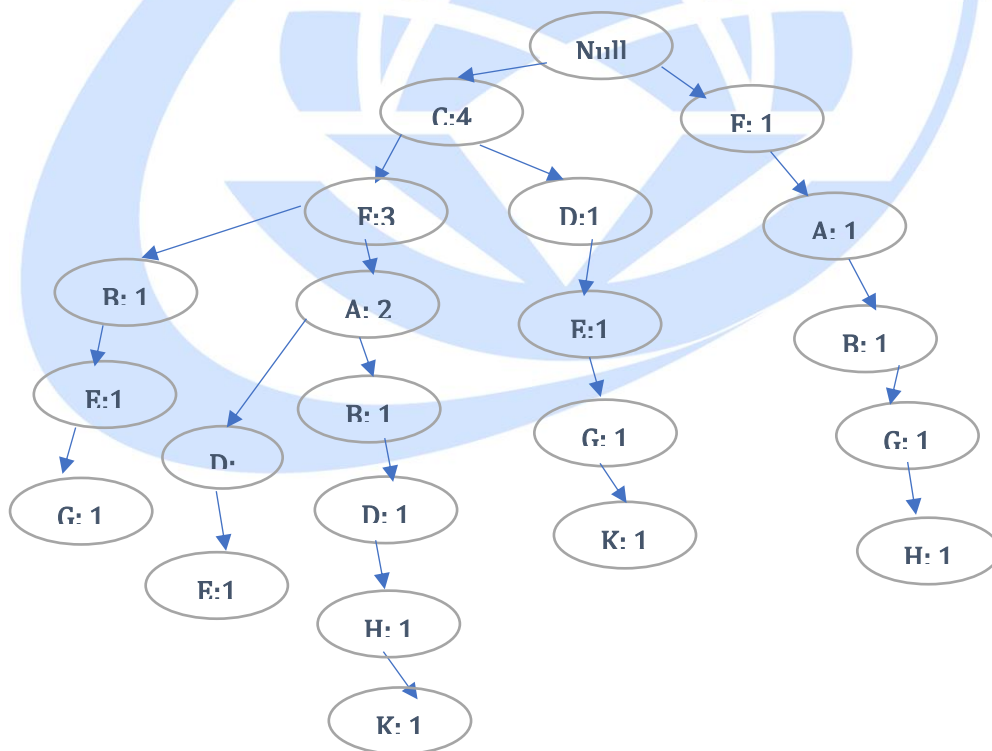
Item	Supp
C	4

F	4
A	3
B	3
D	3
E	3
G	3
H	2
K	2

- Điền vào cột Order Item Frequent

TID	Item	Order Item Frequent
T100	A, B, C, D, F, H, K	C, F, A, B, D, H, K
T200	C, D, E, K, G, E	C, D, E, G, K
T300	A, B, F, G, H	F, A, B, G, H
T400	B, C, E, F, G	C, F, B, E, G
T500	A, C, D, E, F	C, F, A, D, E

- Vẽ cây FP



- Xây dựng cơ sở điều kiện cho mỗi mục phổ biến

Item	Conditional Pattern Base
K	{C, F, A, B, D, H}: 1; {C, D, E, G}: 1
H	{C, F, A, B, D}: 1, {F, A, B, G}: 1
G	{C, F, B, E}: 1; {C, D, E}: 1; {F, A, B}: 1
E	{C, F, B}: 1; {C, F, A, D}: 1; {C, D}: 1
D	{C, F, A}: 1, {C, F, A, B}: 1, {C}: 1
B	{C, F}: 1; {C, F, A}: 1; {F, A}: 1
A	{C, F}: 2, {F}: 1
F	{C}: 3
C	{}

- Xây dựng cây FP điều kiện

Item	Conditional Pattern Base	Condition FP-Tree (supp ≥ 2)
K	{C, F, A, B, D, H}: 1; {C, D, E, G}: 1	{C:2, D:2} K
H	{C, F, A, B, D}: 1, {F, A, B, G}: 1	{F:2, A:2, B:2} H
G	{C, F, B, E}: 1; {C, D, E}: 1; {F, A, B}: 1	{C:1, F:1, B:1, E:1}, {C:1, E:1}, {F:1, B:1} G
E	{C, F, B}: 1; {C, F, A, D}: 1; {C, D}: 1	{C: 2, F:2, D: 1}, {C:1, D:1} E
D	{C, F, A}: 1, {C, F, A, B}: 1, {C}: 1	{C:3, F:2, A:2} D
B	{C, F}: 1; {C, F, A}: 1; {F, A}: 1	{C:2, F:2, A:1}, {F:1, A:1} B
A	{C, F}: 2, {F}: 1	{C, F}:2, {F}:1 A
F	{C}: 3	{C}:3 F
C	{}	{}

- Kết quả các mẫu thường xuyên

Item	Condition FP-Tree (supp ≥ 2)	Frequent Pattern
K	{C:2, D:2} K	{K}: 2, {C, K}:2, {K, D}: 2, {C, K, D}: 2
H	{F:2, A:2, B:2} H	{H}:2, {H,F}: 2, {H, A}:2, {H, B}:2, {H, F, A}: 2, {H, F, B}: 2, {H, A, B}:2, {H, F, A, B}:2
G	{C:1, F:1, B:1, E:1}, {C:1, E:1}, {F:1, B:1} G	{G}:3, {G, C}:2, {G, F}: 2, {G, B}: 2, {G, E}: 2, {C, E, G}: 2, {B, G, F}: 2

E	$\{C: 2, F:2, D: 1\}, \{C:1, D:1\} E$	$\{E\}:3, \{C, E\}: 3, \{F, E\}: 2, \{D, E\}: 2, \{D, E, C\}:2, \{F, C, E\}: 2$
D	$\{C:3, F:2, A:2\} D$	$\{D\}: 3, \{D, C\}: 3, \{D, F\}:2, \{D, A\}:2, \{D, C, F\}: 2, \{D, C, A\}:2, \{D, F, A\}:2, \{D, C, F, A\}:2$
B	$\{C:2, F:2, A:1\}, \{F:1, A:1\} B$	$\{B\}: 3, \{B, C\}:2, \{B, F\}: 3, \{B, A\}:2, \{F, A, B\}:2, \{F, C, B\}:2$
A	$\{C, F\}:2, \{F\}:1 A$	$\{A\}: 3, \{C, A\}:2, \{A, F\}:3, \{F, A, C\}:2$
F	$\{C\}:3 F$	$\{F\}:3, \{C, F\}: 3$
C	$\{\}$	$\{\}$

- Tập phổ biến 2 phần tử

$$L_2 = \{\{C, K\}:2, \{K, D\}:2, \{H, F\}:2, \{H, A\}:2, \{H, B\}:2, \{G, C\}:2, \{G, F\}:2, \{G, B\}:2, \{G, E\}:2, \{C, E\}:3, \{F, E\}:2, \{D, E\}:2, \{D, C\}:3, \{D, F\}:2, \{D, A\}:2, \{B, C\}:2, \{B, F\}:3, \{B, A\}:2, \{C, A\}:2, \{A, F\}: 3, \{C, F\}:3\}$$

- Tập phổ biến 3 phần tử

$$L_3 = \{\{C, K, D\}: 2, \{H, F, A\}:2, \{H, F, B\}:2, \{H, A, B\}:2, \{C, E, G\}:2, \{B, G, F\}:2, \{D, E, C\}:2, \{F, C, E\}:2, \{D, C, F\}: 2, \{D, C, A\}:2, \{D, F, A\}:2, \{F, A, B\}:2, \{F, C, B\}: 2, \{F, A, C\}:2\}$$

- Tập phổ biến 4 phần tử

$$L_4 = \{\{H, F, A, B\}:2, \{D, C, F, A\}:2\}$$

1.4. Thuật toán EClat

Thuật toán EClat [2] sử dụng cấu trúc IT-tree (Tidset Itemset-tree) để lưu tidset của các tập danh mục trên mỗi nút và đưa ra khái niệm lớp tương đương để kết nối các tập danh mục trong cùng một lớp tương đương để tạo ra tập danh mục mới. Thuật toán EClat chỉ cần một lần quét cơ sở dữ liệu là một tiếp cận hiện đại, tiết kiệm thời gian xử lý và có thể áp dụng khai thác tập phổ biến trên nhiều loại cơ sở dữ liệu một cách hiệu quả.

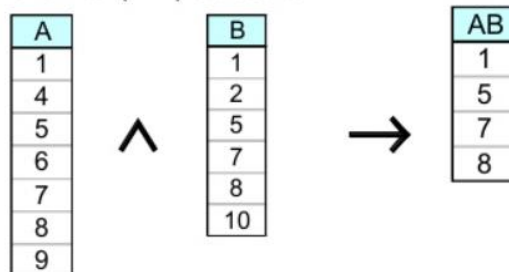
1.4.1. Nội dung thuật toán

EClaT là bài toán điển hình của cách khai thác dữ liệu theo chiều dọc và chỉ cần duyệt qua CSDL một lần duy nhất để hình thành các tập phổ biến. Sau đó, dựa trên các tập phổ biến để tìm kiếm các tập phổ biến đóng.

Có thể hình dung EClat giống như thuật toán tìm kiếm theo chiều sâu trên cây (Depth-First Search), trong khi Apriori là thuật toán tìm kiếm theo chiều rộng (Breadth-First Search). Tương tự như thuật toán tìm kiếm theo chiều sâu, EClat sẽ bắt đầu từ các lá trên cây (nốt con), tới các nốt cha để hình thành tập phổ biến.

Đồng thời, thuật toán EClat cũng không cần sinh ra các tập hợp con (subset) hay kiểm tra tập hợp con tập phổ biến như Apriori. EClat chỉ cần giao giữa hai Tidset, từ đó tạo nên tập phổ biến. Ý tưởng chính của thuật toán dựa theo tính chất này: **tập cha của các tập phổ biến cũng là tập phổ biến**. [3] [4]

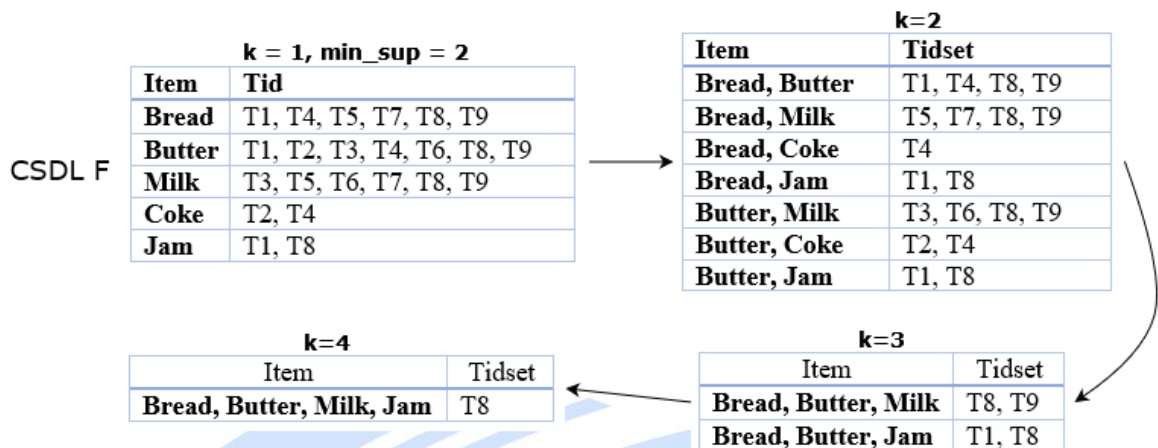
- Determine support of any k-itemset by intersecting tid-lists of two of its (k-1) subsets.



Hình 2. Giao giữa các Tidset trong thuật toán EClat

Dựa theo hình trên, giao giữa 2 Tidset của A và B, ta được một Tidset mới của AB với $support = 2$. Với trường hợp $minsup$ là 2, không cần phải duyệt qua CSDL mà chỉ cần một lần giao là đã có thể tạo ra tập phổ biến.

1.4.2. Các bước thực hiện



Hình 3. Quá trình triển khai thuật toán EClat

Dựa trên cơ sở dữ liệu F đã được khai thác theo chiều dọc như hình trên, ta bắt đầu triển khai thuật toán EClat:

- **Bước 1:** Giao Tidset của **{Bread}** lần lượt với các Tidset còn lại, sau đó từ Tidset của **{Butter}** lần lượt với các Tidset còn lại,... cho đến khi không còn 2 tập Tidset nào chưa được giao nữa.

Ta có các dãy các Tidset với $k = 2$.

- **Bước 2:** Tiếp tục giao từ Tidset của Itemset ($k = 2$) đã có từ bước 1: Giao từ Itemset **{Bread, Butter}** với các Itemset còn lại như đã làm ở bước 1

Ta có các Itemset mới với $k = 3$.

- **Bước 3:** Từ các Itemset có được từ bước 2, thực hiện giao các Tidset cho đến khi chỉ còn lại một Tidset duy nhất.

Ta có Tidset cuối cùng là **{Butter, Bread, Milk, Jam}** với $sup = 1$ (Lưu ý: Tidset cuối cùng không nhất thiết phải là tập phổ biến đóng).

- **Bước 4:** Từ các tập phổ biến trên, ta bắt đầu xét các tập phổ biến với $k \geq 2, minsup \geq 2$.

Ta có các gợi ý mua hàng như sau:

Bảng 3. Kết quả gợi ý mua hàng dựa trên EClaT

Items Bought	Recommended Products
Bread	<i>Butter</i>
Bread	<i>Milk</i>
Bread	<i>Jam</i>
Butter	<i>Milk</i>
Butter	<i>Coke</i>
Butter	<i>Jam</i>
Bread and Butter	<i>Milk</i>
Bread and Butter	<i>Jam</i>

Từ các bước thực hiện trên, ta thấy rằng thuật toán EClaT cơ bản chỉ hoạt động dựa trên phép giao giữa các Tidset. Ta chỉ cần thực hiện phép giao đó cho tới khi chỉ còn lại một Tidset cuối cùng.

Tổng quát các bước thực hiện:

1. Quét toàn bộ CSDL để xác định tập giao dịch (tidset) của các mục. Chọn các mục có *support* thỏa ngưỡng *minsup* (1-itemset).
2. Chèn 1-itemset vào mức 1 của IT-tree
3. Mỗi nút ở mức $k - 1$ kết hợp với các nút có cùng nút cha với nó tạo ra các nút ở mức k nếu *support* của các nút này thỏa ngưỡng *minsup*.
4. Lặp lại bước 3 cho đến khi không thể tạo thêm nút mới trên IT-tree.
5. Duyệt cây IT-tree để lấy ra các frequent items (Tất cả các nút trên IT-tree là frequent items).

1.4.3. Mã giả

INPUT: CSDL D , ngưỡng support σ , tiền tố của item I ($I \subseteq J$)

OUTPUT: Danh sách Itemsets $F[I](D, \sigma)$ với tiền tố được chỉ định

- 1) **Start** ($[P]$)
- 2) **For all** $X_i \in [P]$ **do**

- 3) $R = X_i \cup X_j$
- 4) $d(R) = d(X_i) - d(X_j)$
- 5) **If** $\sigma(R) \geq \text{minsup}$ **then**
- 6) $T_i = T_i \cup \{R\}$ // T_i initially empty
- 7) **If** T_i **then start** (T_i)

1.4.4. Nhận xét

Thuật toán EClat với cấu trúc IT-tree là hướng tiếp cận tốt được biết đến hiện nay với chỉ một lần quét CSDL, tiêu tốn ít bộ nhớ hơn Apriori. Tuy nhiên, phương pháp này có nhược điểm lớn là tốn bộ nhớ sử dụng để lưu tidset của các tập mục, do mỗi giao dịch chứa tập mục cần một ô nhớ. Đồng thời, bộ nhớ tạm cần thiết trong quá trình tính toán trung gian cũng rất lớn. Các hạn chế này làm cho thời gian tính toán của thuật toán EClat chưa được tối ưu.

1.4.5. Áp dụng EClat khai thác luật kết hợp bằng Python

Khai báo thư viện, sử dụng thư viện **pyECLAT** [5]:

```
import numpy as np
import pandas as pd
import plotly

!pip install pyECLAT
```

Sử dụng dataset được tích hợp sẵn trong **pyECLAT**:

```
# importing dataset (example 2 có sẵn trong pyECLAT)
from pyECLAT import Example2

dataset = Example2().get()

dataset
```

	0	1	2	3	4	5	6
0	shrimp	almonds	avocado	vegetables mix	green grapes	whole weat flour	yams
1	burgers	meatballs	eggs	NaN	NaN	NaN	NaN
2	chutney	NaN	NaN	NaN	NaN	NaN	NaN
3	turkey	avocado	NaN	NaN	NaN	NaN	NaN
4	mineral water	milk	energy bar	whole wheat rice	green tea	NaN	NaN
...
2996	green tea	NaN	NaN	NaN	NaN	NaN	NaN
2997	shrimp	cake	NaN	NaN	NaN	NaN	NaN
2998	ham	ground beef	mineral water	NaN	NaN	NaN	NaN
2999	burgers	frozen vegetables	whole wheat pasta	mineral water	chocolate	eggs	blueberries
3000	whole wheat pasta	eggs	cake	french fries	green tea	NaN	NaN

3001 rows x 7 columns

Mỗi hàng thể hiện giao dịch mua hàng của khách hàng tại siêu thị. Ví dụ: Ở hàng 1, khách hàng chỉ mua burger, thịt viên và trứng.

Chi tiết về bộ dữ liệu:

```
dataset.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3001 entries, 0 to 3000
Data columns (total 7 columns):
 #   Column  Non-Null Count  Dtype  
---  -
 0    0      3001 non-null   object  
 1    1      2315 non-null   object  
 2    2      1774 non-null   object  
 3    3      1374 non-null   object  
 4    4      1048 non-null   object  
 5    5       775 non-null   object  
 6    6       581 non-null   object  
dtypes: object(7)
memory usage: 164.2+ KB
```

Để trực quan hóa các tập phổ biến, load dữ liệu vào lớp ECLaT và tạo DataFrame nhị phân:

```
# importing ECLAT module
from pyECLAT import ECLAT

# loading transactions DataFrame to ECLAT class
eclat = ECLAT(data=dataset)

# DataFrame of binary values
eclat.df_bin
```

	cream	gums	frozen smoothie	salt	ketchup	black tea	clothes accessories	strong cheese	zucchini	spaghetti	...
0	0	0	0	0	0	0	0	0	0	0	...
1	0	0	0	0	0	0	0	0	0	0	...
2	0	0	0	0	0	0	0	0	0	0	...
3	0	0	0	0	0	0	0	0	0	0	...
4	0	0	0	0	0	0	0	0	0	0	...
...
2996	0	0	0	0	0	0	0	0	0	0	...
2997	0	0	0	0	0	0	0	0	0	0	...
2998	0	0	0	0	0	0	0	0	0	0	...
2999	0	0	0	0	0	0	0	0	0	0	...
3000	0	0	0	0	0	0	0	0	0	0	...

3001 rows x 119 columns

Mỗi hàng đại diện cho một giao dịch. Các cột là những sản phẩm có thể xuất hiện trong giao dịch. Mỗi ô chứa một trong hai giá trị:

- 0 - sản phẩm không có trong giao dịch
- 1 - sản phẩm có trong giao dịch

Sau đó, thực hiện đếm các items theo mỗi cột và hàng trong DataFrame:

<pre># count items in each column items_total = eclat.df_bin.astype(int).sum(axis=0) items_total</pre>		<pre># count items in each row items_per_transaction = eclat.df_bin.astype(int).sum(axis=1) items_per_transaction</pre>	
cream	3	0	7
gums	27	1	3
frozen smoothie	144	2	1
salt	25	3	2
ketchup	11	4	5
...
champagne	128	2996	1
barbecue sauce	22	2997	2
vegetables mix	59	2998	3
almonds	52	2999	7
antioxydant juice	18	3000	5
Length: 119, dtype: int64		Length: 3001, dtype: int64	

Minh họa phân phối của các items:

```
# load items của mỗi cột vào DataFrame
df = pd.DataFrame({'items': items_total.index, 'transactions': items_total.values})

# clone pandas DataFrame
df_table = df.sort_values("transactions", ascending=False)

# top 5 items phổ biến nhất
df_table.head(5).style.background_gradient(cmap='Blues')
```

	items	transactions
108	mineral water	711
9	spaghetti	549
16	eggs	532
33	chocolate	485
63	french fries	463

Để sinh luật kết hợp, ta cần định nghĩa:

- *Minimum Support*: độ phổ biến tối thiểu
- *Minumum Combinations*: số lượng items tối thiểu trong giao dịch
- *Maximum Combinations*: số lượng items tối đa trong giao dịch

Lưu ý: Maximum Combinations càng cao thì thời gian tính toán càng lâu.

```
# the item should appear at least at 5% of transactions
min_support = 5/100

# start from transactions containing at least 2 items
min_combination = 2

# up to maximum items per transaction
max_combination = max(items_per_transaction)
rule_indices, rule_supports = eclat.fit(min_support=min_support,
                                       min_combination=min_combination,
                                       max_combination=max_combination,
                                       separator=' & ',
                                       verbose=True)
```

```
Combination 2 by 2
253it [00:01, 151.65it/s]
Combination 3 by 3
1771it [00:11, 154.83it/s]
Combination 4 by 4
8855it [01:00, 145.40it/s]
Combination 5 by 5
33649it [04:09, 134.97it/s]
Combination 6 by 6
100947it [13:15, 126.86it/s]
Combination 7 by 7
245157it [33:34, 121.69it/s]
```

Phương thức fit() của lớp ECLAT trả về:

- Chỉ số của luật kết hợp
- Giá trị support của luật kết hợp

```
result = pd.DataFrame(rule_supports.items(), columns=['Item', 'Support'])
result.sort_values(by=['Support'], ascending=False)
```

	Item	Support
0	spaghetti & mineral water	0.060646

2. Độ đo đánh giá























2.1. Support

Support cho biết mức độ phổ biến của một danh mục, được đo bằng tỉ lệ giao dịch mà một danh mục xuất hiện. [6]

Trong bảng 4 bên dưới, support của {táo} là 4 trên 8, hay 50%. Các danh mục cũng có thể chứa nhiều mục. Chẳng hạn, support của {táo, bia, gạo} là 2 trên 8, hay 25%.

$$\text{Support } \{\text{táo}\} = \frac{4}{8}$$

Bảng 4. Ví dụ các danh mục

Transaction 1	   
Transaction 2	  
Transaction 3	 
Transaction 4	 
Transaction 5	   
Transaction 6	  
Transaction 7	 
Transaction 8	 

Nếu doanh số của một mặt hàng vượt quá một tỷ lệ nhất định và có xu hướng tác động đáng kể đến lợi nhuận, có thể cân nhắc sử dụng tỷ lệ đó làm ngưỡng. Xác định các danh mục có giá trị support trên ngưỡng này là danh mục quan trọng.

2.2. Confidence

Confidence cho biết khả năng mặt hàng Y được mua khi mặt hàng X được mua, biểu thị bằng $\{X \rightarrow Y\}$, được đo bằng tỷ lệ giao dịch của mặt hàng X mà trong đó mặt hàng Y cũng xuất hiện.

Trong Bảng 4, độ tin cậy của $\{\text{táo} \rightarrow \text{bia}\}$ là 3 trên 4, hay 75%.

$$\text{Confidence} \{\text{🍎} \rightarrow \text{🍺}\} = \frac{\text{Support} \{\text{🍎}, \text{🍺}\}}{\text{Support} \{\text{🍎}\}}$$

Nhược điểm của *Confidence* là có thể trình bày sai tầm quan trọng của một mối liên hệ. Điều này là do nó chỉ giải thích mức độ phổ biến của táo chứ không phải bia. Nếu bia nói chung cũng rất phổ biến thì nhiều khả năng giao dịch có táo cũng sẽ chứa bia, do đó làm tăng độ tin cậy.

2.3. Lift

Lift cho biết khả năng mua mặt hàng Y khi mặt hàng X được mua (Y phụ thuộc vào X), đồng thời kiểm soát mức độ phổ biến của mặt hàng Y.

Trong Bảng X, mức tăng của $\{\text{táo} \rightarrow \text{bia}\}$ là 1, nghĩa là không có mối liên hệ nào giữa các mặt hàng. Giá trị nâng *Lift* hơn 1 có nghĩa là mặt hàng Y có khả năng được mua nếu mặt hàng X được mua, trong khi giá trị nhỏ hơn 1 có nghĩa là mặt hàng Y khó có thể được mua nếu mặt hàng X được mua.

$$\text{Lift} \{\text{🍎} \rightarrow \text{🍺}\} = \frac{\text{Support} \{\text{🍎}, \text{🍺}\}}{\text{Support} \{\text{🍎}\} \times \text{Support} \{\text{🍺}\}}$$

2.4. Conviction

Conviction cho biết độ tin tưởng vào sự phụ thuộc giữa mặt hàng Y khi mặt hàng X được mua.

Nhận xét mức độ tin cậy cũng tương tự như *Lift*, khi *Conviction* lớn hơn 1, nghĩa là vẫn có khả năng tin tưởng Y được mua khi X được mua.

$$\text{Conv} \{ \text{🍎} \rightarrow \text{🍺} \} = \frac{1 - \text{Support} \{ \text{🍎} \} \times \text{Support} \{ \text{🍺} \}}{1 - \text{Support} \{ \text{🍎, 🍺} \}}$$

3. Tham khảo

- [1] R. a. S. R. Agrawal, "Fast Algorithms for Mining Association Rules," *Proc. 20th Int. Conf. Very Large Data Bases VLDB*, 2000.
- [2] M. Zaki, "Scalable algorithms for association mining. IEEE Trans Knowl Data Eng," *Knowledge and Data Engineering, IEEE Transactions on*, pp. 372 - 390, 2000.
- [3] "ML | ECLAT Algorithm," [Online]. Available: <https://www.geeksforgeeks.org/ml-eclat-algorithm/>.
- [4] "Association Rule Mining using ECLAT Algorithm," [Online]. Available: <https://medium.com/machine-learning-and-artificial-intelligence/3-4-association-rule-mining-using-eclat-algorithm-b6e50aab2147>.
- [5] "pyECLAT 1.0.2," [Online]. Available: <https://pypi.org/project/pyECLAT/>.
- [6] "Association Rules and the Apriori Algorithm: A Tutorial," [Online]. Available: <https://www.kdnuggets.com/2016/04/association-rules-apriori-algorithm-tutorial.html>.