

TẬP PHỔ BIẾN & LUẬT KẾT HỢP

A. MỤC TIÊU CHƯƠNG

Hướng dẫn sinh viên triển khai thực hiện việc tìm tập phổ biến bằng giải thuật Apriori.

Hướng dẫn sinh viên trích xuất luật kết hợp từ tập phổ biến và cách thức trình bày một luật kết hợp

Hướng dẫn sinh viên ôn tập lý thuyết về tập phổ biến và luật kết hợp

Hướng dẫn sinh viên cách tiếp cận để tìm kiếm các đoạn mã nguồn bằng Python hỗ trợ cho việc xây dựng các giải thuật

B. KẾT CẤU CHƯƠNG

Chương bao gồm 2

- Xác định tập phổ biến với giải thuật Apriori
- Xác định luật kết hợp từ tập phổ biến

C. NỘI DUNG CHƯƠNG

4.1. XÁC ĐỊNH TẬP PHỔ BIẾN VỚI GIẢI THUẬT APRIORI

4.1.1. Ôn tập lý thuyết với GPT

- + Giải thuật Apriori hoạt động như thế nào? Hãy giải thích các bước chính trong quá trình tìm tập phổ biến
- + Các tham số như support, confidence, và lift có ý nghĩa gì trong khai thác luật kết hợp? Chúng được sử dụng như thế nào trong Apriori?
- + Sự khác biệt giữa giải thuật Apriori và các thuật toán khác như FP-Growth trong việc tìm tập phổ biến là gì?
- + Viết đoạn code mẫu bằng Python (sử dụng thư viện như **mlxtend** hoặc **apyori**) để triển khai giải thuật Apriori không? Hãy mô tả các bước thực hiện
- + Làm thế nào để tiền xử lý dữ liệu giao dịch (transactional data) trong Python trước khi áp dụng Apriori?
- + Hàm nào trong Python để tính toán tập phổ biến và luật kết hợp từ giải thuật Apriori? Hãy chia sẻ một đoạn code mẫu
- + Điều chỉnh ngưỡng support và confidence trong Apriori để đạt được kết quả tốt hơn không? Hãy mô tả cách thực hiện
- + Cách trực quan hóa các luật kết hợp (association rules) dưới dạng biểu đồ mạng (network graph) hoặc heatmap trong Python

4.1.2. Bài làm mẫu

Bài toán 1: Tìm tập phổ biến trong dữ liệu giao dịch mua hàng tại siêu thị. Dữ liệu có thể lấy từ <https://www.kaggle.com/code/mgmarques/customer-segmentation-and-market-basket-analysis>

Nhiệm vụ 1: sử dụng thư viện mlxtend để tìm tập phổ biến có trong dữ liệu

1. Cài đặt các thư viện **mlxtend** và import các gói dữ liệu

```
# To install mlxtend, go to the Anaconda prompt and execute pip
install mlxtend
!pip install mlxtend
import matplotlib.pyplot as plt
import mlxtend.frequent_patterns
import mlxtend.preprocessing
import numpy
import pandas
```

2. Thực hiện một số tính toán để hiểu các khái niệm cơ bản như support, confident và lift

```
# danh sách các giao dịch mua hàng
example = [
    ['milk', 'bread', 'apples', 'cereal', 'jelly', 'cookies', 'salad', 'tomatoes'],
    ['beer', 'milk', 'chips', 'salsa', 'grapes', 'wine', 'potatoes', 'eggs', 'carrots'],
    ['diapers', 'baby formula', 'milk', 'bread', 'chicken', 'asparagus', 'cookies'],
    ['milk', 'cookies', 'chicken', 'asparagus', 'broccoli', 'cereal', 'orange juice'],
    ['steak', 'asparagus', 'broccoli', 'chips', 'salsa', 'ketchup', 'potatoes', 'salad'],
    ['beer', 'salsa', 'asparagus', 'wine', 'cheese', 'crackers', 'strawberries', 'cookies'],
    ['chocolate cake', 'strawberries', 'wine', 'cheese', 'beer', 'milk', 'orange juice'],
    ['chicken', 'peas', 'broccoli', 'milk', 'bread', 'eggs', 'potatoes', 'ketchup', 'crackers'],
    ['eggs', 'bread', 'cheese', 'turkey', 'salad', 'tomatoes', 'wine', 'steak', 'carrots'],
    ['bread', 'milk', 'tomatoes', 'cereal', 'chicken', 'turkey', 'chips', 'salsa', 'diapers']
]
# Tính support
# the number of transactions
N = len(example)
# the frequency of milk
f_x = sum(['milk' in i for i in example])
# the frequency of bread
f_y = sum(['bread' in i for i in example])
# the frequency of milk and bread
f_xy = sum([
    all(w in i for w in ['milk', 'bread'])
    for i in example
])
# support (supp)
support = f_xy / N
print("Support = {}".format(round(support, 4))) # support = 0.4
# confidence: x -> y
confidence = support / (f_x / N)
print("Confidence = {}".format(round(confidence, 4))) # Confidence = 0.5714
# lift: x -> y
lift = confidence / (f_y / N)
```

```

print("Lift = {}".format(round(lift, 4))) # Lift = 1.1429
# leverage: x -> y
leverage = support - ((f_x / N) * (f_y / N))
print("Leverage = {}".format(round(leverage, 4)))
# conviction: x -> y
conviction = (1 - (f_y / N)) / (1 - confidence)
print("Conviction = {}".format(round(conviction, 4)))

```

3. Nạp dữ liệu, làm sạch dữ liệu và biến đổi dữ liệu về định dạng phù hợp để thực hiện xây dựng mô hình (chuyển dữ liệu từ 1 hóa đơn có nhiều dòng thành 1 hóa đơn chỉ có 1 dòng chứa các sản phẩm)

```

online = pandas.read_excel( io="Online Retail.xlsx",
    sheet_name="Online Retail",
    header=0
)
# create new column called IsCPresent
online['IsCPresent'] = ( # looking for C in InvoiceNo column
    online['InvoiceNo'] # convert column to string type for the apply
        function below
    .astype(str) # set element to 1 if C present otherwise 0
    .apply(lambda x: 1 if x.find('C') != -1 else 0)
)
online1 = (
    online # filter out non-positive quantity values
    .loc[online["Quantity"] > 0] # remove InvoiceNos starting with C
    .loc[online['IsCPresent'] != 1] # column filtering
    .loc[:, ["InvoiceNo", "Description"]]# dropping all rows with at
    least
        one missing value
    .dropna()
)
# extract unique invoice numbers as list
invoice_no_list = online1.InvoiceNo.tolist()
invoice_no_list = list(set(invoice_no_list))
print("Length of list of invoice numbers:
    {}ln{}".format(ln=len(invoice_no_list)))

# get 5000 transaction
subset_invoice_no_list = invoice_no_list[0:5000]
# filter data set down to based on subset of invoice number list
online1 = online1.loc[online1["InvoiceNo"].isin(subset_invoice_no_list)]

```

```

invoice_item_list = []
for num in list(set(online1.InvoiceNo.tolist())):
    # filter data set down to one invoice number
    tmp_df = online1.loc[online1['InvoiceNo'] == num]
    # extract item descriptions and convert to list
    tmp_items = tmp_df.Description.tolist()
    # append list invoice_item_list
    invoice_item_list.append(tmp_items)
print(invoice_item_list[1:5])

```

Kết quả thực hiện

```

[['HAND WARMER UNION JACK', 'HAND WARMER RED POLKA DOT'], ['ASSORTED
COLOUR BIRD ORNAMENT', "POPPY'S PLAYHOUSE BEDROOM ", "POPPY'S PLAYHOUSE
KITCHEN", 'FELTCRAFT PRINCESS CHARLOTTE DOLL', 'IVORY KNITTED MUG COSY ',
'BOX OF 6 ASSORTED COLOUR TEASPOONS', 'BOX OF VINTAGE JIGSAW BLOCKS ',
'BOX OF VINTAGE ALPHABET BLOCKS', 'HOME BUILDING BLOCK WORD', 'LOVE
BUILDING BLOCK WORD', 'RECIPE BOX WITH METAL HEART', 'DOORMAT NEW
ENGLAND'], ['JAM MAKING SET WITH JARS', 'RED COAT RACK PARIS FASHION',
'YELLOW COAT RACK PARIS FASHION', 'BLUE COAT RACK PARIS FASHION'], ['BATH
BUILDING BLOCK WORD']]

```

4. Mã hóa dữ liệu về dạng dữ liệu mà giải thuật apriori trong gói mlxtend yêu cầu

```

online_encoder = mlxtend.preprocessing.TransactionEncoder()
online_encoder_array = online_encoder.fit_transform(invoice_item_list)
online_encoder_df = pandas.DataFrame(
    online_encoder_array,
    columns=online_encoder.columns_
)
# this is a very big table, so for more
# easy viewing only a subset is printed
online_encoder_df.loc[
    4970:4979,
    online_encoder_df.columns.tolist()[0:8]
]
print("Data dimension (row count, col count):
        {dim}".format(dim=online_encoder_df.shape))

```

Kết quả thực hiện

	¹⁰ ₀₁ 4 PURPLE FLOCK DINNER C...	¹⁰ ₀₁ OVAL WALL MIRROR DIAMANTE	¹⁰ ₀₁ SET 2 TEA TOWE
4970	False	False	False
4971	False	False	False
4972	False	False	False
4973	False	False	• True
4974	False	False	False
4975	False	False	False
4976	False	False	False
4977	False	False	False
4978	False	False	False
4979	False	False	False

5. Tìm tập phổ biến bằng giải thuật apriori

```
# case 1: default minimum support = 0.5 does not use colnames (item names)
mod = mlxtend.frequent_patterns.apriori(online_encoder_df)

# case 2:
mod_minsupport = mlxtend.frequent_patterns.apriori(
    online_encoder_df,
    min_support=0.01
)
mod_minsupport.loc[0:6]

# case 3: add colnames for easier interpretability
mod_colnames_minsupport = mlxtend.frequent_patterns.apriori(
    online_encoder_df,
    min_support=0.01,
    use_colnames=True
)
mod_colnames_minsupport['length'] = (
    mod_colnames_minsupport['itemsets'].apply(lambda x: len(x))
)
mod_colnames_minsupport.loc[0:6]
```

4.1.3. Bài tập trên lớp

Bài toán 2: Tìm tập phổ biến cho bài toán đề xuất phim. Dữ liệu lấy tại <https://grouplens.org/datasets/movielens> với MovieLens 100K dataset

Nhiệm vụ 1: Tìm tập phổ biến có trong tập dữ liệu phim

1. Import các gói thư viện, nạp dữ liệu và tiền xử lý dữ liệu vào notebook

```
import os
import pandas as pd
data_folder = "data/ml-100k"
ratings_filename = os.path.join(data_folder, "u.data")
all_ratings = pd.read_csv(ratings_filename, delimiter="\t", header=None,
```

```

        names = ["UserID", "MovieID", "Rating", "Datetime"])
all_ratings["Datetime"] = pd.to_datetime(all_ratings['Datetime'],
                                         unit='s')

# Tạo cột mới tên Favorable
all_ratings["Favorable"] = all_ratings["Rating"] > 3
ratings = all_ratings[all_ratings['UserID'].isin(range(200))]
favorable_ratings = ratings[ratings["Favorable"]]
favorable_reviews_by_users = dict((k, frozenset(v.values)) for k, v in
                                    favorable_ratings.groupby("UserID")["MovieID"])
num_favorable_by_movie = ratings[["MovieID",
                                    "Favorable"]].groupby("MovieID").sum()
num_favorable_by_movie.sort_values(by="Favorable",
                                   ascending=False).head()

```

Kết quả thực hiện

Favorable	
MovieID	Favorable
50	100
100	89
258	83
181	79
174	74

2. Tạo hàm để tìm tập phổ biến

```

from collections import defaultdict
def find_frequent_itemsets(favorable_reviews_by_users, k_1_itemsets,
min_support):
    counts = defaultdict(int)
    for user, reviews in favorable_reviews_by_users.items():
        for itemset in k_1_itemsets:
            if itemset.issubset(reviews):
                for other_reviewed_movie in reviews - itemset:
                    current_superset = itemset |
                        frozenset((other_reviewed_movie,))
                    counts[current_superset] += 1
    return dict([(itemset, frequency) for itemset, frequency in
                counts.items() if frequency >= min_support])

```

3. Tìm tập phổ biến

```
import sys
```

```

frequent_itemsets = {} # itemsets are sorted by length
min_support = 50

# k=1 candidates are the isbns with more than min_support favourable
reviews

frequent_itemsets[1] = dict((frozenset((movie_id,)), row["Favorable"])
                            for movie_id, row in num_favorable_by_movie.iterrows()
                            if row["Favorable"] > min_support)

print("There are {} movies with more than {} favorable
      reviews".format(len(frequent_itemsets[1]), min_support))

sys.stdout.flush()

for k in range(2, 20):
    # Generate candidates of length k, using the frequent itemsets of
    # length k-1

    # Only store the frequent itemsets
    cur_frequent_itemsets =
        find_frequent_itemsets(favorable_reviews_by_users,
                               frequent_itemsets[k-1], min_support)
    if len(cur_frequent_itemsets) == 0:
        print("Did not find any frequent itemsets of length
{}.".format(k))
        sys.stdout.flush()
        break
    else:
        print("I found {} frequent itemsets of length
{}.".format(len(cur_frequent_itemsets), k))
        #print(cur_frequent_itemsets)
        sys.stdout.flush()
        frequent_itemsets[k] = cur_frequent_itemsets
# We aren't interested in the itemsets of length 1, so remove those
del frequent_itemsets[1]

```

Kết quả thực hiện

```

There are 16 movies with more than 50 favorable reviews
I found 93 frequent itemsets of length 2
I found 295 frequent itemsets of length 3
I found 593 frequent itemsets of length 4
I found 785 frequent itemsets of length 5
I found 677 frequent itemsets of length 6
I found 373 frequent itemsets of length 7
I found 126 frequent itemsets of length 8
I found 24 frequent itemsets of length 9
I found 2 frequent itemsets of length 10
Did not find any frequent itemsets of length 11

```

4.1.4. Bài tập về nhà

Tìm hiểu thư viện **apyori** (<https://pypi.org/project/apyori>) để thực hiện tìm tập phổ biến từ dữ liệu mua hàng tại siêu thị. Tham khảo bài hướng dẫn <https://www.kaggle.com/code/rockystats/apriori-algorithm-or-market-basket-analysis>

4.2. XÁC ĐỊNH LUẬT KẾT HỢP TỪ TẬP PHỔ BIẾN

4.2.1. Ôn tập lý thuyết với GPT

- + Luật kết hợp (association rules) là gì? Hãy giải thích các khái niệm support, confidence, và lift trong luật kết hợp
- + Quá trình tìm luật kết hợp từ tập phổ biến bao gồm những bước nào? Làm thế nào để chuyển từ tập phổ biến sang luật kết hợp?
- + Khi nào một luật kết hợp được coi là "mạnh" (strong rule)? Các chỉ số như lift hoặc confidence ảnh hưởng như thế nào đến đánh giá này?
- + Viết đoạn code mẫu bằng Python (sử dụng thư viện như mlxtend hoặc apyori) để tìm luật kết hợp từ tập phổ biến không? Hãy mô tả các bước thực hiện

4.2.2. Bài làm mẫu

Bài toán 1: Xác định các luật kết hợp có trong tập dữ liệu giao dịch mua hàng tại siêu thị. Dữ liệu lấy từ <https://www.kaggle.com/code/mgmarques/customer-segmentation-and-market-basket-analysis>

Nhiệm vụ 1: Xác định các luật kết hợp

1. Thực hiện các bước từ 1 đến 4 và chọn 1 trong 3 case của bước 5 để tìm tập phổ biến có trong dữ liệu giao dịch dạng bảng của siêu thị
2. Trích các luật có trong tập phổ biến đã thực hiện ở bước 1

```
# case 1: hiển thị 6 luật trong 1064 luật được tìm thấy
rules = mlxtend.frequent_patterns.association_rules(
    mod_colnames_minsupport,
    metric="confidence",
    min_threshold=0.6,
    support_only=False
)
rules.loc[0:6]

# case 2: liệt kê các rule thỏa mãn điều kiện min_threshold
rules2 = mlxtend.frequent_patterns.association_rules(
    mod_colnames_minsupport,
    metric="lift", #có thể sử dụng support, confidence, ...
    min_threshold=50,
    support_only=False
)
rules2.loc[0:6]
```

```
print("Number of Associations: {}".format(rules2.shape[0])) # 170 luật
```

Kết quả thực hiện case 1

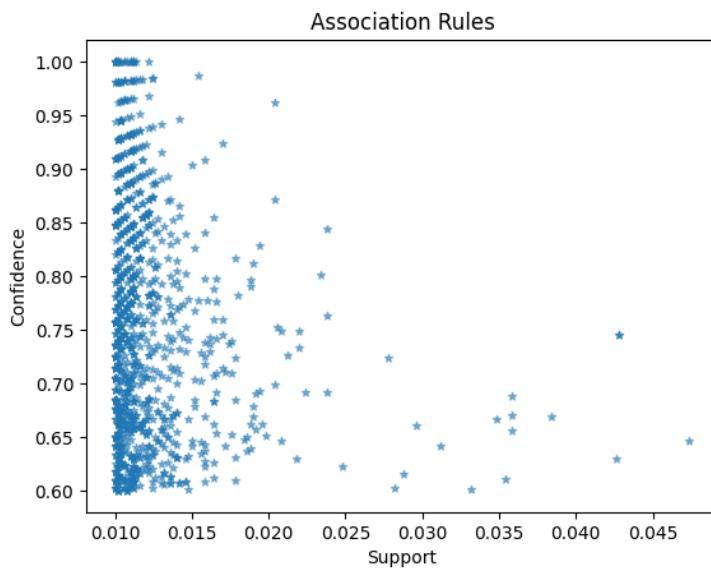
	antecedents	consequents	antecedent support	consequent sup...	support	confidence	lift	1
1	(ALARM CLOCK BAKELIKE CHOCOLATE)	(ALARM CLOCK BAKELIKE GREEN)	0.0208	0.0546	0.0140	0.673077	12.327416	
2	(ALARM CLOCK BAKELIKE CHOCOLATE)	(ALARM CLOCK BAKELIKE RED)	0.0208	0.0520	0.0140	0.673077	12.943787	
0	(12 PENCILS SMALL TUBE SKULL)	(12 PENCILS SMALL TUBE RED RETROSPOT)	0.0222	0.0276	0.0152	0.684685	24.807416	
3	(ALARM CLOCK BAKELIKE IVORY)	(ALARM CLOCK BAKELIKE GREEN)	0.0268	0.0546	0.0164	0.611940	11.207698	
4	(ALARM CLOCK BAKELIKE ORANGE)	(ALARM CLOCK BAKELIKE GREEN)	0.0278	0.0546	0.0268	0.748201	13.703323	
5	(ALARM CLOCK BAKELIKE RED)	(ALARM CLOCK BAKELIKE GREEN)	0.0520	0.0546	0.0358	0.688462	12.609186	
6	(ALARM CLOCK BAKELIKE GREEN)	(ALARM CLOCK BAKELIKE RED)	0.0546	0.0520	0.0358	0.655678	12.609186	

Hình 4.1 – Trình bày 6 luật kết hợp

3. Hình vẽ cho thấy mối quan hệ giữa support, confidence của luật

```
rules.plot.scatter("support", "confidence", alpha=0.5, marker="*")
plt.xlabel("Support")
plt.ylabel("Confidence")
plt.title("Association Rules")
plt.show()
```

Kết quả thực hiện

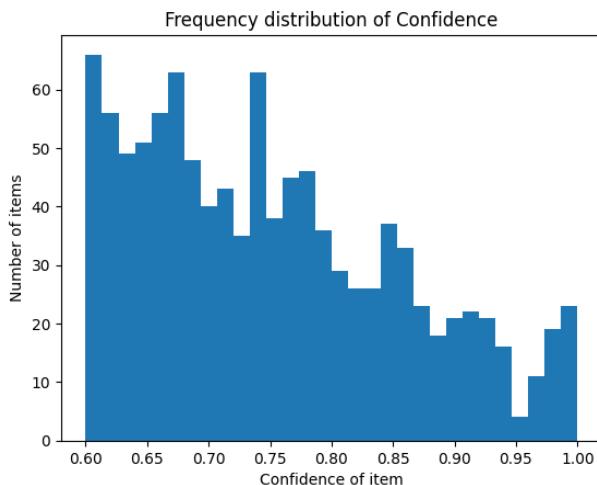


Hình 4.2 - Biểu đồ mối quan hệ giữa support và confidence có trong mỗi kết hợp

4. Hình vẽ cho thấy mối quan hệ giữa số item và confidence của luật

```
rules.hist("confidence", grid=False, bins=30)
plt.xlabel("Confidence of item")
plt.ylabel("Number of items")
plt.title("Frequency distribution of Confidence")
plt.show()
```

Kết quả thực hiện



4.2.3. Bài tập trên lớp

Bài toán 2: Tìm luật kết hợp trong tập dữ liệu phim. Dữ liệu lấy tại <https://grouplens.org/datasets/movielens> với MovieLens 100K dataset

Nhiệm vụ 1: Tìm các luật kết hợp từ tập phổ biến

1. Thực hiện các bước trong bài tập 1 để tìm tập phổ biến có trong dữ liệu đề xuất phim.
2. Xác định danh sách các tập luật ứng tuyển

```
# Now we create the association rules. First, they are candidates until
# the confidence has been tested

candidate_rules = []

for itemset_length, itemset_counts in frequent_itemsets.items():
    for itemset in itemset_counts.keys():
        for conclusion in itemset:
            premise = itemset - set((conclusion,))
            candidate_rules.append((premise, conclusion))

# There are 15285 candidate rules

print("There are {} candidate rules".format(len(candidate_rules)))
```

3. Tính mức độ tin cậy (confidence) từng luật ứng tuyển

```
# Now, we compute the confidence of each of these rules. This is very
# similar to what we did in chapter 1

correct_counts = defaultdict(int)
incorrect_counts = defaultdict(int)

for user, reviews in favorable_reviews_by_users.items():
    for candidate_rule in candidate_rules:
        premise, conclusion = candidate_rule
        if premise.issubset(reviews):
            if conclusion in reviews:
                correct_counts[candidate_rule] += 1
            else:
```

```

        incorrect_counts[candidate_rule] += 1
rule_confidence = {candidate_rule: correct_counts[candidate_rule] /
float(correct_counts[candidate_rule] + incorrect_counts[candidate_rule])
for candidate_rule in candidate_rules}

```

4. Lấy các luật có độ tin cậy > 0.9

```

min_confidence = 0.9
# Filter out the rules with poor confidence
rule_confidence = {rule: confidence for rule, confidence in
rule_confidence.items() if confidence > min_confidence}
print(len(rule_confidence)) #5152 luật

```

5. Liệt kê năm luật kết hợp có độ tin cậy cao nhất

```

from operator import itemgetter
sorted_confidence = sorted(rule_confidence.items(), key=itemgetter(1),
reverse=True)
for index in range(5):
    print("Rule #{0}".format(index + 1))
    (premise, conclusion) = sorted_confidence[index][0]
    print("Rule: If a person recommends {0} they will also recommend
{1}".format(premise, conclusion))
    print(" - Confidence: {:.3f}".format(rule_confidence[(premise,
conclusion)]))
    print("")

```

Kết quả thực hiện

```

Rule #1
Rule: If a person recommends frozenset({98, 181}) they will also
recommend 50
- Confidence: 1.000

Rule #2
Rule: If a person recommends frozenset({172, 79}) they will also
recommend 174
- Confidence: 1.000

Rule #3
Rule: If a person recommends frozenset({258, 172}) they will also
recommend 174
- Confidence: 1.000

```

Hình 4.3 - Các luật kết hợp trình bày dạng văn bản

6. Hiển thị tên phim cụ thể trong các luật kết hợp đã tìm thấy

```

# we can get the movie titles themselves from the dataset
movie_name_filename = os.path.join(data_folder, "u.item")
movie_name_data = pd.read_csv(movie_name_filename, delimiter="|",
header=None, encoding = "mac-roman")

```

```

movie_name_data.columns = ["MovieID", "Title", "Release Date", "Video
                           Release", "IMDB", "<UNK>", "Action", "Adventure",
                           "Animation", "Children's", "Comedy", "Crime",
                           "Documentary", "Drama", "Fantasy", "Film-Noir",
                           "Horror", "Musical", "Mystery", "Romance",
                           "Sci-Fi", "Thriller", "War", "Western"]

def get_movie_name(movie_id):
    title_object = movie_name_data[movie_name_data["MovieID"] == movie_id][["Title"]]
    title = title_object.values[0]
    return title

for index in range(5):
    print("Rule #{0}".format(index + 1))
    (premise, conclusion) = sorted_confidence[index][0]
    premise_names = ", ".join(get_movie_name(idx) for idx in premise)
    conclusion_name = get_movie_name(conclusion)
    print("Rule: If a person recommends {0} they will also recommend
          {1}".format(premise_names, conclusion_name))
    print(" - Confidence: {:.3f}".format(rule_confidence[(premise,
                                                          conclusion)]))
    print("")

```

Kết quả thực hiện

Rule #1
 Rule: If a person recommends Silence of the Lambs, The (1991), Return of the Jedi (1983) they will also recommend Star Wars (1977)
 - Confidence: 1.000

Rule #2
 Rule: If a person recommends Empire Strikes Back, The (1980), Fugitive, The (1993) they will also recommend Raiders of the Lost Ark (1981)
 - Confidence: 1.000

4.2.4. Bài tập về nhà

Tìm hiểu thư viện **apyori** (<https://pypi.org/project/apyori>) để thực hiện tìm luật kết hợp từ tập phổ biến từ dữ liệu mua hàng tại siêu thị. Tham khảo bài hướng dẫn

<https://www.kaggle.com/code/rockystats/apriori-algorithm-or-market-basket-analysis>

D. TÓM TẮT CHƯƠNG

Với 2 phần là xác định tập phổ biến với giải thuật Apriori và xác định luật kết hợp được trình bày trong chương đã phần nào giúp sinh viên có thể tiến hình triển khai tìm tập phổ biến và các luật kết hợp trên một số tập dữ liệu cơ bản.

Với việc thực hành giải quyết các bài tập trên lớp và các bài tập ở nhà giúp sinh viên hiểu rõ hơn các khái niệm lý thuyết đã học và áp dụng tốt vào việc giải quyết các bài toán thực tế.