Chuyên đề chọn lọc trong Hệ thống thông tin

**Neighbor-based Collaborative filtering in the user view**

Thông tin nhóm

Thành viên

|  |  |  |
| --- | --- | --- |
| **MSSV** | **Họ và tên** | **Email** |
| 19120261 | Nguyễn Hữu Khôi | [19120261@student.hcmus.edu.vn](mailto:19120261@student.hcmus.edu.vn) |

Contents

[Các dữ liệu từ hệ thống 2](#_Toc137454199)

Chuẩn bị dữ liệu

# Các dữ liệu từ hệ thống

1. A screenshot of a computer screen

   Description automatically generated with medium confidenceBX-Book-Ratings.csv

Tập dữ liệu BX-Book\_Ratings bao gồm 3 cột: “User-ID”, “ISBN”, “Book-Rating”. Tập dữ liệu này là danh sách các mã khách hàng và đánh giá của họ với mỗi mã sách. Đây là tập dữ liệu chính phục vụ cho mục đích tính toán và dự đoán/tư vấn cho người dùng.

Dữ liệu từ tập này có 1149780 dòng và không có missing data.

1. BX-Books.csv

A screen shot of a computer

Description automatically generated with medium confidenceTập dữ liệu BX-Books có các cột: ISBN, Book-Title, Book-Author, Year-Of-Publication, Publisher, Image-URL-S, Image-URL-M, Image-URL-L. Tập dữ liệu này là danh sách các tựa sách, tác giả của từng tựa sách cũng như năm xuất bản, nhà xuất bản và các hình chụp.

A screen shot of a computer program

Description automatically generated with low confidenceTrong quá trình đọc dữ liệu, có thể xuất hiện lỗi dòng, chúng ta có thể lựa chọn skip các dòng lỗi. Sau khi đọc thành công file, kiểm tra lại số dòng lỗi, cảm thấy tỉ lệ không đáng kể (0.007%), chúng ta có thể hoàn toàn bỏ qua các dòng lỗi.

1. BX-Users.csv

A screenshot of a computer screen

Description automatically generated with medium confidenceTập dữ liệu BX-Users.csv có các cột: User-ID, Location, Age. Tập dữ liệu này lưu thông tin của người dùng. Tập dữ liệu có 278858 dòng, trong đó có một số lượng missing data ở cột Age, nhưng vì thuật toán của chúng ta chưa cần thiết phải sử dụng tuổi của người dùng, missing data sẽ không ảnh hưởng đến kết quả tính toán.

# Phân tích dữ liệu

1. Các dữ liệu cần thiết.

Ở giai đoạn thiết kế thuật toán, dữ liệu chính chúng ta cần là mã người dùng User-ID, mã sách ISBN, và đánh giá của mỗi người dùng với mỗi mã sách họ đã đọc Book-Rating từ tập dữ liệu BX-Book-Ratings.

1. Trực quan hóa.

A graph of green and black hexagons

Description automatically generated with low confidenceĐánh giá sự tương quan giữa điểm trung bình và số lượng sách đã đánh giá của các người dùng cho thấy phần lớn người dùng thường đánh giá trung bình từ 1 đến 2 điểm cho các tựa sách.

A picture containing text, screenshot

Description automatically generatedĐánh giá sự tương quan giữa điểm trung bình và số lần được đánh giá của các mã sách, cho thấy trung bình số điểm các tựa sách nhận được nằm trong khoảng từ 2 đến 4, duy chỉ có một tựa sách có số lượng đánh giá rất cao và nằm trung bình dưới 2 điểm, tuy nhiên, số lượng đánh giá của quyển sách này vẫn không quá đáng kể so với dataset, vậy nên không nhất thiết phải loại bỏ.

A picture containing text, screenshot, display, rectangle

Description automatically generatedBiểu đồ Boxplot có thể hỗ trợ tìm các điểm outlier, theo biểu đồ nhận được, dữ liệu tính toán không có outlier cần loại bỏ.

Tư vấn dựa theo user và độ đo Pearson.

# Công thức theo lý thuyết

Trên đây là công thức tính dự đoán đánh giá của người dùng u1 với sản phẩm i3 dựa theo k người dùng có độ tương đồng cao nhất.

Công thức trên cần tính các giá trị như đánh giá trung bình của người dùng tham gia, đánh giá trung bình của người dùng cần tư vấn, độ tương đồng của người tư vấn và người tham gia.

Để tính độ tương đồng của người tư vấn và người tham gia, ta dựa vào công thức sau:

# Sử dụng python, tạo các hàm tính toán áp dụng 2 công thức trên

1. Tính độ tương đồng giữa 2 user.

*def* calculate\_pcc\_similarity(*user\_ratings\_x*, *user\_ratings\_y*):

    #chọn ra các items đều được đánh giá bởi 2 user

    common\_items = *set*(user\_ratings\_x['ISBN']).intersection(*set*(user\_ratings\_y['ISBN']))

    #nếu danh sách item chung không có tựa sách nào, trả về kết quả 0

    if len(common\_items) == 0:

        return 0.0

    # số điểm đánh giá của 2 user với mỗi item trong danh sách item chung

    ratings\_x = user\_ratings\_x[user\_ratings\_x['ISBN'].isin(common\_items)]['Book-Rating'].to\_numpy()

    ratings\_y = user\_ratings\_y[user\_ratings\_y['ISBN'].isin(common\_items)]['Book-Rating'].to\_numpy()

    book\_ratings\_x = ratings\_x[np.char.isdigit(ratings\_x.astype(*str*))]

    book\_ratings\_y = ratings\_y[np.char.isdigit(ratings\_y.astype(*str*))]

    book\_ratings\_x = book\_ratings\_x.astype(*float*)

    book\_ratings\_y = book\_ratings\_y.astype(*float*)

    #điểm đánh giá trung bình của mỗi user để reduce bias

    avg\_rating\_x = np.mean(book\_ratings\_x)

    avg\_rating\_y = np.mean(book\_ratings\_y)

    #tử số của công thức pearson

    numerator = np.sum((book\_ratings\_x - avg\_rating\_x) \* (book\_ratings\_y - avg\_rating\_y))

    #mẫu số của công thức pearson

    denominator = np.sqrt(np.sum((book\_ratings\_x - avg\_rating\_x) \*\* 2) \* np.sum((book\_ratings\_y - avg\_rating\_y) \*\* 2))

    if denominator == 0:

        return 0.0

    pcc\_similarity = numerator / denominator

    return pcc\_similarity

Hàm calculate\_pcc\_similarity trả về kết quả là độ tương đồng giữa 2 user x và y. Input cho hàm là danh sách đánh giá của 2 user với các mã sách.

Vì độ đo Pearson không phù hợp với các user mới hoặc có quá ít dữ liệu, để thuận tiện hơn cho việc tìm hiểu, chúng ta rút gọn dataset và giữ lại top 100 người dùng có nhiều đánh giá nhất trên hệ thống.

data = pd.read\_csv('top\_ratings.csv')

top\_user\_ids = data['User-ID'].value\_counts().nlargest(100).index

# Filter the data for the top User-IDs

top\_user\_ratings = data[data['User-ID'].isin(top\_user\_ids)]

# Get the top 100 ISBNs with the most book ratings

top\_isbns = top\_user\_ratings['ISBN'].value\_counts().nlargest(100).index

# Filter the data for the top ISBNs

top\_isbn\_ratings = top\_user\_ratings[top\_user\_ratings['ISBN'].isin(top\_isbns)]

# Save the filtered data to a new DataFrame

top\_ratings = top\_isbn\_ratings[['User-ID', 'ISBN', 'Book-Rating']]

# Save the new DataFrame to a CSV file

top\_ratings.to\_csv('top\_ratings.csv', *index*=False)

print("Top 100 User-IDs with the most book ratings data saved to 'top\_ratings.csv'.")

Từ đây, ta sẽ làm việc với dataset mới là top\_ratings.csv.

Vì dataset lớn, việc tính toán độ tương đồng mỗi khi cần tư vấn sẽ rất mất thời gian, chúng ta sẽ tính toán độ tương đồng của toàn bộ người dùng trong dataset, lưu lại thành một file và sẽ cập nhật trong tương lai vào một thời điểm nhất định.

import pandas as pd

import numpy as np

# Tính độ tương đồng giữa mỗi cặp người dùng

user\_ids = data['User-ID'].unique()

num\_users = len(user\_ids)

similarity\_matrix = np.zeros((num\_users, num\_users))

for i in range(num\_users):

    user\_ratings\_x = data[data['User-ID'] == user\_ids[i]]

    for j in range(i + 1, num\_users):

        user\_ratings\_y = data[data['User-ID'] == user\_ids[j]]

        similarity = calculate\_pcc\_similarity(user\_ratings\_x, user\_ratings\_y)

        similarity\_matrix[i, j] = similarity

        similarity\_matrix[j, i] = similarity

        # Hàm kiểm tra tiến trình

        progress = (i \* (num\_users - 1) + j - i - 1) / ((num\_users - 1) \* num\_users / 2) \* 100

        print(*f*"Progress: {progress*:.2f*}%")

# Lưu vào một file

np.savetxt('similarity\_matrix.csv', similarity\_matrix, *delimiter*=',')

print("Similarity matrix calculation completed and saved to 'similarity\_matrix.csv'.")

Từ đó, ta có tập dữ liệu độ tương đồng người dùng là similarity\_matrix.csv.

1. Xây dựng hàm tư vấn dựa trên độ tương đồng và dự đoán đánh giá.

*def* generate\_user\_based\_recommendations(*user\_id*, *data*, *similarity\_matrix*, *k*, *top\_n*):

   # Đánh giá của người dùng

    user\_ratings = data[data['User-ID'] == *int*(user\_id)]

# Lấy độ đo tương đồng của các người dùng khác đối với người dùng cần tư vấn

    similarity\_scores = similarity\_matrix[user\_id]

# Lưu độ đo thành dataframe

    sim\_df = similarity\_scores.reset\_index()

    sim\_df.columns = ['other\_user\_id', 'similarity\_score']

# Sắp xếp giảm dần theo độ đo

    sorted\_sim\_df = sim\_df.sort\_values(*by*='similarity\_score', *ascending*=False)

 # chọn ra k người dùng có độ tương đồng cao nhất

    top\_sim\_df = sorted\_sim\_df.head(k)

  # Tính dự đoán đánh giá

    predicted\_ratings = {}

    for index, row in top\_sim\_df.iterrows():

        other\_user\_id = row['other\_user\_id']

        similarity = row['similarity\_score']

        # Với mỗi người dùng tham gia, lọc ra các đánh giá của người dùng đó

        other\_user\_ratings = data[data['User-ID'] == other\_user\_id]

        for \_, book\_row in other\_user\_ratings.iterrows():

            item\_id = book\_row['ISBN']

            rating = book\_row['Book-Rating']

            # Chỉ quan tâm đến những tựa sách chưa được đánh giá bởi người dùng cần tư vấn

            if item\_id not in user\_ratings['ISBN'].values:

                if item\_id not in predicted\_ratings:

                    predicted\_ratings[item\_id] = {

                        'numerator': 0, 'denominator': 0}

                # Tính tử số và mẫu số của công thức

                predicted\_ratings[item\_id]['numerator'] += similarity \* \

                    (rating - other\_user\_ratings['Book-Rating'].mean())

                predicted\_ratings[item\_id]['denominator'] += similarity

    # Tính dự đoán đánh giá

    predicted\_ratings\_values = {}

    for item\_id, values in predicted\_ratings.items():

        predicted\_ratings\_values[item\_id] = user\_ratings['Book-Rating'].mean(

        ) + values['numerator'] / values['denominator']

    # Sắp xếp dự đoán theo thứ tự giảm dần

    sorted\_predicted\_ratings = sorted(

        predicted\_ratings\_values.items(), *key*=*lambda* *x*: x[1], *reverse*=True)

    # chọn ra top\_n các tựa sách được dự đoán đánh giá cao nhất

    top\_predicted\_ratings = sorted\_predicted\_ratings[:top\_n]

    # Tạo danh sách tư vấn

    recommendations = []

    for item\_id, predicted\_rating in top\_predicted\_ratings:

        recommendations.append((item\_id, predicted\_rating))

    return recommendations, predicted\_ratings\_values

Dựa vào bảng độ tương đồng đã tạo ở bước trước, ta tiến hành tính dự đoán của người dùng với các tựa sách mà người dùng tham gia đã đánh giá, sắp xếp giảm dần và chọn ra top\_n các tựa sách được dự đoán đánh giá cao nhất.

# Thực nghiệm và kiểm thử

1. Thử nghiệm tư vấn cho một người dùng.

similarity\_matrix = pd.read\_csv('similarity\_matrix\_2.csv', *index\_col*=0)

data = pd.read\_csv('top\_ratings.csv')

user\_id="6251"

k=5

top\_n = 10

recommendations, predicted\_ratings = generate\_user\_based\_recommendations(user\_id, data, similarity\_matrix, k, top\_n)

# Print the recommendations

print(*f*"Recommendations for User {user\_id}:")

for item\_id, predicted\_rating in recommendations:

    print(*f*"Item ID: {item\_id}, Predicted Rating: {predicted\_rating}")

Dự đoán và tư vấn cho người dùng mã số 6251 top 10 tựa sách với 5 người dùng có độ tương đồng cao nhất.

Kết quả dự đoán:

A screenshot of a computer screen

Description automatically generated with medium confidence

1. Kiểm tra độ chính xác của dự đoán bằng RMSE.

from sklearn.metrics import mean\_squared\_error

import numpy as np

shuffled\_df = data.sample(*frac*=1, *random\_state*=12)

# Chia dataset thành 2 phần

train\_df = shuffled\_df.sample(*frac*=0.8, *random\_state*=42)

test\_df = shuffled\_df.drop(train\_df.index)

k = 5

top\_n = len(train\_df['ISBN'].unique())

# Danh sách người dùng trong tập train

train\_user\_ids = train\_df['User-ID'].unique()

# Danh sách người dùng trong tập test

test\_user\_ids = test\_df['User-ID'].unique()

rmse\_values = []

actual\_ratings\_list = []

predicted\_ratings\_list = []

for user\_id in test\_user\_ids:

    if user\_id in train\_user\_ids:

        # Tính toán dự đoán

        recommendations, predicted\_ratings\_values = generate\_user\_based\_recommendations(*str*(user\_id), train\_df, similarity\_matrix, k, top\_n)

        # Dự đoán thực tế trong tập test

        actual\_ratings = test\_df[test\_df['User-ID'] == user\_id][['ISBN', 'Book-Rating']].values

        # Lấy danh sách mã sách trong tập test

        item\_ids = actual\_ratings[:, 0]

        # Lấy dự đoán đối với mỗi mã sách trong tập test sau khi tính toán

        predicted\_ratings = [predicted\_ratings\_values.get(item\_id, 0) for item\_id in item\_ids]

        # Bỏ qua missing value bằng cách thay bằng số 0

        actual\_ratings = np.nan\_to\_num(actual\_ratings[:, 1])

        predicted\_ratings = np.nan\_to\_num(predicted\_ratings)

        # Tính RMSE cho mỗi user

        rmse = np.sqrt(mean\_squared\_error(actual\_ratings, predicted\_ratings))

        rmse\_values.append(rmse)

        # Lưu lại đánh giá của tập test và tập dự đoán

        actual\_ratings\_list.extend(actual\_ratings)

        predicted\_ratings\_list.extend(predicted\_ratings)

# Tính RMSE trung bình

average\_rmse = np.mean(rmse\_values)

print(average\_rmse)

Kết quả RMSE cho thấy độ lệch trung bình của dự đoán là xấp xỉ 2.55, với khoảng đánh giá từ 0 đến 10, có thể cho thấy kết quả của dự đoán đưa ra chưa thực sự tốt.

A graph with red and blue dots

Description automatically generated with low confidenceĐể thể hiện rõ hơn về độ chính xác của thuật toán, ta có thể sử dụng biểu đồ với chấm xanh là đánh giá thực tế, chấm đỏ là dự đoán, ta thấy các dự đoán còn có thể rơi vào điểm âm – đây là số điểm không có trong hệ thống.