**BỘ MÔN HỆ THỐNG THÔNG TIN – KHOA CÔNG NGHỆ THÔNG TIN**

**ĐẠI HỌC KHOA HỌC TỰ NHIÊN THÀNH PHỐ HỒ CHÍ MINH, ĐẠI HỌC QUỐC GIA TP HCM**

Sinh viên thực hiện: Nhóm 2122CDCL\_04

GV phụ trách: Hồ Thị Hoàng Vy

Đồ án  - CHUYÊN ĐỀ CHỌN LỌC TRONG HỆ THỐNG THÔNG TIN

HỌC KỲ II – NĂM HỌC 2021-2022

**môn chuyên đề chọn lọc trong hệ thống thông tin**

**BẢNG THÔNG TIN CHI TIẾT NHÓM**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Mã nhóm:** | 2122CDCL\_04 | | | |
| **Tên nhóm:** | 2122CDCL\_04 | | | |
| **Số lượng:** | **4** | | | |
| **MSSV** | **Họ tên** | **Email** | **Điện thoại** | **Hình ảnh** |
| 1712486 | Võ Quốc Hưng | 1712486@student.hcmus.edu.vn |  |  |
| 18120418 | Phạm Minh Khoa | 18120418@student.hcmus.edu.vn |  |  |
| 1712850 | Trần Trung | 1712850@student.hcmus.edu.vn |  |  |
| 1712553 | Bùi Tấn Lân | 1712553@student.hcmus.edu.vn |  |  |

**BẢNG PHÂN CÔNG & ĐÁNH GIÁ HOÀN THÀNH CÔNG VIỆC**

|  |  |  |  |
| --- | --- | --- | --- |
| **Bảng phân công & đánh giá hoàn thành công việc** | | | |
| **Công việc thực hiện** | **Người thực hiện** | **Mức độ hoàn thành** | **Đánh giá của nhóm** |
| Lên ý tưởng thuật toán, code, đánh giá, nhận xét | Võ Quốc Hưng | 70% | 10/10 |
| Lên ý tưởng thuật toán, code, lọc dữ liệu, tìm hiểu thông tin dữ liệu | Phạm Minh Khoa | 100% | 10/10 |
| Lên ý tưởng thuật toán, code, đánh giá, nhận xét | Trần Trung | 70% | 10/10 |
| Lên ý tưởng thuật toán, code, thống kê | Bùi Tấn Lân | 100% | 10/10 |

Mục Lục

[**A.** **YÊU CẦU CỦA ĐỒ ÁN** 2](#_Toc102242150)

[1. Yêu cầu đồ án 2](#_Toc102242151)

[2. Yêu cầu nộp 2](#_Toc102242152)

[**B.** **KẾT QUẢ** 3](#_Toc102242153)

[1. Giải thích dataset, ý nghĩa thuộc tính 3](#_Toc102242154)

[2. Ý tưởng thực hiện thuật toán (mã giả) 7](#_Toc102242155)

[2.1. Tạo ma trận tương đồng item-item (giai đoạn offline) 7](#_Toc102242156)

[2.2. Dự đoán rating (giai đoạn online) 8](#_Toc102242157)

[3. Kết quả thực hiện 10](#_Toc102242158)

[4. Đánh giá 19](#_Toc102242159)

[5. Link tham khảo 19](#_Toc102242160)

# **YÊU CẦU CỦA ĐỒ ÁN**

1. Yêu cầu đồ án
2. Dataset: <https://grouplens.org/datasets/movielens/100k/>
3. Hãy cài đặt hệ tư vấn phim theo kỹ thuật items to items Collaborative Filtering  
   (memory based) sử dụng ngôn ngữ python.

* Mô tả dataset cách chi tiết.
* Mô tả bài toán item to item CF với movilen dataset (mô tả ý tưởng thực hiện)
* Đọc và chuyển dữ liệu vào ma trận
* Quan sát mô tả dữ liệu
* Biến đổi, xử lý dữ liệu
* Giai đoạn offline:
  + tính toán độ tương đồng sử dụng KNN
  + tính toán độ tương đồng sử dụng Jaccar
* Giai đoạn online:
  + Phát sinh các gợi ý
* Đánh giá thuật toán:
  + Lựa chọn độ đo đánh giá độ chính xác của dự đoán

1. Yêu cầu nộp

* .doc: báo cáo giải thích về dataset, ý nghĩa thuộc tính, ý tưởng thực hiện thuật  
  toán (mã giả), kết quả thực hiện, đánh giá…
* Video: quay lại quá trình trao đổi và thực hiện của mỗi thành viên có mở camera
* Source code: .pybn và link google colab
* Làm bài trên google colab
* Lần 1:
  + Nộp source + báo cáo
  + Source: .iynb (có hiển thị kết quả thực thi của mỗi đoạn lệnh)
  + File data trước và sau khi xử lý làm sạch dữ liệu
  + Báo cáo: giải thích, mô tả dataset, các bước làm sạch, biến đổi dữ liệu, …
  + Tính toán độ tương đồng sử dụng Jaccard
* Lần 2: nộp hoàn chỉnh
  + Source + báo cáo
  + Source: gồm đầy đủ các bước cả giai đoạn offline + online, file dữ liệu
  + Báo cáo: phân tích, cài đặt, đánh giá
  + Video giải thích bài làm.

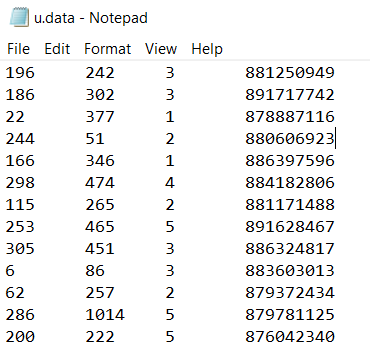
# **Kết quả**

## Giải thích dataset, ý nghĩa thuộc tính

Sử dụng thư viện pandas và file README để tìm hiểu, kiểm tra, lọc data.

Sử dụng hàm read\_csv() để đọc dữ liệu.

* **ml-100k.zip:** chứa tập những file dữ liệu (u.data, u1.base, u1.test…), file chứa script phân data và file thông tin mô tả.
* **u.data:**

****

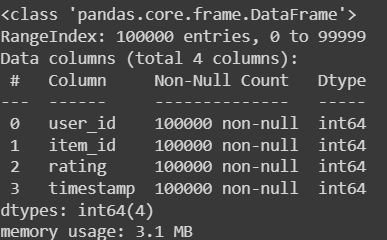
* + Theo mô tả từ file README, file gồm có 100.000 rating (đánh giá từ 1  5) của 943 người dùng về 1682 video. Mỗi user đã đánh giá ít nhất 20 phim. Dữ liệu được sắp xếp ngẫu nhiên và mỗi dòng tuân theo cấu trúc sau: **user id | item id | rating | timestamp**. Các cột cách nhau bởi dấu tab.

***Kiểm tra dữ liệu:***

* + File chỉ chứa dữ liệu, dòng đầu tiên không phải tên của các cột nên sử dụng hàm read\_csv() với tham số names là mảng các tên cột để khỏi nhầm dòng đầu tên là tên các cột. Khi có đối số names, thì tham số chuyển từ mặc định header=0 sang header=non, cho phép lấy mảng names làm tên các cột.
  + Lệnh đọc data từ file u.data trên google drive, sử dụng google colab để thực thi.

pd.read\_csv("/content/drive/MyDrive/Colab Notebooks/moviesLenData/u.data", sep='\t', names=['user\_id','item\_id','rating','timestamp'])

* + Dùng hàm info() để kiểm tra dữ liệu, kết quả:

****

* + - User id, Item id, Rating, timestamp đều là số nguyên.
  + Các **timestamp** là Unix giây kể từ 1/1/1970 UTC, lúc 1/1/1970 là 0 và thời gian sau đó cộng thêm số giây tăng thêm.
  + Kiểm tra giá trị của rating: max=5, min=1

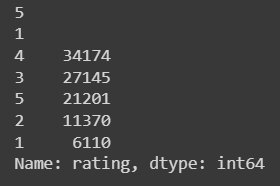
Sử dụng các hàm:

print(df.rating.max())

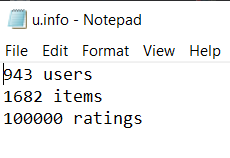
print(df.rating.min())

print(df.rating.value\_counts())

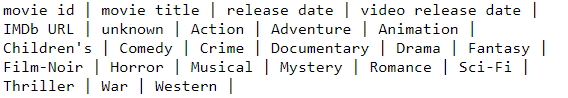
**Kết quả:**

****

* + Tưng tự cho số user và video đều khớp với mô tả trong file README.
  + **Vậy không có dữ liệu sai, null, thiếu sót trong file u.data => không cần lọc dữ liệu.**
* **u.info:** gồm số lượng user, phim, rating trong file **u.data**

****

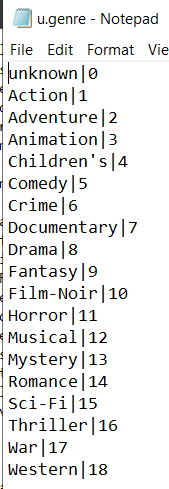
* **u.item:** thông tin về các phim, 19 cột sau cùng là thể loại (từ unknown trở về sau). Giá trị 1 là phim thuộc thể loại, ngược lại, không thuộc là 0. Một phim có thể thuộc nhiều thể loại. movie id dược sử dụng trong file u.data. Cấu trúc như sau:

****

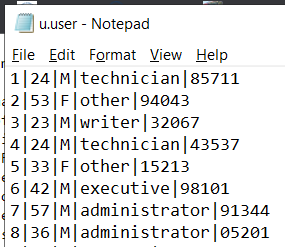
Và dữ liệu được lưu trong file như sau, ngăn cách bằng dấu gạch đứng |, trước cột URL có hai gạch đứng ||:



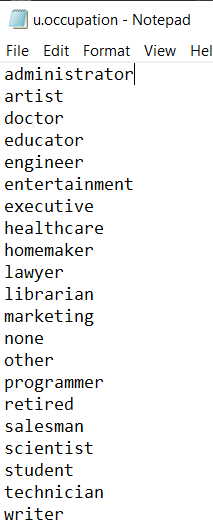
* + Lệnh đọc file u.item: với đối số đầu tiên là đường dẫn tới file u.item, đối số thứ 2 là ký tự ngăn cách (sep), header=None chỉ file dữ liệu không có tên cột cho dữ liệu, tránh việc lấy dòng đầu làm tên cột, names là danh sách tên cột đưa vào, encoding để chỉ loại dữ liệu ký tự bên trong file u.item.
* movieItem = pd.read\_csv('/content/drive/MyDrive/Colab Notebooks/moviesLenData/u.item', sep='|',header=None,names=colItem,encoding='latin-1')
* **u.genre:** danh sách các thể loại, gồm có 19 thể loại. Cột 1 là tên, cột 2 là id, 2 cột được ngăn cách bởi dấu gạch đứng |.

****

* **u.user:** Thông tin nhân khẩu về những user, mỗi cột được ngăn cách bởi dấu gạch đứng | và có cấu trúc như sau: user id | age | gender | occupation | zip code. User id được sử dụng trong u.data.

****

* **u.occupation:** danh cách các nghề nghiệp, gồm 21 nghề.

****

* **u1.base  u5.base và u1.test  u5.test:** .base là dữ liệu train, .test là dữ liệu test. Dữ liệu được lấy ra từ u.data chia 80% cho u1.base và 20% cho u1.test. u1  u5 được tạo khi chạy file thực thi mku.sh.
* **ua.base, ua.test, ub.base, ub.test:** .base là tập dữ liệu train, .test là tập dữ liệu test với mỗi user lấy 10 đánh giá của user (tức lấy 10 phim được đánh giá bởi mỗi người dùng). File ua.test bị lỗi mở bằng notepad, lên toàn tiếng trung, nhưng mở bằng visual code thì được. Dữ liệu được lấy từ file u.data khi chạy file mku.sh.
* **allbut.pl:**  script tạo ra các tập training và test trong đó tất cả trừ n rating người dùng nằm trong dữ liệu training.

## Ý tưởng thực hiện thuật toán (mã giả)

### Tạo ma trận tương đồng item-item (giai đoạn offline)

1. Lấy dữ liệu từ nguồn
2. Chuyển thành ma trận
3. Nén ma trận thưa thớt thành ma trận dày
4. Tạo ma trận movie tương đồng sử dụng thuật toán Jarcard

Jaccard:

* tập những user mà đã đánh giá cả 2 sản phẩm i và j
* tập hợp những user mà đã đánh giá i hoặc j

1. Mã giả cho thuật toán tính độ tương đồng giữa các movie (đầu ra là danh sách các movie id tăng dần theo độ tương đồng)

 createSimilarMoviesMatrix():

   amountMovie = lấy số lượng movie trong hệ thống

amountUser = lấy số lượng user trong hệ thống

   Khởi tạo tập chứa các danh sách movie tương đồng itemSimilarMatrix = []

   Khởi tạo biến đếm i,j=0,0

   while i < amountMovie:

     userRatedI = danh sách user đã đánh giá movie i

     Khởi tạo danh sách độ tương đồng đối với movie i iScoreList = []

     while j < amountMovie:

       userRatedJ = danh sách user đã đánh giá movie j

       union = Số user đã đánh giá movie i hoặc j

       intersection = Số user đã đánh giá cả movie i và j

       score = intersection / union

       iScoreList.append(score)

       Tăng j lên 1 đơn vị

     Khởi tạo lại j = 0 cho một vòng lặp mới

Sắp xếp danh sách (iScoreList) theo độ tương đồng (score) tăng dần mà không thay đổi id của danh sách

     itemSimilarMatrix.Thêm danh sách id user đã sắp xếp()

     Xóa dữ liệu iScoreList

     Tăng I lên 1 đơn vị

   return itemSimilarMatrix

* Chú ý là thứ tự id đã có từ ma trận rating nên tương đồng với biến đếm j. Khi thêm vào iScoreList sẽ có thứ tự id tương ứng.

### Dự đoán rating (giai đoạn online)

1. Dựa vào công thức tính rating sau:
2. Hình thành các hàm tính toán dự đoán rating

* Có một hàm chính tính dự đoán rating, hàm này sẽ gọi các hàm khác như: hàm tính bias (), hàm tính tử (), hàm tính mẫu (), hàm tìm danh sách phim tương đồng. Hàm chính sẽ có 3 tham số: id của active user, id của movie cần dự đoán, và số phim tương đồng nhất cần lấy. Hàm sẽ trả về rating dự đoán.
* Hàm lấy danh sách phim tương đồng: hàm truyền vào 2 tham số: id phim cần dự đoán rating, số phim tương đồng nhất cần lấy. Trả về dánh sách id các phim tương đồng nhất. Lấy danh sách dựa trên ma trận đã tính ở giai đoạn offline (similarRankingMatrix)
* Hàm tính bias: tính bias của một user, hàm gồm 1 tham số là id user cần tính bias. Trả về bias theo công thức trung bình cộng, có nghĩa là tính trung bình những rating mà user đã đánh giá trước đó. Dựa trên ma trận dày movieRatingDenseMatrix.
* Hàm tính similarity giữa 2 phim: phục vụ cho tính toán cho những hàm tính tử số và mẫu số. Hàm có 2 tham số là 2 id của phim. Dựa vào công thức Jarcard item-based để tính. Lấy dữ liệu từ ma trận movieRatingDenseMatrix.
* Hàm tính tử số: Hàm có 4 tham số là id user, id phim, danh sách phim tương đồng, bias của active user. Hàm sẽ lặp lại số lần theo danh sách phim tương đồng.
* Hàm tình mẫu số: hàm có 2 tham số là id phim cần dự đoán rating và danh sách id phim tương đồng. Hàm sẽ tính tổng các độ tương đồng.

1. Mã giả

* Hàm chính, dự đoán rating:

function predictRatingMovie(input User Id, input Movie Id, số phim tương đồng=10):

  movieSims=lấy danh sách phim tương đồng nhất id phim đưa vào (inputMovieId, k)

  bias=tính bias của user(input User Id)

  num=tính tử số(inputUserId, inputMovieId, movieSims, bias)

  dentính mẫu số(inputMovieId, movieSims)

  print('bias:',bias,'num:', num,'den:', den)

  return (bias + num/den)

* Hàm Lấy danh sách phim tương đồng

function getSimilarMovies(inputMovieId, n=10):

   return similarRankingMatrix[inputMovieId-1][-n-1:-1]

* Hàm tính bias

function calBiasUser(inputUserId):

  k=lấy tất cả rating mà user đã đánh giá trước đây từ ma trận dày

  count=0

  mean=0

  for i in k:

    if loại những rating = 0, tức chưa rating

      mean=mean+i

      count+=1

  mean=mean/count

  return mean[0]

* Hàm tính độ tương đồng giữa 2 phim

function calSimMovie(movieId1, movieId2):

  userRated1 = lấy danh sách user đã rating cho phim có id 1 từ ma trận dày

  userRated2 = lấy danh sách user đã rating cho phim có id 1 từ ma trận dày movieRatingDenseMatrix

  union = tính số user đã rating cho hai phim

  intersection = tính số user đã rating cho cả 2 phim

  return intersection / union

* Hàm tính tử số

function calNumerator(idUser, idMovie, movies, bias):

  num=0

  for i in movies:

    rating= lấy rating của user đã đánh giá cho phim I từ ma trận movieRatingMatrix

    sim= gọi hàm tính độ tương đồng giữa 2 phim, calSimMovie(idMovie,i)

    num+=(sim \* (rating-bias))

  return num

* Hàm tính mẫu số

function calDenominator(idMovie, movies):

  den=0

  for i in movies:

    sim= gọi hàm tính độ tương đồng giữa hai phim calSimMovie(idMovie,i)

    den+=sim

  return den

## Kết quả thực hiện

# **Lấy data từ website grouplens và giải nén, không cần kết nối drive**

**- Giải nén file**

[ ]

DATASET\_LINK='http://files.grouplens.org/datasets/movielens/ml-100k.zip'  
!wget -nc http://files.grouplens.org/datasets/movielens/ml-100k.zip  
!unzip -n ml-100k.zip

--2022-04-30 13:31:09-- <http://files.grouplens.org/datasets/movielens/ml-100k.zip>

Resolving files.grouplens.org (files.grouplens.org)... 128.101.65.152

Connecting to files.grouplens.org (files.grouplens.org)|128.101.65.152|:80... connected.

HTTP request sent, awaiting response... 200 OK

Length: 4924029 (4.7M) [application/zip]

Saving to: ‘ml-100k.zip’

ml-100k.zip 100%[===================>] 4.70M 28.5MB/s in 0.2s

2022-04-30 13:31:10 (28.5 MB/s) - ‘ml-100k.zip’ saved [4924029/4924029]

Archive: ml-100k.zip

creating: ml-100k/

inflating: ml-100k/allbut.pl

inflating: ml-100k/mku.sh

inflating: ml-100k/README

inflating: ml-100k/u.data

inflating: ml-100k/u.genre

inflating: ml-100k/u.info

inflating: ml-100k/u.item

inflating: ml-100k/u.occupation

inflating: ml-100k/u.user

inflating: ml-100k/u1.base

inflating: ml-100k/u1.test

inflating: ml-100k/u2.base

inflating: ml-100k/u2.test

inflating: ml-100k/u3.base

inflating: ml-100k/u3.test

inflating: ml-100k/u4.base

inflating: ml-100k/u4.test

inflating: ml-100k/u5.base

inflating: ml-100k/u5.test

inflating: ml-100k/ua.base

inflating: ml-100k/ua.test

inflating: ml-100k/ub.base

inflating: ml-100k/ub.test

[ ]

#adding library  
import numpy as np  
  
import pandas as pd

**- Xem file u.data**

[ ]

column\_names1 = ['user id','movie id','rating','timestamp']  
dataset = pd.read\_csv('ml-100k/u.data', sep='\t',header=None,names=column\_names1)  
dataset.head()

# **Check data infomation, datatype, file formation, seperate character**

[ ]

#adding library  
import numpy as np  
  
import pandas as pd

[ ]

#add data from drive  
#from google.colab import drive  
#drive.mount('/content/drive')

Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force\_remount=True).

[ ]

#get data into data frame, then check data (u.data)  
df  = pd.read\_csv("ml-100k/u.data", sep='\t', names=['user\_id','item\_id','rating','timestamp'])  
df.info()

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 100000 entries, 0 to 99999

Data columns (total 4 columns):

# Column Non-Null Count Dtype

--- ------ -------------- -----

0 user\_id 100000 non-null int64

1 item\_id 100000 non-null int64

2 rating 100000 non-null int64

3 timestamp 100000 non-null int64

dtypes: int64(4)

memory usage: 3.1 MB

[ ]

#check min, max rating and count how many the distincts value  
print("rating:")  
print(df.rating.max())  
print(df.rating.min())  
print(df.rating.value\_counts())  
  
#check how many the distincts value of users id  
print("user:")  
print(df.user\_id.value\_counts())  
#check how many the distincts value of video  
print("video:")  
print(df.item\_id.value\_counts())

rating:

5

1

4 34174

3 27145

5 21201

2 11370

1 6110

Name: rating, dtype: int64

user:

405 737

655 685

13 636

450 540

276 518

...

441 20

36 20

812 20

895 20

93 20

Name: user\_id, Length: 943, dtype: int64

video:

50 583

258 509

100 508

181 507

294 485

...

852 1

1505 1

1653 1

1452 1

1641 1

Name: item\_id, Length: 1682, dtype: int64

[ ]

df['rating'].describe()

count 100000.000000

mean 3.529860

std 1.125674

min 1.000000

25% 3.000000

50% 4.000000

75% 4.000000

max 5.000000

Name: rating, dtype: float64

[ ]

[ ]

plt.hist(df['rating'],log=True);

plt.plot();

plt.show();

# **Offline phase - Caculate the similarity**

**- Thêm thư viện - kết nối drive để lấy dữ liệu**

[ ]

#Thêm thư viện, thêm đường dẫn lấy dữ liệu  
#from google.colab import drive  
#drive.mount('/content/drive')  
DATASET\_LINK='http://files.grouplens.org/datasets/movielens/ml-100k.zip'  
!wget -nc http://files.grouplens.org/datasets/movielens/ml-100k.zip  
!unzip -n ml-100k.zip  
import numpy as np  
  
import pandas as pd

File ‘ml-100k.zip’ already there; not retrieving.

Archive: ml-100k.zip

**- Lấy dữ liệu**

[ ]

## Lấy dữ liệu từ file u.data và lọc dữ liệu (xóa duplicate, null, NaN)  
movieDataRating  = pd.read\_csv("ml-100k/u.data",   
                               sep='\t', names=['user\_id','movie\_id','rating','timestamp'])  
movieDataRating.dropna(inplace=True)  
movieDataRating.drop\_duplicates(inplace=True)  
## Lấy dữ liệu gồm title, id phim từ u.item và lọc  
col = 'movie id | movie title | release date | video release date | IMDb URL | unknown | Action '  
col+= '| Adventure | Animation | Children | Comedy | Crime | Documentary | Drama | Fantasy | Film-Noir '  
col+= '| Horror | Musical | Mystery | Romance | Sci-Fi | Thriller | War | Western'  
colItem = col.split(' | ')  
movieItem = pd.read\_csv('ml-100k/u.item',   
                        sep='|',header=None,names=colItem,encoding='latin-1')  
movieItem = movieItem[['movie id','movie title']]  
movieItem.dropna(inplace=True)  
movieItem.drop\_duplicates(inplace=True)

**- Chuyển dataframe thành ma trận user-item rating. Cột là các user id, dòng là các movie id**

[ ]

## Chuyển dataframe thành ma trận user-item rating. Cột là các user id, dòng là các movie id  
## Dùng hàm pivot chuyển dataframe thành bảng. Hàm fillna() dùng để thay các giá trị NaN, Null bằng 0  
movieRatingMatrix = movieDataRating.pivot(  
    index='movie\_id',  
     columns='user\_id',  
      values='rating').fillna(0)  
movieRatingMatrix

| **user\_id** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** | **...** | **934** | **935** | **936** | **937** | **938** | **939** | **940** | **941** | **942** | **943** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **movie\_id** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **1** | 5.0 | 4.0 | 0.0 | 0.0 | 4.0 | 4.0 | 0.0 | 0.0 | 0.0 | 4.0 | ... | 2.0 | 3.0 | 4.0 | 0.0 | 4.0 | 0.0 | 0.0 | 5.0 | 0.0 | 0.0 |
| **2** | 3.0 | 0.0 | 0.0 | 0.0 | 3.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | ... | 4.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 5.0 |
| **3** | 4.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | ... | 0.0 | 0.0 | 4.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| **4** | 3.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 5.0 | 0.0 | 0.0 | 4.0 | ... | 5.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 2.0 | 0.0 | 0.0 | 0.0 |
| **5** | 3.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | ... | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **1678** | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | ... | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| **1679** | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | ... | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| **1680** | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | ... | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| **1681** | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | ... | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| **1682** | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | ... | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |

1682 rows × 943 columns

**- Chuyển ma trận thưa (nhiều giá trị 0) về ma trận dày đặt phục vụ cho việc dễ tính toán, giảm bộ nhớ lưu trữ.**

[ ]

## Chuyển ma trận thưa (nhiều giá trị 0) về ma trận dày đặt phục vụ cho việc dễ tính toán, giảm bộ nhớ lưu trữ.  
from scipy.sparse import csr\_matrix

[ ]

movieRatingDenseMatrix = csr\_matrix(movieRatingMatrix.values)  
movieRatingDenseMatrix

<1682x943 sparse matrix of type '<class 'numpy.float64'>'

with 100000 stored elements in Compressed Sparse Row format>

**- Tạo ma trận tương đồng. Loại tính toán tương đồng cho chính movie ở giai đoạn online (trong hàm lấy top k movie tương đồng nhất). Movie id sẽ giảm xuống 1 đơn vị!**

[ ]

#Tạo hàm xây dựng ma trận tương đồng  
def createSimilarMoviesMatrix():  
   amountMovie,amountUser=movieRatingMatrix.shape  
   itemSimilarMatrix = []  
   i,j=0,0  
   while i < amountMovie:  
     userRatedI = movieRatingDenseMatrix.getrow(i).indices.tolist()  
     iScoreList = []  
     while j < amountMovie:  
       userRatedJ = movieRatingDenseMatrix.getrow(j).indices.tolist()  
       union = len(set(userRatedI) | set(userRatedJ))  
       intersection = len(set.intersection(set(userRatedI), set(userRatedJ)))  
       score = intersection / union  
       iScoreList.append(score)  
       j+=1  
     j=0  
     itemSimilarMatrix.append(np.argsort(iScoreList))  
     iScoreList.clear()  
     i+=1  
   return itemSimilarMatrix

**- Ma trận gồm danh sách các movie tương đồng, sắp xếp theo thứ tự độ tương đồng tăng dần** (train gần 4 phút)

[ ]

## Ma trận gồm danh sách các movie tương đồng, sắp xếp theo thứ tự độ tương đồng tăng dần  
similarRankingMatrix=createSimilarMoviesMatrix()

**- Kiểm thử:**

[ ]

print('Số dòng ma trận movie tương đồng:',len(similarRankingMatrix))  
movie1=similarRankingMatrix[0]  
m=len(movie1)  
print('Số cột ma trận movie tương đồng:',m)  
print('Danh sách các movie id có độ tương đồng với movie 1 tăng dần:',movie1)  
print('Top 5 movie tương đồng nhất với movie 1:',movie1[-5:])

Số dòng ma trận movie tương đồng: 1682

Số cột ma trận movie tương đồng: 1682

Danh sách các movie id có độ tương đồng với movie 1 tăng dần: [1568 1586 1587 ... 120 49 0]

Top 5 movie tương đồng nhất với movie 1: [116 180 120 49 0]

# **Online phase - Recommend films**

# **Công thức Jarcard: tính độ tương đồng của 2 movie**

**- Tạo hàm tính toán lấy danh sách movie tương đồng:**

[ ]

## Tạo hàm tính toán lấy danh sách movie tương đồng  
def getSimilarMovies(inputMovieId, n=10):  
   return similarRankingMatrix[inputMovieId-1][-n-1:-1]

[ ]

movieTopNSimilar=getSimilarMovies(1, 5)  
print('Top 5 movie tương đồng nhất với movie 1:',movieTopNSimilar)

Top 5 movie tương đồng nhất với movie 1: [ 99 116 180 120 49]

**- Tính bias của một user**

[ ]

# Tính bias của một user =  trung bình tất cả các rating đã đánh  
def calBiasUser(inputUserId):  
  k=movieRatingDenseMatrix.getcol(inputUserId-1).toarray()  
  count=0  
  mean=0  
  for i in k:  
    if i[0] != 0:  
      mean=mean+i[0]  
      count+=1  
  mean=mean/count  
  return mean  
  
print('Bias của user 5:', calBiasUser(1))

Bias của user 5: 3.610294117647059

**- Hàm tính sim**

[ ]

#Hàm tính độ tương đồng của 2 phim, đầu vào là 2 id, đầu ra là độ tương đồng (float)  
def calSimMovie(movieId1, movieId2):  
  userRated1 = movieRatingDenseMatrix.getrow(movieId1-1).indices.tolist()  
  userRated2 = movieRatingDenseMatrix.getrow(movieId2-1).indices.tolist()  
  union = len(set(userRated1) | set(userRated2))  
  intersection = len(set.intersection(set(userRated1), set(userRated2)))  
  return intersection / union  
print('Check hàm - Độ tương đồng của phim 1 với phim 1:', calSimMovie(1,1))  
print('Độ tương đồng của phim 1 với phim 2:', calSimMovie(1,2))

Check hàm - Độ tương đồng của phim 1 với phim 1: 1.0

Độ tương đồng của phim 1 với phim 2: 0.21711899791231734

**- Tính tử số và mẫu số cho công thức dự đoán rating của một phim mà người xem có thể đánh giá phim như thế**

[ ]

# Tính tử số và mẫu số cho công thức dự đoán rating của một phim mà người xem có thể đánh giá phim như thế  
def calNumerator(idUser, idMovie, movies, bias):  
  num=0  
  for i in movies:  
    rating=movieRatingMatrix[idUser][i+1]  
    sim=calSimMovie(idMovie,i)  
    num+=(sim \* (rating-bias))  
  return num  
  
def calDenominator(idMovie, movies):  
  den=0  
  for i in movies:  
    sim=calSimMovie(idMovie,i)  
    den+=sim  
  return den

**- Hàm Predict rating movie:**

[ ]

# predict rating movie  
def predictRatingMovie(inputUserId, inputMovieId, k=10):  
  movieSims=getSimilarMovies(inputMovieId, k)  
  bias=calBiasUser(inputUserId)  
  num=calNumerator(inputUserId, inputMovieId, movieSims, bias)  
  den=calDenominator(inputMovieId, movieSims)  
  print('bias:',bias,'num:', num,'den:', den)  
  return (bias + num/den)

**- Kiểm thử:**

[ ]

user=1  
movie=1  
k=10  
print('predict:', predictRatingMovie(user,movie,k))  
print('truth:', movieRatingMatrix[user][movie])

bias: 3.610294117647059 num: 0.28056705257353093 den: 1.5218071872218446

predict: 3.7946585068609933

truth: 5.0

[ ]

movieRatingMatrix

| **user\_id** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** | **...** | **934** | **935** | **936** | **937** | **938** | **939** | **940** | **941** | **942** | **943** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **movie\_id** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **1** | 5.0 | 4.0 | 0.0 | 0.0 | 4.0 | 4.0 | 0.0 | 0.0 | 0.0 | 4.0 | ... | 2.0 | 3.0 | 4.0 | 0.0 | 4.0 | 0.0 | 0.0 | 5.0 | 0.0 | 0.0 |
| **2** | 3.0 | 0.0 | 0.0 | 0.0 | 3.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | ... | 4.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 5.0 |
| **3** | 4.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | ... | 0.0 | 0.0 | 4.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| **4** | 3.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 5.0 | 0.0 | 0.0 | 4.0 | ... | 5.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 2.0 | 0.0 | 0.0 | 0.0 |
| **5** | 3.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | ... | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **1678** | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | ... | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| **1679** | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | ... | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| **1680** | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | ... | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| **1681** | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | ... | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| **1682** | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | ... | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |

1682 rows × 943 columns

## Đánh giá

**Đánh giá sơ bộ**

* Những phim ít đánh giá sẽ cho kết quả dự đoán rating thấp
* Nhưng phim không có đánh giá nào (phim mới vào hệ thống) sẽ không tính toán được.

## Link tham khảo

1. Chuyển dataframe thành item matrix: <https://stackoverflow.com/questions/37576594/rearrange-a-pandas-data-frame-to-create-a-2d-ratings-matrix>
2. Sparse matrix (sparse dataframe) to dense dataframe, or compressed sparse row matrix: <https://discuss.dizzycoding.com/pandas-sparse-dataframe-to-sparse-matrix-without-generating-a-dense-matrix-in-memory/>

<https://docs.scipy.org/doc/scipy/reference/generated/scipy.sparse.csr_matrix.html>

1. Link google colab: <https://colab.research.google.com/drive/16UqHiFQB1BHt3kJdWZns7BUBFVfk6O47?usp=sharing>