

ĐẠI HỌC QUỐC GIA TP.HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
KHOA TOÁN - TIN HỌC



PYTHON CHO KHOA HỌC DỮ LIỆU

Ngành: Khoa Học Dữ Liệu

Lớp: 20KDL1

Nhóm: 18

Giảng viên: Hà Văn Thảo

◇ THÀNH VIÊN ◇

STT	Tên	MSSV
1	Võ Thái Bình	20280007
2	Phạm Bảo Cương	20280010
3	Huỳnh Bảo Khang	20280050
4	Đặng Yến Linh	20280058

Ngày 20 tháng 1 năm 2023

Mục lục

1 Lời nói đầu	1
2 Loading library and dataset - Đọc thư viện và bộ dữ liệu	1
3 Exploratory Data Analysis - Phân tích dữ liệu tổng quan	2
4 Data Visualization - Trực quan hóa dữ liệu	4
5 Data Pre-processing - Tiền xử lý dữ liệu	7
6 System Building - Xây dựng hệ thống	8
6.1 POPULARITY BASED RECOMMENDATION SYSTEM	9
6.2 ITEM-BASED COLLABORATIVE FILTERING	11
6.3 CONTENT-BASED COLLABORATIVE FILTERING	15
6.4 USER-BASED COLLABORATIVE FILTERING	19
7 Kết luận	23
8 Tài liệu tham khảo	24

* BÁO CÁO *

* BOOK RECOMMENDATION SYSTEM *

* HỆ THỐNG ĐỀ XUẤT SÁCH *

1 Lời nói đầu

Sách không chỉ là những trang giấy có chữ in. Xuyên suốt lịch sử, sách là nơi ghi chép các sự kiện và là kho lưu trữ kiến thức chung của nhân loại. Nếu không có văn bản hoặc văn bản in, nền văn minh nhân loại sẽ không như ngày nay. Những nhà tư tưởng, nhà văn và những người có ảnh hưởng vĩ đại luôn là những người đọc sách. Họ dành phần lớn thời gian đọc sách để thu thập kiến thức mà sau này sẽ giúp họ thay đổi thế giới.

Sách có rất nhiều thể loại, và mọi người đều đọc sách với sở thích, nhu cầu, mục đích không giống nhau. Do đó, hệ thống đề xuất là công cụ lọc thông tin mạnh mẽ có thể giúp chúng ta tận dụng nguồn dữ liệu dồi dào hiện có để giúp cho việc đưa ra lựa chọn những cuốn sách phù hợp với nhu cầu, sở thích của mỗi cá nhân một cách dễ dàng hơn. Có hai lợi ích của việc sử dụng hệ thống đề xuất: Một mặt, nó có thể giảm lượng lớn công sức tìm kiếm sản phẩm của người dùng và giảm thiểu vấn đề quá tải thông tin. Mặt khác, nó có thể tăng giá trị kinh doanh cho các nhà cung cấp dịch vụ trực tuyến và trở thành nguồn doanh thu quan trọng.

2 Loading library and dataset - Đọc thư viện và bộ dữ liệu

Chú thích : Importing Libraries - Thêm những thư viện cần thiết

```
#Calculation, visualizing and analysis libraries
import pandas as pd
import numpy as np
import random
import matplotlib.pyplot as plt
import re

#Extract images
from google_images_download import google_images_download
from PIL import Image
import requests

#Pre-processing
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.metrics.pairwise import cosine_similarity
```

Chú thích :

- **Pandas** : Thư viện phục vụ các thao tác xử lý, phân tích dữ liệu,...
- **Numpy** : Thư viện toán học hỗ trợ xử lý các dãy số, ma trận nhiều chiều,...
- **Random** : Thư viện hỗ trợ phát sinh số ngẫu nhiên.
- **Re** : Thư viện biểu thức chính quy, Regular Expression (RegEx).
- **Matplotlib** : Thư viễn hỗ trợ trực quan hóa dữ liệu,...
- **google-images-download và PIL** : Thư viện cung cấp công cụ để tài ảnh từ Google image search và xử lý ảnh,...
- **Sklearn** : Thư viện hỗ trợ các thuật toán máy học,...

* Thư viện google-images-download ta phải thực hiện việc download và cài đặt module này thủ công.

```
def get_image_url(key_word):
    response = google_images_download.googleimagesdownload()
    arguments = {"keywords":f"{key_word} the book
                  "amazon", "limit":1, "print_urls":True, "no_download":True,
                  "size":"medium", "silent_mode":True}
    paths= response.download(arguments)
    images_paths = []
    for k, v in paths[0].items():
        images_paths += v
    return str(images_paths[0])
```

* Thiết lập hàm lấy hình ảnh từ internet qua tên quyền sách (Nguồn ảnh: Google).

Reading dataset: đọc bộ dữ liệu

```
books=pd.read_csv("Books.csv")
ratings=pd.read_csv("Ratings.csv")
users=pd.read_csv("Users.csv")
```

Nguồn bộ dữ liệu : Kaggle.

<https://www.kaggle.com/datasets/arashnic/book-recommendation-dataset>

3 Exploratory Data Analysis - Phân tích dữ liệu tổng quan

Được thu thập bởi Cai-Nicolas Ziegler từ cộng đồng Book-Crossing với sự cho phép của Ron Hornbaker, CTO của Humankind Systems. Chứa 278.858 người dùng (tên danh nhưng có thông tin nhân

khẩu học) cung cấp 1.149.780 xếp hạng (rõ ràng/ẩn) về 271.379 cuốn sách.

Chú thích : Tập dữ liệu Book-Crossing bao gồm 3 tệp như sau:

- **Users:** Chứa người dùng. Lưu ý rằng ID người dùng (User-ID) đã được ẩn danh và ánh xạ tới số nguyên. Dữ liệu nhân khẩu học được cung cấp (Location, Age) nếu có. Một khác, các trường này chứa giá trị NULL.
- **Books:** Sách được xác định bởi ISBN tương ứng của chúng. ISBN không hợp lệ đã bị xóa khỏi tập dữ liệu. Ngoài ra, một số thông tin dựa trên nội dung được cung cấp (Book-Title, Book-Author, Year-Of-Publication, Publisher), được lấy từ Amazon Web Services. Lưu ý rằng trong trường hợp có nhiều tác giả, chỉ tác giả đầu tiên được cung cấp.
- **Ratings:** Chứa thông tin xếp hạng sách. Xếp hạng (Book-Rating) là rõ ràng, được biểu thị trên thang điểm từ 1-10 (giá trị càng cao biểu thị sự đánh giá cao hơn) hoặc ẩn, được biểu thị bằng 0.

```
# number of rows and columns of the 3 dataset
print("Books Shape: " ,books.shape )
print("Ratings Shape: " ,ratings.shape )
print("Users Shape: " ,users.shape )
```

Kết quả :

```
Books Shape: (271360, 5)
Ratings Shape: (1149780, 3)
Users Shape: (278858, 3)
```

Chú thích : Bộ dữ liệu Books gồm có 271360 dòng và 5 cột, Ratings có 1149780 dòng và 3 cột còn bộ Users có 278858 dòng và 3 cột.

```
#Calculate sum null value
print("Any null values in Books:\n" ,books.isnull().sum())
```

Kết quả :

```
Any null values in Books:
ISBN          0
Book-Title     0
Book-Author    1
Year-Of-Publication  0
Publisher      2
dtype: int64
```

```
print("Any null value Users:\n",users.isnull().sum())
```

Kết quả :

```
Any null values in Users:  
User-ID      0  
Location     0  
Age         110762  
dtype: int64
```

```
print("Any null values in Ratings:\n ",ratings.isnull().sum())
```

Kết quả :

```
Any null values in Ratings:  
User-ID      0  
ISBN        0  
Book-Rating  0  
dtype: int64
```

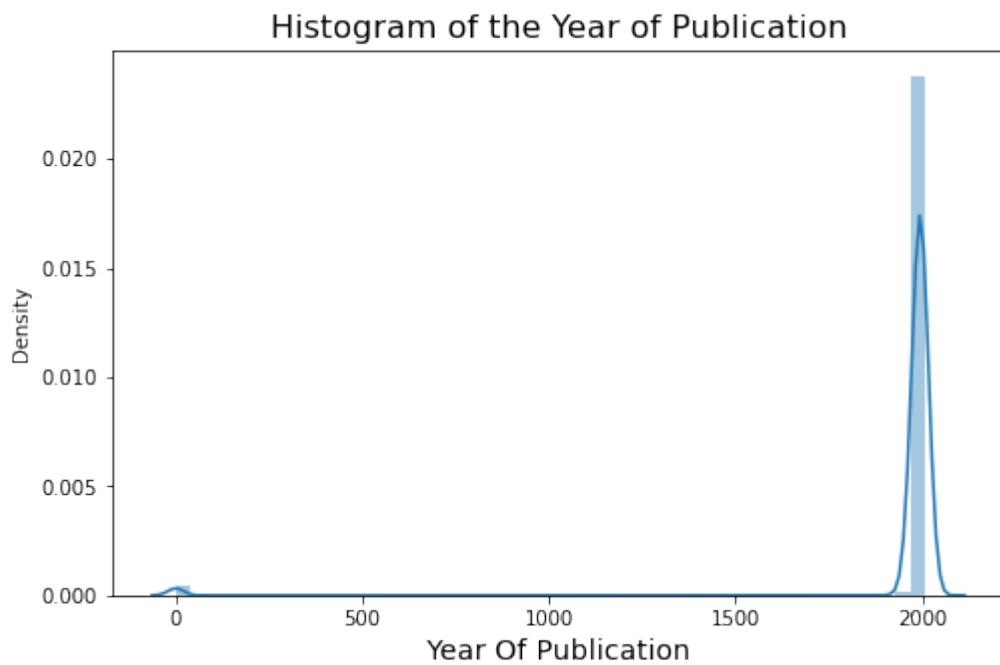
Từ các kết quả trên ta nhận thấy rằng bộ dữ liệu ta đang sử dụng tồn tại dữ liệu không hợp lệ Null nên ta sẽ tiến hàng xử lý dữ liệu.

4 Data Visualization - Trực quan hóa dữ liệu

Vẽ đồ thị cột dữ liệu năm xuất bản sách.

```
fig=plt.figure(figsize=(8,5))  
y1 = books[books['Year-Of-Publication'] >= 1960]  
y1 = y1[y1['Year-Of-Publication'] <= 2005]  
sns.distplot(books['Year-Of-Publication'])  
plt.xlabel('Year Of Publication',size=14)  
plt.title('Histogram of the Year of Publication',size=16)  
plt.show()
```

Kết quả:



Nhận thấy số năm sách xuất bản đa số nằm ở khoảng [1960,2005]. Tuy vậy cũng có những năm xuất bản 0 (dữ liệu không hợp lệ).

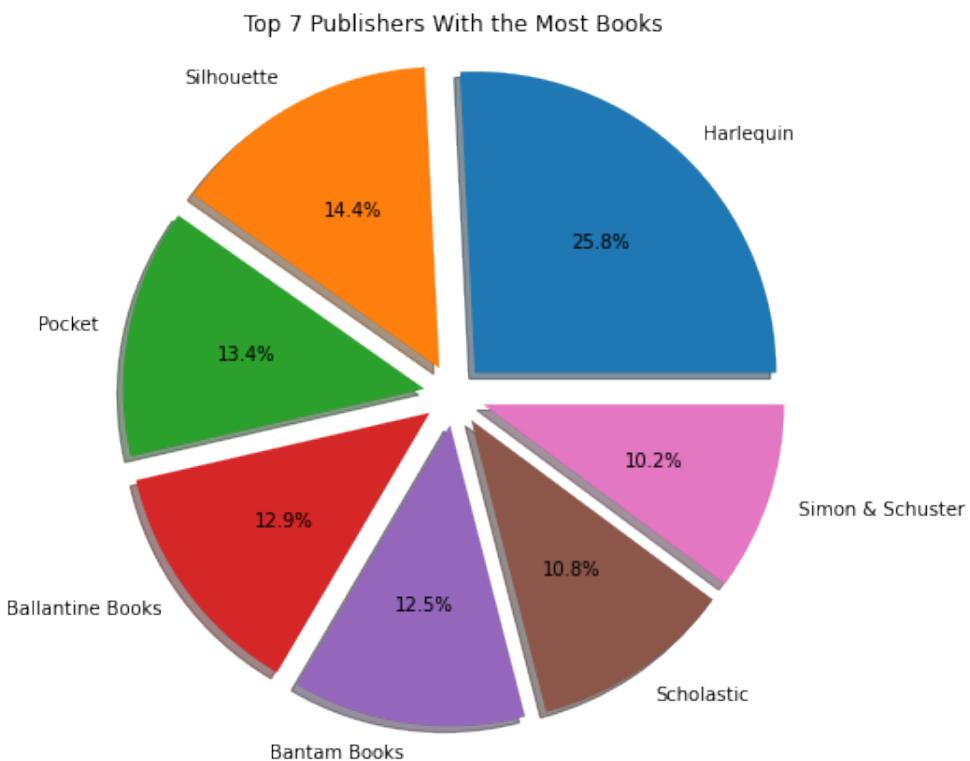
```

my_dict=(books['Publisher'].value_counts()).to_dict()
count= pd.DataFrame(list(my_dict.items()),columns = ['c','count'])
a = count.sort_values(by=['count'], ascending = False)
a.head(7)

labels = 'Harlequin', 'Silhouette', 'Pocket', 'Ballantine Books', 'Bantam
Books', 'Scholastic', 'Simon & Schuster'
sizes = [count['count'].iloc[0],count['count'].iloc[1],count['count'].iloc[2],
count['count'].iloc[3],count['count'].iloc[4],
      count['count'].iloc[5],count['count'].iloc[6]]
explode = (0.1, 0.1, 0.1, 0.1,0.1, 0.1,0.1 )
fig1 , ax1 = plt.subplots(figsize=(7,7))
ax1.pie(sizes,
         explode = explode,
         labels = labels,
         autopct = '%1.1f%%',
         shadow = True,
         startangle = 0)
plt.title("Top 7 Publishers With the Most Books")
ax1.axis ('equal')
plt.show()

```

Kết quả:

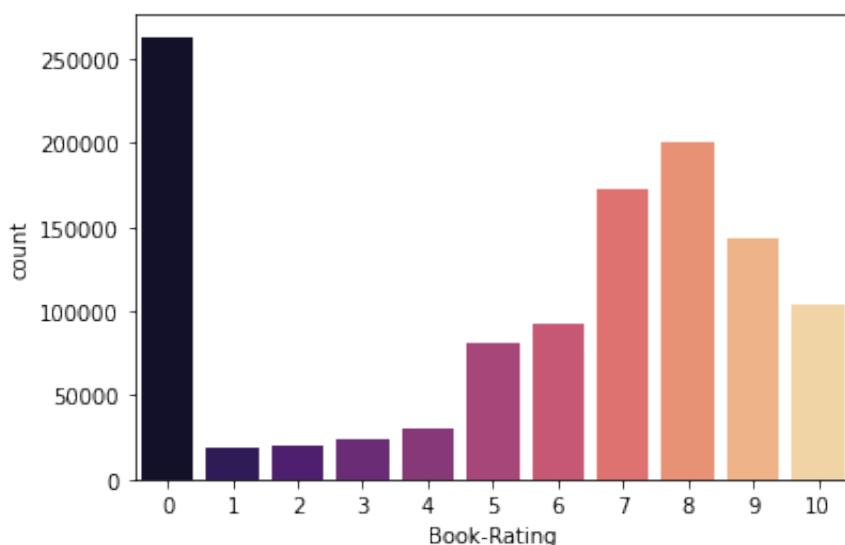


★ Đây là top 7 nhà xuất bản có số lượng xuất bản sách nhiều nhất.

Để có một cái nhìn tổng quát về lượt đánh giá trong bộ dữ liệu ta tiến hành vẽ đồ thị của cột đánh giá.

```
sns.countplot(data=ratings , x='Book-Rating' , palette='magma')
```

Kết quả :



★ Từ đồ thị ta thấy bộ dữ liệu ratings có dữ liệu không hợp lệ nên ta tiến hành các bước xử lý: bỏ đi các dữ liệu NaN, bỏ đi những cột dữ liệu ta không dùng đến, bỏ đi những Rating bằng 0,...

5 Data Pre-processing - Tiền xử lý dữ liệu

Với dữ liệu bảng, việc các trường thông tin bị thiếu là thường xuyên xảy ra. Việc này đến từ quá trình thu thập dữ liệu. Các giá trị bị thiếu này có thể ảnh hưởng đến độ chính xác của hệ thống nên ta cần xử lý nó. Trong trường hợp này, bộ dữ liệu không có giá trị null mà mang giá trị 0 (được xem như một giá trị bị thiếu). Có hai giải pháp để xử lý dữ liệu bị thiếu này : Một là loại bỏ hết các giá trị 0, và hai là thay thế các giá trị này của từng biến bằng giá trị trung bình của biến đó.

Vì trong bộ dữ liệu ratings có những quyển sách mà ISBN của chúng không có trong bộ dữ liệu books (không hợp lệ) nên ta sẽ loại bỏ chúng.

```
books_data=books.merge(ratings, on="ISBN")
books_data.shape
```

Kết quả :

```
(1031136, 7)
```

- * Bộ dữ liệu sau khi loại bỏ dữ liệu không hợp lệ gồm có 1031136 dòng và 7 cột. Lúc này ISBN giảm từ 1149780 dòng xuống 1031136 dòng và 7 cột.

- * Từ bước trực quan hóa dữ liệu thấy bộ dữ liệu ratings có dữ liệu không hợp lệ nên ta tiến hành bước xử lý tiếp theo.

```
df=books_data.copy()
df.dropna(inplace=True)
df.reset_index(drop=True, inplace=True)
df.drop(columns=["ISBN", "Year-Of-Publication"], axis=1, inplace=True)
df.drop(index=df[df["Book-Rating"]==0].index, inplace=True)
df["Book-Title"]=df["Book-Title"].apply(lambda x: re.sub("[\W_]+", " ", x).strip())
df.head(10)
```

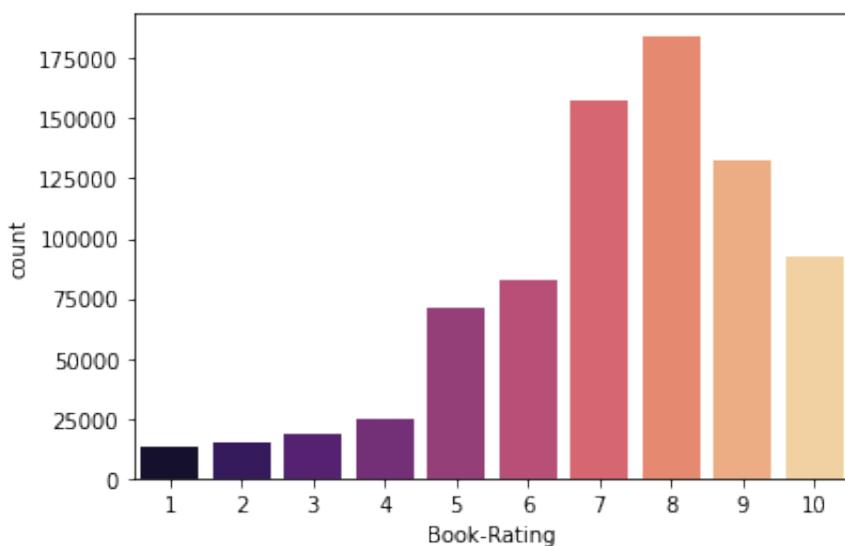
Kết quả :

	Book-Title	Book-Author	Publisher	User-ID	Book-Rating
0	Classical Mythology	Mark P. O. Morford	Oxford University Press	2	8
1	Clara Callan	Richard Bruce Wright	HarperFlamingo Canada	8	5
2	Clara Callan	Richard Bruce Wright	HarperFlamingo Canada	11400	9
3	Clara Callan	Richard Bruce Wright	HarperFlamingo Canada	11676	8
4	Clara Callan	Richard Bruce Wright	HarperFlamingo Canada	41385	7
5	Clara Callan	Richard Bruce Wright	HarperFlamingo Canada	67544	8
6	Clara Callan	Richard Bruce Wright	HarperFlamingo Canada	85526	7
7	Clara Callan	Richard Bruce Wright	HarperFlamingo Canada	96054	9
8	Clara Callan	Richard Bruce Wright	HarperFlamingo Canada	116866	9
9	Clara Callan	Richard Bruce Wright	HarperFlamingo Canada	123629	9

* Dataframe mới sau khi đã xử lý làm sạch dữ liệu. Từ đây ta thu được bộ dataframe chính là df.

```
sns.countplot(data=df , x='Book-Rating' , palette='magma')
```

Kết quả :



* Từ đồ thị sau khi xử lý dữ liệu ta thấy đồ thị dữ liệu lượt đánh giá tuân theo phân phối chuẩn có độ lệch trái. Cho thấy dữ liệu tập trung chủ yếu ở xung quanh lượt đánh giá 8 với tần xuất xuất hiện cao nhất. Từ đó nhận thấy đa số người dùng trong bộ dữ liệu đánh giá sách dao động xung quanh điểm 8.

6 System Building - Xây dựng hệ thống

Hệ thống đề xuất là hệ thống được thiết kế để giới thiệu mọi thứ cho người dùng dựa trên nhiều yếu tố khác nhau. Các hệ thống này dự đoán sản phẩm có khả năng nhất mà người dùng có nhiều

khả năng mua và quan tâm nhất. Các công ty như Netflix , Amazon, v.v. sử dụng hệ thống đề xuất để giúp người dùng của họ xác định đúng sản phẩm cho họ. Các hệ thống đề xuất dựa trên mức độ phổ biến dựa trên xếp hạng các mặt hàng của tất cả người dùng. Hệ thống đề xuất dựa trên mức độ phổ biến hoạt động theo xu hướng. Về cơ bản, nó sử dụng các mặt hàng đang là xu hướng hiện nay.

6.1 POPULARITY BASED RECOMMENDATION SYSTEM

```
def popular_books(df,n=100):
    rating_count=df.groupby("Book-Title").count()[["Book-Rating"]].reset_index()
    rating_count.rename(columns={"Book-Rating":"NumberOfVotes"},inplace=True)
    rating_average=df.groupby("Book-Title")["Book-Rating"].mean().reset_index()
    rating_average.rename(columns={"Book-Rating":"AverageRatings"},inplace=True)

    popularBooks=rating_count.merge(rating_average,on="Book-Title")

    def weighted_rate(x):
        v=x["NumberOfVotes"]
        R=x["AverageRatings"]

        return ((v*R) + (m*C)) / (v+m)

    C=popularBooks["AverageRatings"].mean()
    m=popularBooks["NumberOfVotes"].quantile(0.90)

    popularBooks=popularBooks[popularBooks["NumberOfVotes"] >=250]
    popularBooks["Popularity"]=popularBooks.apply(weighted_rate,axis=1)
    popularBooks=popularBooks.sort_values(by="Popularity",ascending=False)
    return popularBooks[["Book-Title","NumberOfVotes","AverageRatings",
    "Popularity"]].reset_index(drop=True).head(n)
```

* Công thức trên được sử dụng để tính 250 đầu sách được xếp hạng cao nhất. Công thức này cung cấp một 'ước tính Bayes' đúng, có tính đến số phiếu bầu mà mỗi tựa sách đã nhận được, số phiếu bầu tối thiểu cần thiết để có tên trong danh sách và số phiếu bầu trung bình cho tất cả các tựa sách:

Xếp hạng có trọng số (WR) = $((v \times R) + (m \times C)) / (v+m)$

Ở đây:

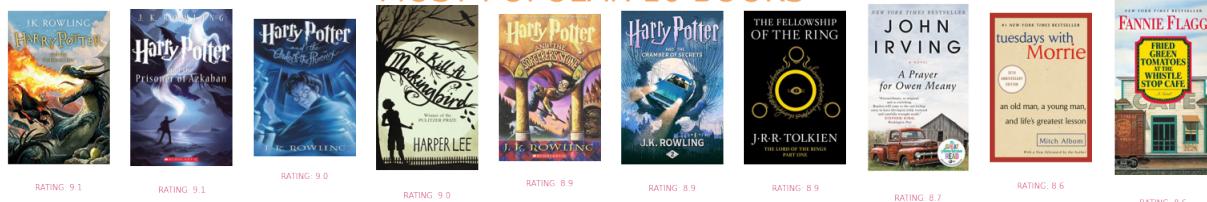
- R = trung bình cho sách (trung bình) = (xếp hạng)
- v = số lượt bình chọn cho sách = (phiếu)
- m = số phiếu tối thiểu cần thiết để được liệt kê trong danh sách Top 250 (hiện tại là 25.000)
- C = bình chọn trung bình trên toàn bộ báo cáo

- ★ M nhóm đã sử dụng 90% làm giới hạn, nghĩa là để một cuốn sách xuất hiện trong danh sách, nó phải có nhiều phiếu bầu hơn ít nhất 90% .

```
n=10
top=pd.DataFrame(popular_books(df,n))
fig,ax=plt.subplots(1,n,figsize=(27,4))
fig.suptitle(f" MOST POPULAR {n} BOOKS",fontsize=40,color="sandybrown")
for i in range(n):
    title=top["Book-Title"].tolist()[i]
    url = get_image_url(title)
    img=Image.open(requests.get(url,stream=True).raw)
    ax[i].imshow(img)
    ax[i].axis("off")
    ax[i].set_title("RATING: {}"
                    .format(round(df[df["Book-Title"]==top["Book-Title"].tolist()[i]]
                    ["Book-Rating"].mean(),1)),y=-0.20,color="palevioletred",fontsize=10)
```

Kết quả :

MOST POPULAR 10 BOOKS



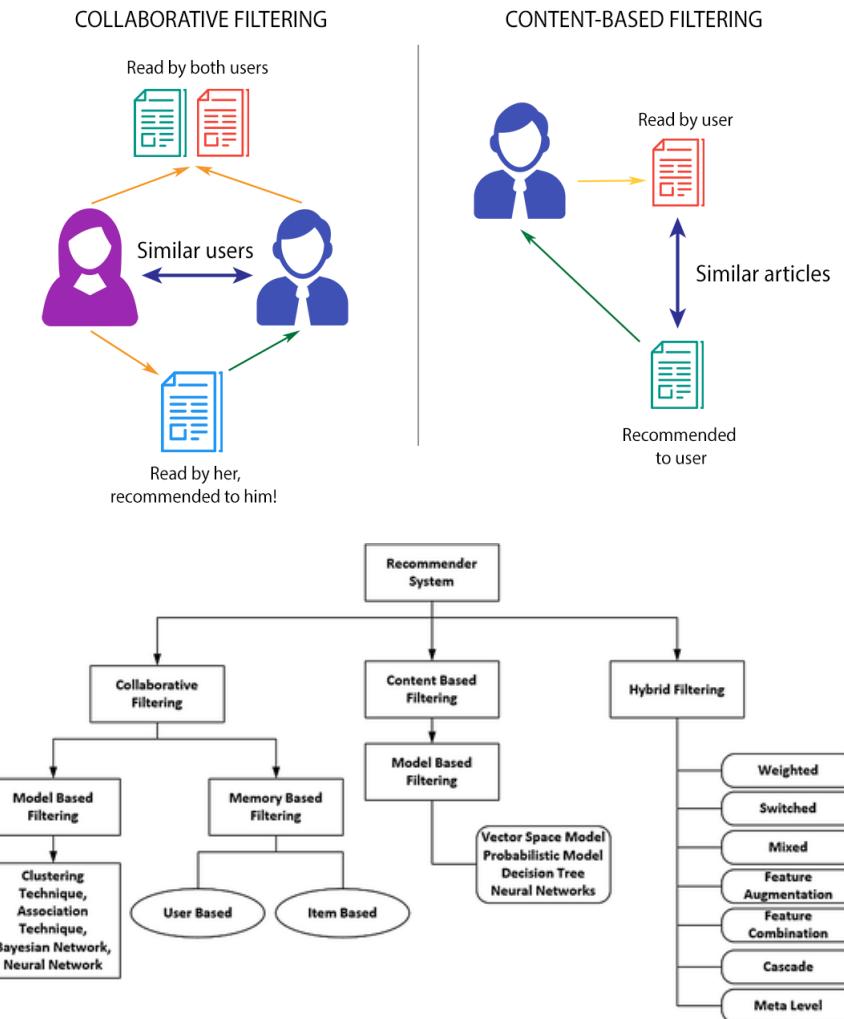
- ★ Kết hợp các hàm ta xây dựng được hệ thống đề xuất top 10 cuốn sách phổ biến. Ta thấy rằng đa số là một loạt tiểu thuyết huyền bí Harry Potter của nhà văn J. K. Rowling rất nổi tiếng.

Tiếp theo ta xây dựng các tính năng đặc thù khác phục vụ cho việc đề xuất tốt hơn.

Nhìn chung có 2 loại hệ thống đề xuất chính : Lọc dựa trên nội dung (content based) và lọc cộng tác (collaborative filtering).

Sự khác biệt chính giữa mỗi loại, có thể được tổng hợp theo loại của tuyên bố mà người tiêu dùng có thể đưa ra. Chẳng hạn, mô hình chính của hệ thống đề xuất trên nội dung được điều khiển bởi tuyên bố: "Chỉ cho tôi nhiều hơn những gì tôi đã thích trước đây." Các hệ thống dựa trên nội dung cố gắng tìm ra những khía cạnh yêu thích của người dùng đối với một mặt hàng là gì và sau đó đưa ra đề xuất về các mặt hàng cùng chia sẻ những khía cạnh đó.

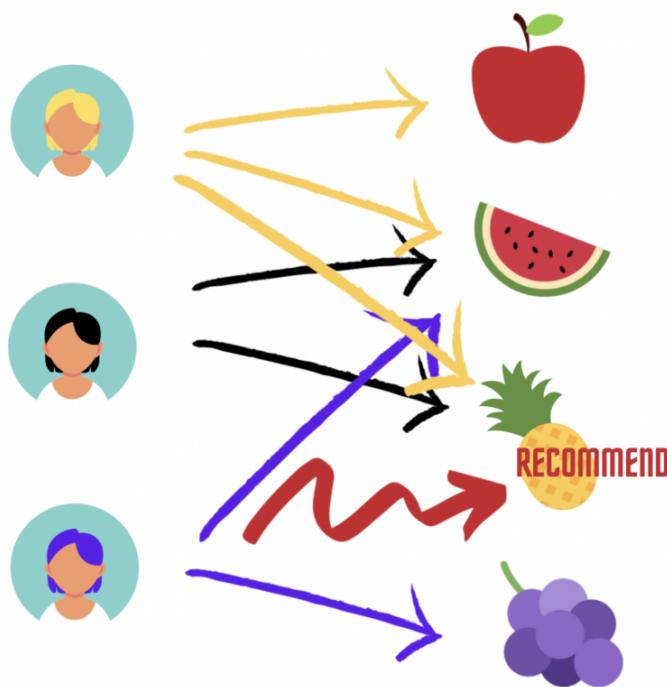
Lọc cộng tác dựa trên câu nói của người dùng , "Nói cho tôi biết những gì phổ biến với các hàng xóm của tôi vì tôi cũng có thể thích nó." Kỹ thuật lọc cộng tác tìm thấy các nhóm người dùng tương tự và cung cấp các đề xuất dựa trên thị hiếu tương tự trong nhóm đó. Nói tóm lại, nó giả định rằng một người sử dụng có thể quan tâm đến những gì các người dùng tương tự khác quan tâm.



★ Có một loại hệ thống đề xuất thứ ba, được gọi là Hybrid Filtering.

6.2 ITEM-BASED COLLABORATIVE FILTERING

Hệ tư vấn lọc cộng tác dựa trên sản phẩm là một mô hình hệ tư vấn được phát triển dựa trên hệ tư vấn lọc cộng tác. Nó được giới thiệu lần đầu tiên trên tạp chí ACM vào năm 2001. Không lâu sau đó, hệ thống này được Amazon.com ứng dụng để giới thiệu các sản phẩm của họ đến người dùng.



Biểu đồ này minh họa cách hoạt động của tính năng lọc cộng tác dựa trên sản phẩm bằng cách sử dụng một ví dụ đơn giản.

- Ms. Blond thích táo, dưa hấu và dứa. Ms. Black thích dưa hấu và dứa. Ms. Purple thích dưa hấu và nho.
- Bởi vì dưa hấu và dứa được những người dùng mẫu yêu thích nên chúng được coi là những mặt hàng tương tự nhau.
- Vì Ms. Purple thích dưa hấu và Ms. Purple chưa tiếp xúc với dứa nên hệ thống khuyến nghị giới thiệu dứa cho Ms. Purple.

Lọc các quyền sách bằng phương pháp lọc cộng tác dựa trên các sản phẩm. Trong phương pháp này, hệ tư vấn tìm ra các sản phẩm để giới thiệu đến người dùng bằng cách tìm ra sự tương đồng giữa các sản phẩm qua được suy luận từ kết quả xếp hạng của người dùng. Ví dụ, một người dùng cụ thể thích một tập hợp các sản phẩm thì tất cả các sản phẩm đó được coi là tương đồng với nhau. Nếu hai sản phẩm được xếp hạng cao cùng một người dùng thì hai sản phẩm này được xem là tương đồng và người dùng được hy vọng sẽ có cùng sở thích trên các sản phẩm tương đồng. Hệ tư vấn lọc cộng tác dựa trên sản phẩm là giải pháp hiệu quả cho các hệ thống tư vấn online. Bởi vì trong các hệ thống này, số lượng người dùng thường tăng rất nhanh so với số lượng sản phẩm. Do đó, việc tìm ra kết quả tư vấn dựa trên các người dùng sẽ phức tạp và mất nhiều thời gian hơn so với việc tìm ra kết quả tư vấn dựa trên các sản phẩm.

```
def item_based(bookTitle):
```

```

bookTitle=str(bookTitle)
n= 5
fig,ax=plt.subplots(1,n,figsize=(17,6))
if bookTitle in df[ "Book-Title" ].values:
    rating_count=pd.DataFrame(df[ "Book-Title" ].value_counts())
    rare_books=rating_count[rating_count[ "Book-Title" ]<=200].index
    common_books=df[~df[ "Book-Title" ].isin(rare_books)]


if bookTitle in rare_books:
    #number of book recommendation
    most_common=pd.Series(common_books[ "Book-Title" ].unique()).sample(n).values
    print("No Recommendations for this Book \n ")
    fig.suptitle(f"You May Try",fontsize=40,color="coral")
    for i in range(n):
        title = most_common[i]
        url = get_image_url(title)
        img=Image.open(requests.get(url,stream=True).raw)
        ax[i].imshow(img)
        ax[i].axis("off")
        ax[i].set_title("RATING: {}".format(round(df[df[ "Book-Title" ]==
            most_common[i]][ "Book-Rating" ].mean(),1)),y=-0.20,color="palevioletred",
        fontsize=20)

else:
    common_books_pivot=common_books.pivot_table(index=[ "User-ID" ],
    columns=[ "Book-Title" ],values="Book-Rating")
    title=common_books_pivot[bookTitle]
    recommendation_df=pd.DataFrame(
        common_books_pivot.corrwith(title).sort_values(ascending=False))
    .reset_index(drop=False)

    if bookTitle in [title for title in recommendation_df[ "Book-Title" ]]:
        recommendation_df=recommendation_df.drop(recommendation_df[recommendation_df[ "Book-Title" ]==bookTitle].index[0])

    less_rating=[]
    for i in recommendation_df[ "Book-Title" ]:
        if df[df[ "Book-Title" ]==i][ "Book-Rating" ].mean() < 5:
            less_rating.append(i)
    if recommendation_df.shape[0] - len(less_rating) > 5:
        recommendation_df=recommendation_df[~recommendation_df

```

```

["Book-Title"].isin(less_rating)]


recommendation_df=recommendation_df[0:n]
recommendation_df.columns=["Book-Title","Correlation"]


fig.suptitle("WOULD YOU LIKE to TRY THESE
BOOKS?", fontsize=40, color="deepskyblue")
for i in range(len(recommendation_df["Book-Title"].tolist())):
    title=recommendation_df["Book-Title"].tolist()[i]
    url = get_image_url(title)
    img=Image.open(requests.get(url, stream=True).raw)
    ax[i].imshow(img)
    ax[i].axis("off")
    ax[i].set_title("RATING:{}".format(round(df[df["Book-Title"]==recommendation_df["Book-Title"].tolist()[i]]["Book-Rating"].mean(),1)),
    y=-0.20, color="palevioletred", fontsize=22)
else:
    print("    COULD NOT FIND    ")

```

Chú thích:

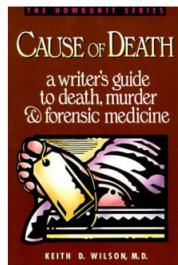
Trong phương pháp này ta sẽ phân những quyển sách có số lượng đánh giá nhỏ hơn 200 là những quyển sách ít được đánh giá còn lại là những quyển sách phổ biến. Nếu quyển sách nằm trong bộ những quyển sách ít được đánh giá ta sẽ báo không có đề xuất dành cho quyển sách này và đưa ra đề xuất những quyển sách phổ biến khác.

Ngược lại, ta sẽ tạo một pivot_table với index User-ID còn từng cột là Book-Title, giá trị của bảng chính là số lượt đánh giá của từng người dùng đối với mỗi quyển sách. Từ đó, có thể biết được số lượt đánh giá của từng người vào cuốn sách đó và dùng phương thức corrwith() để nhận được các mối tương quan giữa title đối với common_books_pivot - sắp xếp theo chiều giảm dần, sau cùng tìm được đề xuất từ quyển sách mà ta cần đề xuất.

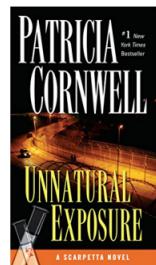
```
item_based("The Da Vinci Code")
```

Kết quả :

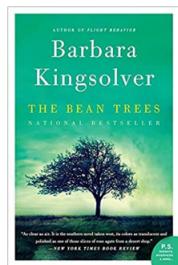
WOULD YOU LIKE to TRY THESE BOOKS?



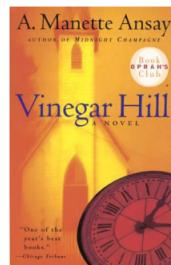
RATING: 7.6



RATING: 7.9



RATING: 8.5



RATING: 6.9



RATING: 7.9

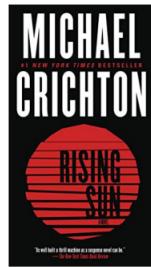
```
item_based("The Snow Garden")
```

Kết quả : No Recommendations for this Book

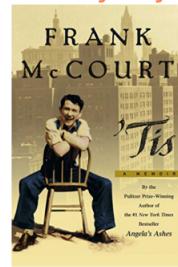
You May Try



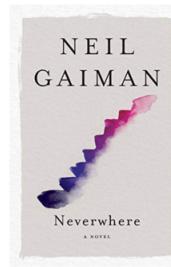
RATING: 4.1



RATING: 7.1



RATING: 7.4



RATING: 8.1



RATING: 8.1

```
item_based("Khoa Hoc Du Lieu")
```

Kết quả : COULD NOT FIND

6.3 CONTENT-BASED COLLABORATIVE FILTERING

Lọc các quyển sách bằng phương pháp lọc công tác dựa trên nội dung. Lọc dựa trên nội dung sử dụng các điểm tương đồng trong sản phẩm, dịch vụ, cũng như thông tin tích lũy được về người dùng để đưa ra đề xuất các sản phẩm khác tương tự như nội dung người dùng thích, dựa trên các thao tác trước đó hoặc dựa trên đặc tính của sản phẩm.

Trong các hệ thống content-based, chúng ta cần xây dựng một bộ hồ sơ (profile) cho mỗi item. Profile này được biểu diễn dưới dạng toán học là một feature vector. Trong những trường hợp đơn giản, feature vector được trực tiếp trích xuất từ item.

Lọc công tác dựa trên nội dung sử dụng CountVectorizer - một công cụ tuyệt vời được cung cấp bởi thư viện scikit-learning bằng Python. Nó được sử dụng để chuyển một văn bản nhất định thành một vectơ trên cơ sở tần suất (số lượng (count)) của mỗi từ xuất hiện trong toàn bộ văn bản.

Cosine Similarity:

Cosin của hai vectơ khác 0 có thể được suy ra bằng cách sử dụng công thức tích Euclidean :

$$\mathbf{A} \cdot \mathbf{B} = \|\mathbf{A}\| \|\mathbf{B}\| \cos \theta$$

Cho trước hai vectơ thuộc tính, A và B , similarity cosin, $\cos(\theta)$, được biểu diễn bằng tích vô hướng và độ lớn:

$$\text{cosine similarity} = S_C(A, B) := \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\| \|\mathbf{B}\|} = \frac{\sum_{i=1}^n A_i B_i}{\sqrt{\sum_{i=1}^n A_i^2} \sqrt{\sum_{i=1}^n B_i^2}}$$

Dộ similarity của hai vector là 1 số trong đoạn [-1, 1]. Giá trị bằng 1 thể hiện hai vector hoàn toàn similar nhau. Hàm số cos của một góc bằng 1 nghĩa là góc giữa hai vector bằng 0, tức một vector bằng tích của một số dương với vector còn lại. Giá trị cos bằng -1 thể hiện hai vector này hoàn toàn trái ngược nhau. Tức khi hành vi của hai users là hoàn toàn ngược nhau thì similarity giữa hai vector đó là thấp nhất.

* Python có hàm hỗ trợ tính toán độ tương tự cosin trong thư viện scikit-learning: `cosine_similarity()`

```
def content_based(bookTitle):
    bookTitle=str(bookTitle)
    n=5
    if bookTitle in df["Book-Title"].values:
        rating_count=pd.DataFrame(df["Book-Title"].value_counts())
        rare_books=rating_count[rating_count["Book-Title"]<=200].index
        common_books=df[~df["Book-Title"].isin(rare_books)]

        if bookTitle in rare_books:
            most_common=pd.Series(common_books["Book-Title"].unique()).sample(n).values
            print("No Recommendations for this Book \n ")
            fig,ax=plt.subplots(1,n,figsize=(27,6))
            fig.suptitle(f"You May Try",fontsize=40,color="coral")
            for i in range(n):
                title = most_common[i]
                url = get_image_url(title)
                img=Image.open(requests.get(url,stream=True).raw)
                ax[i].imshow(img)
                ax[i].axis("off")
                ax[i].set_title("RATING: {}".format(round(df[df["Book-Title"]==most_common[i]]["Book-Rating"].mean(),1)),y=-0.20,
                               color="palevioletred",fontsize=20)
        else:
            common_books=common_books.drop_duplicates(subset=["Book-Title"])
```

```

common_books.reset_index(inplace=True)
common_books["index"]=[i for i in range(common_books.shape[0])]
targets=["Book-Title", "Book-Author", "Publisher"]
common_books["all_features"] = [" ".join(common_books[targets].iloc[i,:].values)
                                for i in range(common_books[targets].shape[0])]

vectorizer=CountVectorizer()
common_booksVector=vectorizer.fit_transform(common_books["all_features"])
similarity=cosine_similarity(common_booksVector)

index=common_books[common_books["Book-Title"]==bookTitle]["index"].values[0]
similar_books=list(enumerate(similarity[index]))
similar_booksSorted=sorted(similar_books,key=lambda x:x[1],reverse=True)[1:n+1]
books=[]

for i in range(len(similar_booksSorted)):

    books.append(common_books[common_books["index"]==similar_booksSorted[i][0]]["Book-Title"].item())

fig,ax=plt.subplots(1,len(books),figsize=(27,5))
fig.suptitle("YOU MAY ALSO LIKE THESE BOOKS",fontsize=40,color="deepskyblue")
for i in range(len(books)):

    title=books[i]
    url = get_image_url(title)
    img=Image.open(requests.get(url,stream=True).raw)
    ax[i].imshow(img)
    ax[i].axis("off")
    ax[i].set_title("RATING: "+str(round(df[df["Book-Title"]==books[i]]["Book-Rating"].mean(),1)),y=-0.2,color="palevioletred",fontsize=22)
fig.show()

else:
    print(" COULD NOT FIND ")

```

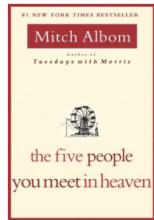
```
content_based("Tuesdays with Morrie An Old Man a Young Man and Life s Greatest Lesson")
```

Kết quả :

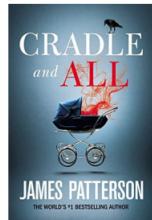
YOU MAY ALSO LIKE THESE BOOKS



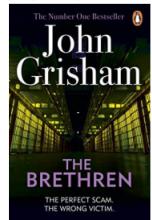
RATING: 7.6



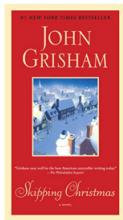
RATING: 8.0



RATING: 7.4



RATING: 7.6

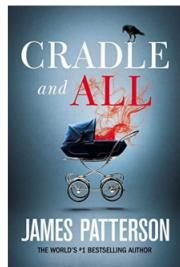


RATING: 7.5

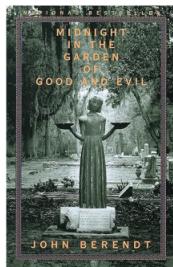
content_based("A Soldier of the Great War")

Kết quả : No Recommendations for this Book

You May Try



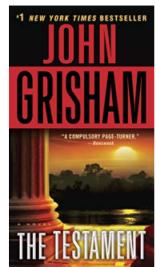
RATING: 7.4



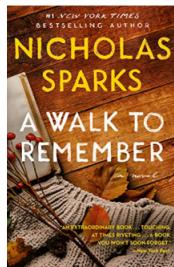
RATING: 7.9



RATING: 8.1



RATING: 7.6



RATING: 7.9

content_based("Dai Hoc Khoa Hoc Tu Nhien")

Kết quả : COULD NOT FIND

*Chú thích : Để dễ hiểu hơn về phương pháp này và cách áp dụng Cosine similarity vào bài toán cụ thể, ta cùng xét ví dụ sau:

Đưa ra bên dưới là một bảng tập hợp có chứa một số nội dung và người dùng đã xếp hạng các nội dung đó. Xếp hạng rõ ràng và theo thang điểm từ 1 đến 10. Mỗi mục trong bảng biểu thị xếp hạng do Người dùng thứ Ai đưa ra cho Nội dung thứ Aj . Trong hầu hết các trường hợp, phần lớn các ô trống do người dùng chỉ xếp hạng cho một số nội dung.

User/Content	Content_1	Content_2	Content_3
User_1	2	-	3
User_2	5	2	-
User_3	3	3	1
User_4	-	2	2

- **Bước 1:** Tìm điểm tương đồng của tất cả các cặp nội dung (ta tìm giá trị cho các CountVectorizer từ đó đưa ra kết luận).

Hình thành các cặp nội dung. Ví dụ: trong ví dụ này, các cặp mặt hàng là (Mặt hàng_1,

Mặt hàng_2), (Mặt hàng_1, Mặt hàng_3) và (Mặt hàng_2, Mặt hàng_3). Chọn từng nội dung để ghép nối cùng cái một. Sau đó, chúng tôi tìm thấy tất cả những người dùng đã xếp hạng cho cả hai mặt hàng trong cặp mặt hàng. Tạo một vectơ cho mỗi nội dung(CountVectorizer()) và tính toán sự giống nhau giữa hai nội dung(cosine_similarity) bằng cách sử dụng công thức cosin đã nêu ở trên.

$$\text{Similarity}(\text{Content1}, \text{Content2})$$

Trong bảng, chúng ta chỉ có thể thấy User_1 và User_2 đã xếp hạng cho cả content 1 và 2.

Vì vậy, hãy để I1 là vectơ cho Content_1 và I2 là cho Content_2. Sau đó:

- $I1 = 5U2 + 3U3$
- $I2 = 2U2 + 3U3$

$$\Rightarrow S(I1, I2) = \frac{(5 * 2) + (3 * 3)}{\sqrt{5^2 + 3^2} \sqrt{2^2 + 3^2}} = 0.90$$

Ta thực hiện tương tự với Similarity(Content1, Content3) và Similarity(Content1, Content3)

- **Bước 2:** Tạo xếp hạng còn thiếu trong bảng(Bước này ta không thực hiện bởi vì ta đã xử lý hết các đánh giá bị trống ở phần Tiền xử lí dữ liệu, tuy nhiên em vẫn giới thiệu sơ qua cách áp dụng) Cụ thể bây giờ, trong bước này, ta tính toán xếp hạng bị thiếu trong bảng.

- Xếp hạng Content_2 cho User_1:

$$\Rightarrow r(U_1, I_2) = \frac{r(U_1, I_1)*s_{I_1 I_2} + r(U_1, I_3)*s_{I_3 I_2}}{s_{I_1 I_2} + s_{I_3 I_2}} = \frac{(2*0,9)+(3*0,869)}{(0,9+0,869)} = 2,49$$

- Xếp hạng Content_3 cho User_2:

$$\Rightarrow r(U_2, I_3) = \frac{r(U_2, I_1)*s_{I_1 I_3} + r(U_2, I_2)*s_{I_2 I_3}}{s_{I_1 I_3} + s_{I_2 I_3}} = \frac{(5*0,789)+(2*0,869)}{(0,789+0,869)} = 3,43$$

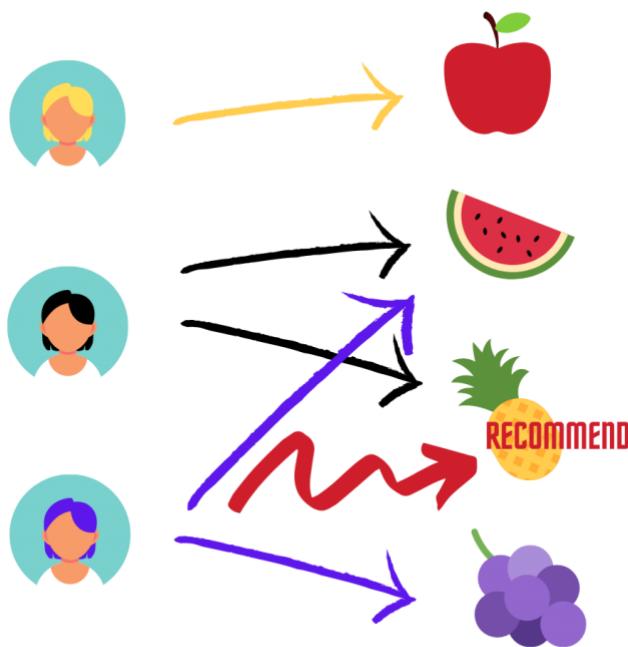
- Xếp hạng Content_1 cho User_4:

$$\Rightarrow r(U_4, I_1) = \frac{r(U_4, I_2)*s_{I_1 I_2} + r(U_4, I_3)*s_{I_1 I_3}}{s_{I_1 I_2} + s_{I_1 I_3}} = \frac{(2*0,9)+(2*0,789)}{(0,9+0,789)} = 2,0$$

Tiếp tục quay lại bước 1 thực hiện tương tự cho các cặp bị thiếu trong mảng, qua đó ta có thể dựa vào độ similarity của 2 vector để đưa ra kết quả bài toán (chọn ra được các quyển sách bằng phương pháp lọc nội dung).

6.4 USER-BASED COLLABORATIVE FILTERING

Lọc các quyển sách bằng phương pháp lọc cộng tác dựa trên một người đọc sử dụng sự tương đồng giữa các người đọc để dự đoán sở thích của người đọc từ đó đưa ra đề xuất.



Biểu đồ này minh họa cách hoạt động của tính năng lọc cộng tác dựa trên người dùng bằng cách sử dụng một ví dụ đơn giản hóa.

- Ms. Blond thích táo. Ms. Black thích dưa hấu và dứa. Ms. Purple thích dưa hấu và nho.
- Vì Ms. Black và Ms. Purple cùng thích một loại trái cây là dưa hấu nên họ là những đối tượng sử dụng giống nhau.
- Do Ms. Black thích dứa và Ms. Purple chưa tiếp xúc với dứa nên hệ thống khuyến nghị giới thiệu dứa cho Ms. Purple.

Việc xác định mức độ quan tâm của mỗi user tới một item dựa trên mức độ quan tâm của similar users tới item đó còn được gọi là User-user collaborative filtering. Công việc quan trọng nhất phải làm trước tiên trong User-user Collaborative Filtering là phải xác định được sự giống nhau (similarity) giữa hai users. Tính toán độ tương tự giữa hai người dùng, chúng ta có thể sử dụng độ tương tự cosine.

```
# Drop users who vote less than 100 times.
new_df=df[df['User-ID'].map(df['User-ID'].value_counts()) > 100]
users_pivot=new_df.pivot_table(index=["User-ID"],columns=["Book-Title"],values="Book-Rating")
users_pivot.fillna(0,inplace=True)
```

- ★ Loại bỏ những người đọc với đánh giá nhỏ hơn 100 lần. Sau đó, ta sẽ tạo một pivot_table với index User-ID còn từng cột là Book-Title, giá trị của bảng chính là số lượt đánh giá của từng người dùng đối với mỗi quyển sách.

```
def users_choice(id):

    users_fav=new_df[new_df["User-ID"]==id].sort_values(["Book-Rating"],ascending=False)[0:5]
    return users_fav
```

- * Tìm những quyển sách mà người đọc đánh giá cao nhất (thích) bằng cách sắp xếp lượt đánh giá theo chiều giảm dần.

```
def common(new_df,user,user_id):
    x=new_df[new_df["User-ID"]==user_id]
    recommend_books=[]
    user=list(user)
    for i in user:
        y=new_df[(new_df["User-ID"]==i)]
        books=y[y["Book-Title"].isin(x["Book-Title"])]
        books=books.sort_values(["Book-Rating"],ascending=False)[0:5]
        recommend_books.extend(books["Book-Title"].values)

    return recommend_books[0:5]
```

- * Tìm những quyển sách phổ biến đối với từng người dùng từ đó đưa ra top 5 quyển sách đề xuất.

```
def user_based(new_df,id):
    if id not in new_df["User-ID"].values:
        print("    User NOT FOUND    ")

    else:
        index=np.where(users_pivot.index==id)[0][0]
        similarity=cosine_similarity(users_pivot)
        similar_users=list(enumerate(similarity[index]))
        similar_users = sorted(similar_users,key = lambda x:x[1],reverse=True)[0:5]

        user_rec=[]

        for i in similar_users:
            data=df[df["User-ID"]==users_pivot.index[i[0]]]
            user_rec.extend(list(data.drop_duplicates("User-ID")["User-ID"].values))

    return user_rec
```

- * Xây dựng hàm lọc cộng tác theo người dùng. Hàm sử dụng độ tương tự cosine như các phương

pháp lọc cộng tác khác bên ta vừa trình bày để xây dựng được cơ sở sự tương đồng giữa các người dùng với nhau từ đó ta có thể xây dựng nên hệ thống và đề xuất cho từng người dùng.

```

user_id=random.choice(new_df["User-ID"].values)
user_choice_df=pd.DataFrame(users_choice(user_id))
user_favorite=users_choice(user_id)
n=len(user_choice_df["Book-Title"].values)
print("      USER: {}".format(user_id))

fig,ax=plt.subplots(1,n,figsize=(27,5))
fig.suptitle("YOUR FAVORITE BOOKS",fontsize=40,color="orchid")

for i in range(n):
    title=user_choice_df["Book-Title"].tolist()[i]
    url = get_image_url(title)
    img=Image.open(requests.get(url,stream=True).raw)
    ax[i].imshow(img)
    ax[i].axis("off")
    ax[i].set_title("RATING: {}".format(round(new_df[new_df["Book-Title"]==
        user_choice_df["Book-Title"].tolist()[i]]["Book-Rating"].mean(),1)),
        y=-0.20,color="palevioletred",fontsize=22)

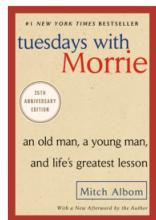
user_based_rec=user_based(new_df,user_id)
books_for_user=common(new_df,user_based_rec,user_id)
books_for_userDF=pd.DataFrame(books_for_user,columns=["Book-Title"])

fig,ax=plt.subplots(1,5,figsize=(27,5))
fig.suptitle("YOU MAY ALSO LIKE THESE BOOKS",fontsize=40,color="deepskyblue")
for i in range(5):
    title=books_for_userDF["Book-Title"].tolist()[i]
    url = get_image_url(title)
    img=Image.open(requests.get(url,stream=True).raw)
    ax[i].imshow(img)
    ax[i].axis("off")
    ax[i].set_title("RATING: {}".format(round(new_df[new_df["Book-Title"]==
        books_for_userDF["Book-Title"].tolist()[i]]["Book-Rating"].mean(),1)),y=-0.20,
        color="palevioletred",fontsize=22)

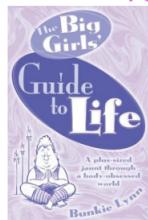
```

Kết quả :

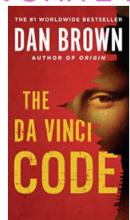
YOUR FAVORITE BOOKS



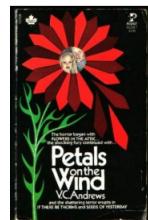
RATING: 8.8



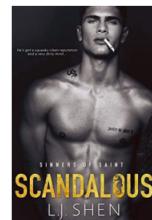
RATING: 10.0



RATING: 8.2

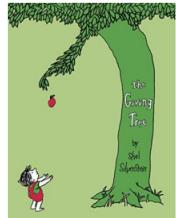


RATING: 7.7

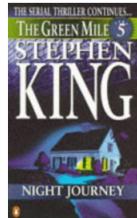


RATING: 6.7

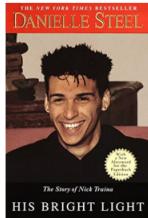
YOU MAY ALSO LIKE THESE BOOKS



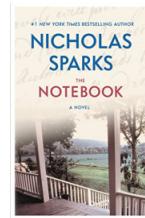
RATING: 9.5



RATING: 8.7



RATING: 7.8



RATING: 8.0



RATING: 10.0

★ Ta sẽ bốc ngẫu nhiên 1 người dùng trong bộ dữ liệu ta đang thao tác. Từ đó tìm ra những quyển sách mà người đó đánh giá cao nhất và đưa ra sách người đó thích nhất.

Với người dùng đó ta cũng tìm những cuốn sách khác để đề xuất từ những người dùng cùng chung sở thích bằng cách hàm ta đã xây dựng. Lúc này ta tìm được những quyển sách đề xuất mà người dùng đó có thể thích bằng phương pháp lọc cộng tác dựa trên người dùng.

7 Kết luận

Hệ thống đề xuất là một công nghệ mới mạnh mẽ để trích xuất giá trị bổ sung cho doanh nghiệp từ cơ sở dữ liệu người dùng của nó. Hệ thống mang lại lợi ích cho người dùng bằng cách cho phép họ tìm thấy các nhiều cuốn sách mà họ thích một cách dễ dàng hơn. Ngược lại, họ giúp doanh nghiệp bằng cách tạo ra nhiều doanh thu hơn.

Về phần báo cáo mà nhóm em đã thực hiện. Từ một bộ dữ liệu chứa thông tin về của hơn 271000 cuốn sách, và các người dùng đã đánh giá nó, nhóm đã trực quan hóa dữ liệu, xử lý các dữ liệu bất hợp lý, cũng như đưa ra các đề xuất cho người dùng theo nhiều cách khác nhau. Ở đây, nếu người dùng của chúng ta là người dùng mới, chúng ta sẽ đề xuất cho người dùng dựa trên top 10 các cuốn sách nổi tiếng nhất. Nếu là người dùng đã từng sử dụng nền tảng trực tuyến của chúng ta, khi họ nhập vào ID mình thì có thể thấy được top 5 cuốn sách yêu thích nhất của bản thân, bên cạnh đó hệ thống cũng sẽ đề xuất cho họ những cuốn sách mà họ có thể thích. Ngoài ra, người dùng cũng có thể tìm kiếm sự đề xuất bằng cách nhập vào tên của cuốn sách, hệ thống sẽ tiến hành đề xuất dựa trên nội dung, hay dựa trên sự tương đồng của sản phẩm. Bằng các cách lọc này hệ thống sẽ đưa ra cho người dùng các cuốn sách tương tự cuốn sách mà họ tìm kiếm - nếu cuốn sách mà họ tìm kiếm nằm

trong bộ dữ liệu và có một độ phổ biến nhất định, ngược lại những cuốn sách chưa phổ biến thì sẽ không có sự đề xuất nào cả.

Nhìn chung về bài làm mà nhóm đã thực hiện code vẫn chưa thật sự tối ưu. Hơn nữa, hệ thống của chúng ta vẫn còn nhiều thiếu sót, chẳng hạn khi ta có một cuốn sách mới và cuốn sách này chưa phổ biến nhưng có đánh giá khá tốt, nghĩa là cuốn sách này sẽ có xu hướng có thể nổi tiếng trong thời gian tới, đối với cách làm mà nhóm thực hiện vẫn chưa cải thiện được trường hợp này. Nhóm em sẽ cố gắng cải thiện trong thời gian tới để hệ thống được hoàn thiện hơn.

Cuối cùng, đọc sách không chỉ cung cấp tri thức cho chúng ta, mà còn mang lại nhiều lợi ích hơn thế .Đọc sách giúp ta cải thiện sự tập trung và tăng cường khả năng tư duy, phân tích ; vốn từ của chúng ta cũng sẽ được mở rộng thông qua việc đọc sách.Đọc sách còn là một hình thức giải trí , giảm căng thẳng ... Mọi người nên khắc sâu thói quen đọc sách để cải thiện khả năng đọc, từ vựng và nhận thức về các chủ đề khác nhau.

Hy vọng bằng cách sử dụng hệ thống đề xuất sách, chúng ta có thể tìm kiếm những cuốn sách phù hợp với sở thích , nhu cầu dễ dàng hơn, từ đó mà chúng ta có thể nuôi dưỡng thói quen đọc sách.

8 Tài liệu tham khảo

<https://www.quora.com/How-does-IMDB-compute-popularity>

<https://www.analyticsvidhya.com/blog/2022/02/introduction-to-collaborative-filtering/>

<https://machinelearningcoban.com/2017/05/24/collaborativefiltering/>

https://en.wikipedia.org/wiki/Cosine_similarity

<https://grabngoinfo.com/recommendation-system-item-based-collaborative-filtering/>