TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI

Trường Công nghệ thông tin và Truyền thông

BÁO CÁO BÀI TẬP LỚN

HỆ THỐNG GỢI Ý PHIM

Học phần: Nhập môn Trí tuệ Nhân tạo – IT3160

GVHD: TS. Trần Thế Hùng

Nhóm 14

Họ và tên	MSSV
Phạm Việt Hải	20215044
Trần Bình Minh	20215094
Nguyễn Lê Sơn	20215131
Nguyễn Văn Dũng	20215012

MỤC LỤC

I. Lời mở đầu

1. Đề tài

Hệ thống gợi ý phim là một trong những ứng dụng quan trọng của trí tuệ nhân tạo và học máy, được sử dụng rộng rãi trên các nền tảng như Netflix, Amazon Prime, và nhiều dịch vụ xem phim trực tuyến khác. Mục tiêu chính của hệ thống này là đề xuất cho người dùng những bộ phim mà họ có thể thích dựa trên lịch sử xem phim của họ cũng như hành vi và sở thích của người dùng tương tự.

2. Hệ thống gợi ý

2.1. Khái niệm

Hệ gợi ý (Recommender System) là một hệ thống sử dụng các kỹ thuật và phương pháp của trí tuệ nhân tạo, học máy và xử lý dữ liệu để dự đoán và đề xuất các mục (items) mà người dùng có thể quan tâm. Các mục này có thể là sản phẩm, dịch vụ, nội dung số như phim, âm nhạc, sách, bài viết, và nhiều loại hình khác. Hệ gợi ý được sử dụng rộng rãi trong các ngành công nghiệp như thương mại điện tử, truyền thông, giải trí và mạng xã hội.

2.2. Mục tiêu

Mục tiêu chính của hệ gợi ý là cải thiện trải nghiệm người dùng bằng cách cá nhân hóa các đề xuất dựa trên sở thích và hành vi của họ. Bằng cách này, hệ gợi ý giúp:

- Tăng cường sự hài lòng của người dùng: Đưa ra các gợi ý phù hợp với sở thích của người dùng, giúp họ tìm thấy những mục thú vị một cách dễ dàng.

- Tăng doanh thu và tương tác: Khuyến khích người dùng mua sắm hoặc tương tác nhiều hơn bằng cách đề xuất các sản phẩm hoặc nội dung liên quan.

- Khám phá các mục mới: Giúp người dùng khám phá những mục mà họ chưa biết đến nhưng có thể quan tâm.

2.3. Các phương pháp của hệ gợi ý

Có hai phương pháp chính được sử dụng trong hệ gợi ý:

- Lọc cộng tác (Collaborative Filtering):

+ Dựa trên người dùng (User-Based): Đề xuất các mục dựa trên sở thích của những người dùng có hành vi tương tự.

+ Dựa trên mục (Item-Based): Đề xuất các mục dựa trên sự tương đồng giữa các mục với nhau.

- Lọc nội dung (Content-Based Filtering): Đề xuất các mục dựa trên các thuộc tính và đặc điểm của chính các mục đó. Ví dụ: đề xuất các phim có cùng thể loại hoặc cùng đạo diễn.

II. Cơ sở lý thuyết

1. Học máy

1.1. Khái niệm

Học máy (machine learning) là một lĩnh vực của trí tuệ nhân tạo (AI) tập trung vào việc xây dựng các mô hình và thuật toán cho phép máy tính học hỏi từ dữ liệu và cải thiện hiệu suất của chúng theo thời gian mà không cần phải lập trình một cách rõ ràng. Thay vì được lập trình cụ thể để thực hiện một nhiệm vụ, máy tính sử dụng các thuật toán học máy để tự động học hỏi và rút ra các mẫu hoặc xu hướng từ dữ liệu.

1.2. Tập dữ liệu tập luyện và kiểm tra

- Tập Dữ Liệu Tập Luyện (Train Set)

Khái niệm: Tập dữ liệu tập luyện là tập hợp các ví dụ được sử dụng để huấn luyện mô hình học máy. Dữ liệu trong tập luyện bao gồm cả đầu vào (input) và đầu ra (output) mong muốn.

Vai trò: Mô hình sử dụng dữ liệu này để tìm hiểu mối quan hệ giữa các biến đầu vào và đầu ra, điều chỉnh các tham số của mình sao cho có thể đưa ra dự đoán chính xác.

- Tập Dữ Liệu Kiểm Tra (Test Set)

Khái niệm: Tập dữ liệu kiểm tra là tập hợp các ví dụ được sử dụng để đánh giá hiệu suất của mô hình sau khi nó đã được huấn luyện. Dữ liệu trong tập kiểm tra thường không được sử dụng trong quá trình huấn luyện.

Vai trò: Mô hình sử dụng dữ liệu này để kiểm tra xem nó có thể tổng quát hóa tốt như thế nào với dữ liệu mới, tức là có thể dự đoán chính xác các đầu ra mà nó chưa từng thấy trước đó.

1.3. Xác thực chéo (Cross-Validation)

- Khái niệm: Cross-validation (xác thực chéo) là một kỹ thuật được sử dụng trong học máy để đánh giá độ chính xác của mô hình và đảm bảo rằng mô hình không bị overfitting (quá khớp) hoặc underfitting (không đủ khớp). Cross-validation giúp tận dụng tối đa dữ liệu có sẵn để huấn luyện và kiểm tra mô hình, đảm bảo rằng mô hình có thể tổng quát hóa tốt trên các dữ liệu chưa từng thấy.

- Phương pháp:

+ K-Fold: Chia tập dữ liệu thành k phần (fold) có kích thước gần bằng nhau. Lặp lại k lần: trong mỗi lần lặp, sử dụng k-1 phần để huấn luyện mô hình và 1 phần để kiểm tra mô hình. Lấy trung bình các kết quả kiểm tra để đánh giá hiệu suất của mô hình.

+ Leave-One-Out Cross-Validation (LOOCV): Mỗi mẫu dữ liệu sẽ lần lượt được sử dụng làm tập kiểm tra trong khi các mẫu còn lại được sử dụng để huấn luyện. Lặp lại quá trình này cho tất cả các mẫu trong tập dữ liệu. Tính trung bình các kết quả kiểm tra để đánh giá mô hình.

- Mục tiêu:

Đánh giá Độ Tin Cậy của Mô Hình: Giúp đưa ra ước lượng chính xác hơn về hiệu suất của mô hình so với việc chỉ sử dụng một tập kiểm tra duy nhất.

Giảm Nguy Cơ Overfitting: Bằng cách sử dụng nhiều tập kiểm tra khác nhau, cross-validation giúp đảm bảo rằng mô hình không chỉ học tốt trên dữ liệu huấn luyện mà còn trên dữ liệu mới.

Sử Dụng Tối Đa Dữ Liệu: Giúp tận dụng tối đa dữ liệu có sẵn, đặc biệt hữu ích khi có ít dữ liệu.

1.4. Quy trình

- Thu thập và tiền xử lý dữ liệu

- Chia dữ liệu: (70-80% tập luyện và 20-30% tập kiểm tra)

- Huấn luyện mô hình

- Đánh giá mô hình

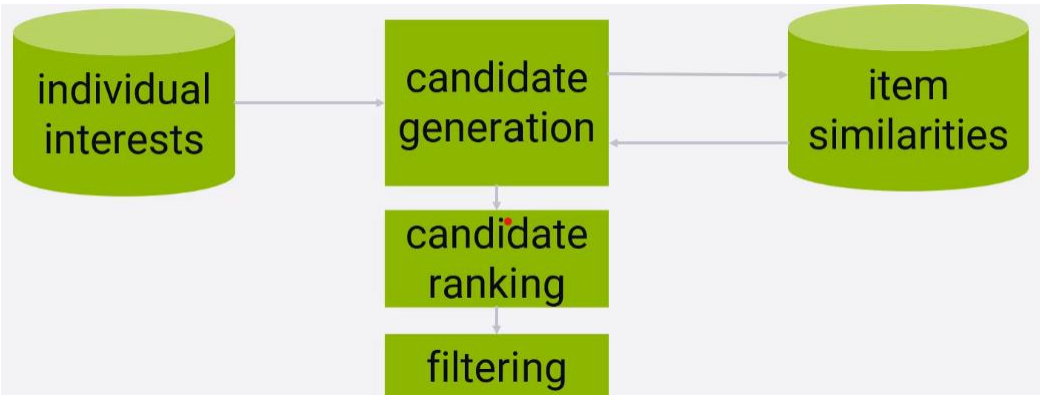
- Triển khai và cải tiến

2. Kiến trúc Top-N

2.1. Khái niệm

Gợi ý Top-N (Top-N Recommendation) là một loại hệ thống gợi ý trong đó mục tiêu là đề xuất một danh sách ngắn các mục tốt nhất (thường là N mục) mà người dùng có thể quan tâm. Ví dụ, trong một hệ thống gợi ý phim, mục tiêu là đưa ra danh sách N bộ phim mà người dùng có khả năng sẽ thích dựa trên sở thích và hành vi của họ.

2.2. Quy trình



- Xác định sở thích cá nhân, dựa vào lịch sử đánh giá phim của người dùng

- Tạo danh sách ứng viên (candidate generation): Từ dữ liệu về sở thích cá nhân, hệ thống sẽ tạo ra một danh sách các mục ứng viên có khả năng phù hợp với người dùng dựa trên nội dung và cộng tác.

- Tính toán độ tương đồng: Trong bước này, hệ thống sẽ tính toán độ tương đồng giữa các mục để xác định những mục nào là tương tự nhau.

- Lọc kết quả

3. Thông số đánh giá

- MAE (Mean Absolute Error): MAE là giá trị trung bình của các lỗi tuyệt đối giữa giá trị dự đoán và giá trị thực tế. Nó đo lường mức độ chênh lệch trung bình giữa các dự đoán của mô hình và các giá trị thực tế. Giá trị MAE càng thấp, mô hình càng chính xác.

$$\frac{\sum_{i=1}^{n}|y_i - x_i|}{n}$$

- RMSE (Root Mean Square Error): RMSE là căn bậc hai của giá trị trung bình của các lỗi bình phương giữa giá trị dự đoán và giá trị thực tế. Nó nhạy cảm với các lỗi lớn hơn so với MAE. Giá trị RMSE càng thấp, mô hình càng tốt. RMSE = 0 là tốt nhất.

$$\sqrt{\frac{\sum_{i=1}^{n}(y_i - x_i)^2}{n}}$$

- HR (Hit Rate): Tỷ lệ trúng đo lường tỷ lệ dự đoán chính xác của hệ thống gợi ý, tức là hệ thống gợi ý một mục và mục đó thực sự được người dùng ưa thích hoặc tương tác. HR càng cao càng tốt.

- AHRH (Average Reciprocal Hit Rate): ARHR là một biến thể của Hit Rate, tính đến vị trí của mục được trúng trong danh sách gợi ý. Nó đánh trọng số cao hơn cho các mục xuất hiện ở các vị trí đầu. ARHR càng cao càng tốt.

$$\frac{\sum_{i=1}^{n}\frac{1}{rank_i}}{Users}$$

- cHR (Cumulative Hit Rate): Cumulative Hit Rate là tổng hợp của các Hit Rate qua các danh sách gợi ý khác nhau. Nó đánh giá khả năng của hệ thống gợi ý trên toàn bộ tập hợp các danh sách gợi ý. cHR càng cao càng tốt.

- rHR (Rating Hit Rate): Rating Hit Rate đo lường tỷ lệ các gợi ý có xếp hạng đúng với mong đợi của người dùng. Nó đánh giá độ chính xác của các xếp hạng trong hệ thống gợi ý. rHR càng cao càng tốt.

- Coverage: Coverage là tỷ lệ các mục được hệ thống gợi ý đến tổng số mục trong tập dữ liệu. Nó đánh giá phạm vi bao phủ của hệ thống gợi ý. Coverage càng cao càng tốt, cho thấy hệ thống có khả năng gợi ý nhiều mục khác nhau.

- Diversity: Diversity đánh giá mức độ khác nhau giữa các mục trong danh sách gợi ý. Nó quan trọng để tránh gợi ý các mục quá giống nhau. Diversity càng cao càng tốt, cho thấy hệ thống gợi ý đa dạng các mục khác nhau.

- Novelty: Novelty đánh giá mức độ mới mẻ của các mục trong danh sách gợi ý, tức là hệ thống gợi ý các mục mà người dùng chưa từng thấy hoặc ít tương tác. Novelty càng cao càng tốt, giúp người dùng khám phá các mục mới.

- Churn: Churn đo lường mức độ thay đổi trong danh sách gợi ý qua các lần gợi ý khác nhau. Nó quan trọng để đảm bảo rằng hệ thống gợi ý không quá tĩnh. Churn vừa phải là tốt, quá cao hoặc quá thấp đều không tốt.

- Responsiveness: Responsiveness đánh giá tốc độ phản hồi của hệ thống gợi ý khi có thay đổi trong hành vi người dùng. Responsiveness càng cao càng tốt, cho thấy hệ thống có thể điều chỉnh nhanh chóng theo thay đổi của người dùng.

- A/B Tests: A/B Tests là phương pháp so sánh hiệu quả giữa hai phiên bản của hệ thống gợi ý (phiên bản A và phiên bản B) trên một tập hợp người dùng ngẫu nhiên. Dựa vào kết quả A/B Tests, chọn phiên bản có hiệu suất tốt hơn dựa trên các chỉ số đánh giá như HR, MAE, RMSE, etc.

4. Content-based Filtering

4.1. Khái niệm

Content-Based Filtering (lọc dựa trên nội dung) là một kỹ thuật gợi ý dựa trên các thuộc tính của các mục mà người dùng đã thể hiện sự quan tâm hoặc tương tác trước đó. Hệ thống sử dụng thông tin về nội dung của các mục (như thể loại phim, tác giả, diễn viên, vv.) và tạo ra hồ sơ sở thích của người dùng dựa trên những thông tin này.

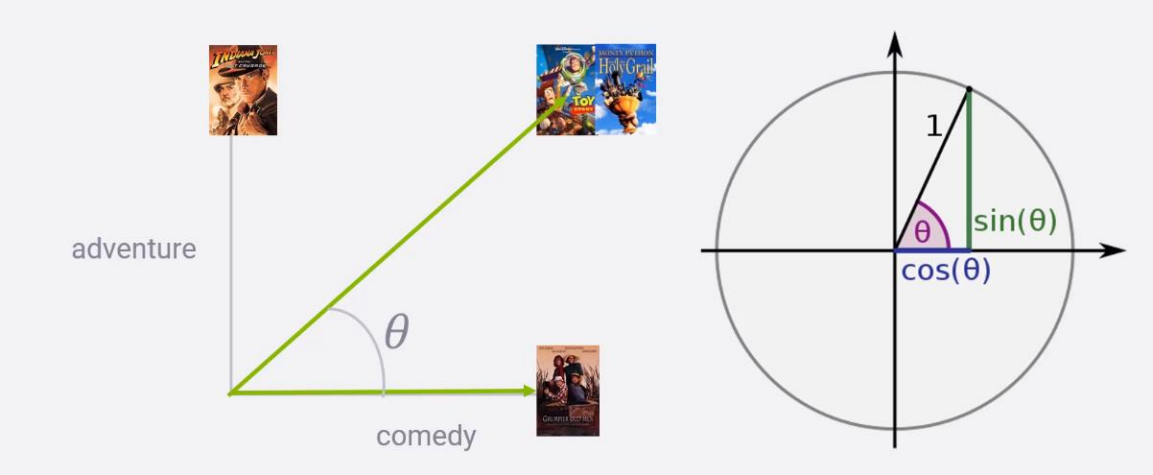
4.2. Nguyên lý và ứng dụng

- Biểu diễn nội dung của các bộ phim: Mỗi phim có 18 thể loại (genres) có thể thuộc vào, mỗi thể loại sẽ là 1 chiều của vector

movieId	title	genres
1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy
2	Jumanji (1995)	Adventure|Children|Fantasy
3	Grumpier Old Men (1995)	Comedy|Romance
4	Waiting to Exhale (1995)	Comedy|Drama|Romance
5	Father of the Bride Part II (1995)	Comedy

- Biểu diễn người dùng: Mỗi người dùng được biểu diễn bằng một hồ sơ (profile) của các bộ phim họ đã tương tác trước đó.

- Tính toán độ tương tự: Độ tương tự giữa các mặt hàng được tính bằng cách so sánh đặc trưng hoặc thuộc tính của chúng. Phương pháp tính độ tương tự được sử dụng bao gồm cosine similarity:



Chuyển phim thành các vector:

0 title	genres
1 Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy
2 Jumanji (1995)	Adventure|Children|Fantasy
3 Grumpier Old Men (1995)	Comedy|Romance
4 Waiting to Exhale (1995)	Comedy|Drama|Romance

Movie	action	adventure	animation	children's	comedy	crime	documentary	drama	fantasy	film-noir	horror	musical	western	mystery	romance	sci-fi	thriller	war	western2
Toy Story	0	1	1	1	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0
Jumanji	0	1	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
Grumpier Old Men	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0
Waiting to Exhale	0	0	0	0	1	0	0	1	0	0	0	0	0	0	1	0	0	0	0
Father of the Bride	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Công thức tính độ tương tự:

$$CosSim(x,y) = \frac{\sum_i x_i y_i}{\sqrt{\sum_i x_i^2}\sqrt{\sum_i y_i^2}}$$

Thời gian phát hành của phim:

Toy Story (1995)
Jumanji (1995)
Grumpier Old Men (1995)
Waiting to Exhale (1995)
Father of the Bride Part II (1995)
Heat (1995)
Sabrina (1995)

- Đề xuất mặt hàng tương tự: k-nearest-neighbors (đưa ra đánh giá dự đoán của một người dùng cho một bộ phim họ chưa đánh giá)

+ Tính toán độ tương tự của bộ phim đang xét đối với tất cả các bộ phim đã được người dùng đánh giá

+ Các bộ phim gần nhất có độ tương đồng cao nhất

+ Lấy trung bình độ tương đồng với đánh giá của bộ phim nó tương đồng với

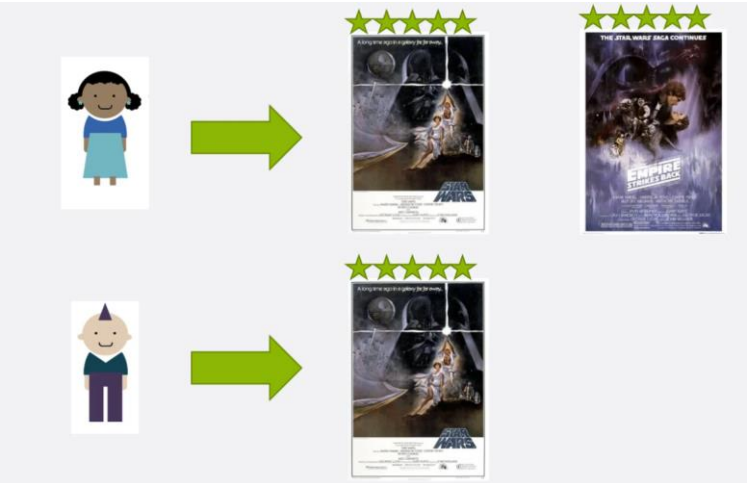
5. Neighborhood-based Collaborative Filtering

5.1. Khái niệm

Collaborative Filtering (lọc cộng tác) là một kỹ thuật gợi ý dựa trên hành vi và sở thích của người dùng khác. Phương pháp này không cần thông tin về nội dung của các mục, mà dựa trên tương tác (ví dụ: xếp hạng, click) giữa các người dùng và các mục.

5.2. Nguyên lý hoạt động

- User-based collaborative filtering:



Tìm những người dùng tương đồng lịch sử hoạt động và gợi ý phim chưa tương tác

Ma trận dữ liệu:

	Indiana Jones	Star Wars	Empire Strikes Back	Incredibles	Casablanca
Bob	4	5			
Ted					1
Ann		5	5	5	

Chuyển các user thành các vector với chiều là những bộ phim, trọng số là đánh giá (VD: Bob (4, 5, 0, 0, 0)) => Tương đồng qua công thức cosin

Ma trận tương đồng:

	Bob	Ted	Ann
Bob	1	0	1
Ted	0	1	0
Ann	1	0	1

=> Đánh giá các hàng xóm của Bob: Ann - 1.0; Ted - 0

Gợi ý các bộ phim với điểm cao nhất (đánh giá * trọng số tương đồng * số lần xuất hiện (trong TH có nhiều người dùng khác tham chiếu)

- Item-based colaborative filtering:

Gợi ý những phim tương đồng với phim đã tương tác

Ma trận dữ liệu:

	Bob	Ted	Ann
Indiana Jones	4		
Star Wars	5		5
Empire Strikes Back			5
Incredibles			5
Casablanca		1	

Chuyển các bộ phim thành vector với chiều là những người dùng, trọng số là đánh giá (VD: Indiana Jones (4, 0, 0)) => Tương đồng qua công thức cosin

Ma trận tương đồng:

	Indiana Jones	Star Wars	Empire Strikes Back	Incredibles	Casablanca
Indiana Jones	1	1	0	0	0
Star Wars	1	1	1	1	0
Empire Strikes Back	1	1	1	1	0
Incredibles	1	1	1	1	0
Casablanca	0	0	0	0	1

* Chú ý: các chỉ số trong ma trận tương đồng của 2 VD trong thực tế có trọng số ý nghĩa hơn chỉ 0 và 1.

III. Triển khai

1. Công nghệ sử dụng

1.1. Ngôn ngữ lập trình và môi trường

- Python: Python là một ngôn ngữ lập trình bậc cao, dễ đọc, dễ viết và rất linh hoạt. Nó hỗ trợ nhiều phong cách lập trình khác nhau như lập trình hướng đối tượng, lập trình hàm và lập trình thủ tục.

- Anaconda Navigator: Anaconda Navigator là một giao diện đồ họa người dùng (GUI) quản lý các môi trường và gói phần mềm trong Anaconda, một phân phối Python cho khoa học dữ liệu. Anaconda Navigator giúp tạo, quản lý và chuyển đổi giữa các môi trường ảo Python một cách dễ dàng.

- Spyder: Spyder là một môi trường phát triển tích hợp (IDE) dành cho Python, được thiết kế chủ yếu cho khoa học dữ liệu và kỹ thuật. Spyder tích hợp mạnh mẽ với các thư viện như NumPy, SciPy, Matplotlib, Pandas, IPython và các công cụ khác.

1.2. Thư viện

- Suprise: Surprise (Simple Python Recommendation System Engine) là một thư viện Python dành cho việc xây dựng và phân tích các hệ thống gợi ý, đặc biệt là xử lý dữ liệu đánh giá. Surprise cung cấp nhiều thuật toán gợi ý sẵn có như SVD, KNN, NMF... giúp dễ dàng so sánh và thử nghiệm. Người dùng có thể dễ dàng tạo và thử nghiệm các thuật toán gợi ý mới.

- BeautifulSoup: BeautifulSoup là một thư viện Python để phân tích cú pháp HTML và XML, giúp trích xuất dữ liệu từ các trang web.

- Pandas: Pandas là một thư viện Python mạnh mẽ dành cho việc thao tác và phân tích dữ liệu, đặc biệt là dữ liệu dạng bảng (tabular data).

1.3. Quản lý

- Microsoft Teams: Họp, thảo luận

- Github: Quản lý phiên bản

2. Thu thập và xử lý dữ liệu

2.1. Thu thập

Mô tả: Dữ liệu được thu thập trên imdb, gồm 3600 các đánh giá của người dùng cho các bộ phim.

Code:

```
import requests
from bs4 import BeautifulSoup
import pandas as pd
import logging

# Setup logging configuration
logging.basicConfig(level=logging.INFO, format='%(asctime)s - %(levelname)s - %(message)s')

# URLs to scrape
urls = [
    'https://www.imdb.com/title/tt0068646/reviews?ref_=tt_urv',
    'https://www.imdb.com/title/tt0108052/reviews?ref_=tt_urv',
    'https://www.imdb.com/title/tt0110912/reviews?ref_=tt_urv',
    'https://www.imdb.com/title/tt0111161/reviews?ref_=tt_urv',
    'https://www.imdb.com/title/tt0071562/reviews?ref_=tt_urv',
    'https://www.imdb.com/title/tt3783958/reviews?ref_=tt_urv',
    'https://www.imdb.com/title/tt0109830/reviews?ref_=tt_urv',
    'https://www.imdb.com/title/tt0133093/reviews?ref_=tt_urv',
    'https://www.imdb.com/title/tt0245429/reviews?ref_=tt_urv',
    'https://www.imdb.com/title/tt1375666/reviews?ref_=tt_urv'
]

headers = {'User-Agent': 'Mozilla/5.0', 'Accept-Language': 'en-US'}
base_url = 'https://www.imdb.com'
all_data = []

for url in urls:
    try:
        logging.info(f'Starting to process URL: {url}')
        response = requests.get(url, headers=headers)
        soup = BeautifulSoup(response.text, 'html.parser')
```

```
        review_containers = soup.find_all('div', class_='lister-item-content')

        for container in review_containers:
            user_link_tag = container.find('span', class_='display-name-link').find('a')
            if user_link_tag:
                user_profile_url = base_url + user_link_tag['href']
                user_id = user_link_tag['href'].split('/')[2]
                user_ratings_url = user_profile_url.split('?')[0] + 'ratings/'
                logging.info(f'Processing ratings for user {user_id} from URL: {user_ratings_url}')

                try:
                    user_response = requests.get(user_ratings_url, headers=headers)
                    user_soup = BeautifulSoup(user_response.text, 'html.parser')

                    for item in user_soup.find_all('div', class_='lister-item mode-detail', limit=15):
                        movie_link_tag = item.find('h3').find('a')
                        title = movie_link_tag.text if movie_link_tag else 'N/A'
                        movie_id = movie_link_tag['href'].split('/')[2] if movie_link_tag else 'N/A'
                        year_tag = item.find('span', class_='lister-item-year text-muted unbold')
                        year = year_tag.text.strip('()') if year_tag else 'N/A'
                        genre_tag = item.find('span', class_='genre')
                        genre = genre_tag.text.strip() if genre_tag else 'N/A'
                        user_rating_tags = item.find_all('span', class_='ipl-rating-star__rating')
                        user_rating = user_rating_tags[1].text if len(user_rating_tags) > 1 else 'N/A'
                        directors_stars_text = item.find_all('p', class_='text-muted text-small')[1]
                        directors_stars_links = directors_stars_text.find_all('a')
                        director = directors_stars_links[0].text if directors_stars_links else 'N/A'
                        stars = ', '.join([star.text for star in directors_stars_links[1:]]) if len(directors_stars_links) > 1 else 'N/A'

                        all_data.append({
                            'Movie ID': movie_id,
                            'Title': title,
```

```python
                        'Year': year,
                        'Genre': genre,
                        'User Rating': user_rating,
                        'Director': director,
                        'Stars': stars,
                        'User ID': user_id
                    })
                except requests.exceptions.RequestException as e:
                    logging.error(f"Failed to retrieve ratings for user {user_id}: {e}")

        except requests.exceptions.RequestException as e:
            logging.error(f"Failed to process URL {url}: {e}")

    # Convert to DataFrame and save to CSV
    if all_data:
        df = pd.DataFrame(all_data)
        df.to_csv('imdb_user_ratingssss.csv', index=False)
        logging.info('Data scraping completed and saved to IMDb_User_Ratings.csv')
    else:
        logging.info('No data collected.')
```

Python

https://github.com/PVHai1111/Recommender-System/blob/main/data/imdb_user_rating_scrape.ipynb

Dữ liệu thu thập được:

Movie ID	Title	Year	Genre	User Rating	Director	Stars
tt9419884	Doctor Strange in the Multiverse of Madness	2022	Action, Adventure, Fantasy	5.0	Sam Raimi	Benedict Cumberbatch, Elizabeth Olsen, Chiwete
tt11466222	Jackass Forever	2022	Documentary, Action, Comedy	9.0	Jeff Tremaine	Johnny Knoxville, Steve-O, Chris Pontius, Dave E
tt10293406	The Power of the Dog	2021	Drama, Western	9.0	Jane Campion	Benedict Cumberbatch, Kirsten Dunst, Jesse Pler
tt1877830	The Batman	2022	Action, Crime, Drama	8.0	Matt Reeves	Robert Pattinson, Zoë Kravitz, Jeffrey Wright, Co
tt11286314	Don't Look Up	2021	Comedy, Drama, Sci-Fi	4.0	Adam McKay	Leonardo DiCaprio, Jennifer Lawrence, Meryl Str
tt10872600	Spider-Man: No Way Home	2021	Action, Adventure, Fantasy	7.0	Jon Watts	Tom Holland, Zendaya, Benedict Cumberbatch, J
tt1160419	Dune: Part One	2021	Action, Adventure, Drama	7.0	Denis Villeneuve	Timothée Chalamet, Rebecca Ferguson, Zendaya
tt10322274	Schumacher	2021	Documentary, Biography, Sport	5.0	Hanns-Bruno Kammertöns	Vanessa Nöcker, Michael Wech, Ross Brawn, Flav
tt0829482	Superbad	2007	Comedy	8.0	Greg Mottola	Michael Cera, Jonah Hill, Christopher Mintz-Plas
tt0266697	Kill Bill: Vol. 1	2003	Action, Crime, Thriller	9.0	Quentin Tarantino	Uma Thurman, David Carradine, Daryl Hannah, M
tt0387808	Idiocracy	2006	Adventure, Comedy, Sci-Fi	4.0	Mike Judge	Luke Wilson, Maya Rudolph, Dax Shepard, Terry
tt9770150	Nomadland	2020	Drama	9.0	Chloé Zhao	Frances McDormand, David Strathairn, Linda Ma
tt0118884	Contact	1997	Drama, Mystery, Sci-Fi	10.0	Robert Zemeckis	Jodie Foster, Matthew McConaughey, Tom Skerr
tt10618286	Mank	2020	Biography, Comedy, Drama	8.0	David Fincher	Gary Oldman, Amanda Seyfried, Lily Collins, Tom
tt8923484	Crip Camp	2020	Documentary, History	8.0	James Lebrecht	Nicole Newnham, James Lebrecht, Lionel Je'Woo

ar	Genre	User Rating	Director	Stars	User ID
22	Action, Adventure, Fantasy	5.0	Sam Raimi	Benedict Cumberbatch, Elizabeth Olsen, Chiwetel Ejiofor, Benedict Wong	ur86182727
22	Documentary, Action, Comedy	9.0	Jeff Tremaine	Johnny Knoxville, Steve-O, Chris Pontius, Dave England	ur86182727
21	Drama, Western	9.0	Jane Campion	Benedict Cumberbatch, Kirsten Dunst, Jesse Plemons, Kodi Smit-McPhee	ur86182727
22	Action, Crime, Drama	8.0	Matt Reeves	Robert Pattinson, Zoë Kravitz, Jeffrey Wright, Colin Farrell	ur86182727
21	Comedy, Drama, Sci-Fi	4.0	Adam McKay	Leonardo DiCaprio, Jennifer Lawrence, Meryl Streep, Cate Blanchett	ur86182727
21	Action, Adventure, Fantasy	7.0	Jon Watts	Tom Holland, Zendaya, Benedict Cumberbatch, Jacob Batalon	ur86182727
21	Action, Adventure, Drama	7.0	Denis Villeneuve	Timothée Chalamet, Rebecca Ferguson, Zendaya, Oscar Isaac	ur86182727
21	Documentary, Biography, Sport	5.0	Hanns-Bruno Kammertöns	Vanessa Nöcker, Michael Wech, Ross Brawn, Flavio Briatore, Luca Cordero di Montezemolo, David Coulthard	ur86182727
07	Comedy	8.0	Greg Mottola	Michael Cera, Jonah Hill, Christopher Mintz-Plasse, Bill Hader	ur86182727
03	Action, Crime, Thriller	9.0	Quentin Tarantino	Uma Thurman, David Carradine, Daryl Hannah, Michael Madsen	ur86182727
06	Adventure, Comedy, Sci-Fi	4.0	Mike Judge	Luke Wilson, Maya Rudolph, Dax Shepard, Terry Crews	ur86182727
20	Drama	9.0	Chloé Zhao	Frances McDormand, David Strathairn, Linda May, Gay DeForest	ur86182727
97	Drama, Mystery, Sci-Fi	10.0	Robert Zemeckis	Jodie Foster, Matthew McConaughey, Tom Skerritt, John Hurt	ur86182727
20	Biography, Comedy, Drama	8.0	David Fincher	Gary Oldman, Amanda Seyfried, Lily Collins, Tom Pelphrey	ur86182727
20	Documentary, History	8.0	James Lebrecht	Nicole Newnham, James Lebrecht, Lionel Je'Woodyard, Joseph O'Conor, Ann Cupolo Freeman	ur86182727

* Chú thích: Các trường lần lượt từ trái qua phải

- Movie ID: ID của các bộ phim trên imdb
- Title: Tên đầy đủ của phim
- Year: Năm phát hành
- Genre: Các tag thể loại của phim
- User Rating: Đánh giá của người dùng cho phim
- Director: Đạo diễn
- Stars: Các diễn viên được promote
- User ID: ID của người dùng

https://github.com/PVHai1111/Recommender-System/blob/main/data/combined_imdb_user_ratingss.csv

2.2. Tiền xử lý

Làm sạch dữ liệu: Chuyển id của các user và phim về id nội bộ; Chỉnh sửa lại lỗi hiển thị của trường năm phát hành; Quy chuẩn đánh giá về thang điểm 5; Loại bỏ các trường không cần thiết như Director

Phân tách các file csv phục vụ cho các chức năng khác nhau:

movies.csv

movieId	title	genres
1	Doctor Strange in the Multiverse of Madness	Action, Adventure, Fantasy
2	Jackass Forever	Documentary, Action, Comedy
3	The Power of the Dog	Drama, Western
4	The Batman	Action, Crime, Drama
5	Don't Look Up	Comedy, Drama, Sci-Fi
6	Spider-Man: No Way Home	Action, Adventure, Fantasy
7	Dune: Part One	Action, Adventure, Drama
8	Schumacher	Documentary, Biography, Sport
9	Superbad	Comedy
10	Kill Bill: Vol. 1	Action, Crime, Thriller
11	Idiocracy	Adventure, Comedy, Sci-Fi
12	Nomadland	Drama
13	Contact	Drama, Mystery, Sci-Fi
14	Mank	Biography, Comedy, Drama
15	Crip Camp	Documentary, History

https://github.com/PVHai1111/Recommender-System/blob/main/data/movies.csv

ratings.csv:

userId	movieId	rating
1	1	2.5
1	2	4.5
1	3	4.5
1	4	4.0
1	5	2.0
1	6	3.5
1	7	3.5
1	8	2.5
1	9	4.0
1	10	4.5
1	11	2.0
1	12	4.5

https://github.com/PVHai1111/Recommender-System/blob/main/data/ratings.csv

links.csv

movieId	imdbId
1	tt9419884
2	tt11466222
3	tt10293406
4	tt1877830
5	tt11286314
6	tt10872600
7	tt1160419
8	tt10322274
9	tt0829482
10	tt0266697
11	tt0387808
12	tt9770150

https://github.com/PVHai1111/Recommender-System/blob/main/data/links.csv

Code:

```
import pandas as pd

# Load the dataset
file_path = 'combined_imdb_user_ratingss.csv'
data = pd.read_csv(file_path)

# Display the first few rows of the dataset to understand its structure
data.head()
```

	Movie ID	Title	Year	Genre	User Rating	Director	Stars	User ID
0	tt9419884	Doctor Strange in the Multiverse of Madness	2022	Action, Adventure, Fantasy	5.0	Sam Raimi	Benedict Cumberbatch, Elizabeth Olsen, Chiwete...	ur86182727
1	tt11466222	Jackass Forever	2022	Documentary, Action, Comedy	9.0	Jeff Tremaine	Johnny Knoxville, Steve-O, Chris Pontius, Dave...	ur86182727
2	tt10293406	The Power of the Dog	2021	Drama, Western	9.0	Jane Campion	Benedict Cumberbatch, Kirsten Dunst, Jesse Ple...	ur86182727
3	tt1877830	The Batman	2022	Action, Crime, Drama	8.0	Matt Reeves	Robert Pattinson, Zoë Kravitz, Jeffrey Wright,...	ur86182727
4	tt11286314	Don't Look Up	2021	Comedy, Drama, Sci-Fi	4.0	Adam McKay	Leonardo DiCaprio, Jennifer Lawrence, Meryl St...	ur86182727

```
# Create a unique list of movies with a new inner ID
movies = data[['Movie ID', 'Title', 'Year', 'Genre']].drop_duplicates()
movies['movieId'] = range(1, len(movies) + 1)

# Create movies.csv
movies_csv = movies[['movieId', 'Title', 'Genre']]
movies_csv.columns = ['movieId', 'title', 'genres']
movies_csv.to_csv('movies.csv', index=False)

# Create links.csv
links_csv = movies[['movieId', 'Movie ID']]
links_csv.columns = ['movieId', 'imdbId']
links_csv.to_csv('links.csv', index=False)

# Normalize the user ratings to a 5-star scale
data['User Rating'] = data['User Rating'] / 2

# Create a unique list of users with a new inner ID
users = data[['User ID']].drop_duplicates()
users['userId'] = range(1, len(users) + 1)

# Merge the movies and users back into the original data to get inner IDs
ratings = data.merge(movies[['movieId', 'Movie ID']], on='Movie ID').merge(users, on='User ID')

# Create ratings.csv
ratings_csv = ratings[['userId', 'movieId', 'User Rating']]
ratings_csv.columns = ['userId', 'movieId', 'rating']
ratings_csv.to_csv('ratings.csv', index=False)

# Display the first few rows of the created CSV files to verify
movies_csv.head(), links_csv.head(), ratings_csv.head()
```

https://github.com/PVHai1111/Recommender-System/blob/main/data/data_cleaning_csv_split.ipynb

2.3. Dữ liệu mẫu

Để các mô hình được huấn luyện và đưa ra kết quả tốt hơn, bọn em sử dụng bộ dữ liệu mẫu gồm 9125 bộ phim, 100004 đánh giá của người dùng được thu thập vào ngày 16/10/2016

Nguồn: https://www.sundog-education.com/recsys/

Mô tả: Người dùng được chọn ngẫu nhiên và đều đánh giá trên 20 bộ phim. Có tổng cộng 18 thể loại của các bộ phim:

* Action

* Adventure

* Animation

* Children's

* Comedy

* Crime

* Documentary

* Drama

* Fantasy
* Film-Noir
* Horror
* Musical
* Mystery
* Romance
* Sci-Fi
* Thriller
* War
* Western

Ảnh và nguồn minh họa:

movieId	title	genres
1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy
2	Jumanji (1995)	Adventure|Children|Fantasy
3	Grumpier Old Men (1995)	Comedy|Romance
4	Waiting to Exhale (1995)	Comedy|Drama|Romance
5	Father of the Bride Part II (1995)	Comedy
6	Heat (1995)	Action|Crime|Thriller
7	Sabrina (1995)	Comedy|Romance
8	Tom and Huck (1995)	Adventure|Children
9	Sudden Death (1995)	Action
10	GoldenEye (1995)	Action|Adventure|Thriller
11	American President, The (1995)	Comedy|Drama|Romance
12	Dracula: Dead and Loving It (1995)	Comedy|Horror
13	Balto (1995)	Adventure|Animation|Children
14	Nixon (1995)	Drama
15	Cutthroat Island (1995)	Action|Adventure|Romance

```
userId,movieId,rating,timestamp
1,31,2.5,1260759144
1,1029,3.0,1260759179
1,1061,3.0,1260759182
1,1129,2.0,1260759185
1,1172,4.0,1260759205
1,1263,2.0,1260759151
1,1287,2.0,1260759187
1,1293,2.0,1260759148
1,1339,3.5,1260759125
1,1343,2.0,1260759131
1,1371,2.5,1260759135
1,1405,1.0,1260759203
1,1953,4.0,1260759191
1,2105,4.0,1260759139
1,2150,3.0,1260759194
1,2193,2.0,1260759198
1,2294,2.0,1260759108
1,2455,2.5,1260759113
1,2968,1.0,1260759200
1,3671,3.0,1260759117
2,10,4.0,835355493
2,17,5.0,835355681
2,39,5.0,835355604
2,47,4.0,835355552
2,50,4.0,835355586
2,52,3.0,835356031
```

movieId	imdbId	tmdbId
1	0114709	862
2	0113497	8844
3	0113228	15602
4	0114885	31357
5	0113041	11862
6	0113277	949
7	0114319	11860
8	0112302	45325
9	0114576	9091
10	0113189	710
11	0112346	9087
12	0112896	12110
13	0112453	21032
14	0113987	10858
15	0112760	1408

https://github.com/PVHai1111/Recommender-System/tree/main/ml-latest-small

2.4. Tích hợp

Nguồn:

https://github.com/PVHai1111/Recommender-System/blob/main/Framework/MovieLens.py

3. Framework đánh giá mô hình

Thông số sử dụng: MAE, RMSE, HR, cHR, rHR, ARHR, Coverage, Diversity, Novelty

```
def HitRate(topNPredicted, leftOutPredictions):
    hits = 0
    total = 0

    # For each left-out rating
    for leftOut in leftOutPredictions:
        userID = leftOut[0]
        leftOutMovieID = leftOut[1]
        # Is it in the predicted top 10 for this user?
        hit = False
        for movieID, predictedRating in topNPredicted[int(userID)]:
            if (int(leftOutMovieID) == int(movieID)):
                hit = True
                break
        if (hit) :
            hits += 1

        total += 1

    # Compute overall precision
    return hits/total
```

```
def CumulativeHitRate(topNPredicted, leftOutPredictions, ratingCutoff=0):
    hits = 0
    total = 0

    # For each left-out rating
    for userID, leftOutMovieID, actualRating, estimatedRating, _ in leftOutPredictions:
        # Only look at ability to recommend things the users actually liked...
        if (actualRating >= ratingCutoff):
            # Is it in the predicted top 10 for this user?
            hit = False
            for movieID, predictedRating in topNPredicted[int(userID)]:
                if (int(leftOutMovieID) == movieID):
                    hit = True
                    break
            if (hit) :
                hits += 1

            total += 1

    # Compute overall precision
    return hits/total
```

```
def RatingHitRate(topNPredicted, leftOutPredictions):
    hits = defaultdict(float)
    total = defaultdict(float)

    # For each left-out rating
    for userID, leftOutMovieID, actualRating, estimatedRating, _ in leftOutPredictions:
        # Is it in the predicted top N for this user?
        hit = False
        for movieID, predictedRating in topNPredicted[int(userID)]:
            if (int(leftOutMovieID) == movieID):
                hit = True
                break
        if (hit) :
            hits[actualRating] += 1

        total[actualRating] += 1

    # Compute overall precision
    for rating in sorted(hits.keys()):
        print (rating, hits[rating] / total[rating])
```

```
def AverageReciprocalHitRank(topNPredicted, leftOutPredictions):
    summation = 0
    total = 0
    # For each left-out rating
    for userID, leftOutMovieID, actualRating, estimatedRating, _ in leftOutPredictions:
        # Is it in the predicted top N for this user?
        hitRank = 0
        rank = 0
        for movieID, predictedRating in topNPredicted[int(userID)]:
            rank = rank + 1
            if (int(leftOutMovieID) == movieID):
                hitRank = rank
                break
        if (hitRank > 0) :
            summation += 1.0 / hitRank

        total += 1

    return summation / total
```

```
def UserCoverage(topNPredicted, numUsers, ratingThreshold=0):
    hits = 0
    for userID in topNPredicted.keys():
        hit = False
        for movieID, predictedRating in topNPredicted[userID]:
            if (predictedRating >= ratingThreshold):
                hit = True
                break
        if (hit):
            hits += 1

    return hits / numUsers
```

```python
def Diversity(topNPredicted, simsAlgo):
    n = 0
    total = 0
    simsMatrix = simsAlgo.compute_similarities()
    for userID in topNPredicted.keys():
        pairs = itertools.combinations(topNPredicted[userID], 2)
        for pair in pairs:
            movie1 = pair[0][0]
            movie2 = pair[1][0]
            innerID1 = simsAlgo.trainset.to_inner_iid(str(movie1))
            innerID2 = simsAlgo.trainset.to_inner_iid(str(movie2))
            similarity = simsMatrix[innerID1][innerID2]
            total += similarity
            n += 1

    S = total / n
    return (1-S)
```

```python
def Novelty(topNPredicted, rankings):
    n = 0
    total = 0
    for userID in topNPredicted.keys():
        for rating in topNPredicted[userID]:
            movieID = rating[0]
            rank = rankings[movieID]
            total += rank
            n += 1
    return total / n
```

Unit Test:

```
Algorithm   RMSE      MAE       HR        cHR       ARHR      Coverage  Diversity  Novelty
SVD         0.9034    0.6978    0.0298    0.0298    0.0112    0.9553    0.0445     491.5768
Random      1.4385    1.1478    0.0089    0.0089    0.0015    1.0000    0.0719     557.8365
```

https://github.com/PVHai1111/Recommender-System/blob/main/Framework/RecommenderMetrics.py

4. Content-based Filtering

Hàm tính sự tương đồng giữa 2 bộ phim:

```
def computeGenreSimilarity(self, movie1, movie2, genres):
    genres1 = genres[movie1]
    genres2 = genres[movie2]
    sumxx, sumxy, sumyy = 0, 0, 0
    for i in range(len(genres1)):
        x = genres1[i]
        y = genres2[i]
        sumxx += x * x
        sumyy += y * y
        sumxy += x * y

    return sumxy/math.sqrt(sumxx*sumyy)
```

Hệ số tương đồng dựa vào năm phát hành:

```
def computeYearSimilarity(self, movie1, movie2, years):
    diff = abs(years[movie1] - years[movie2])
    sim = math.exp(-diff / 10.0)
    return sim
```

Hàm tính đánh giá dự đoán:

```
def estimate(self, u, i):

    if not (self.trainset.knows_user(u) and self.trainset.knows_item(i)):
        raise PredictionImpossible('User and/or item is unkown.')

    # Build up similarity scores between this item and everything the user rated
    neighbors = []
    for rating in self.trainset.ur[u]:
        genreSimilarity = self.similarities[i,rating[0]]
        neighbors.append( (genreSimilarity, rating[1]) )

    # Extract the top-K most-similar ratings
    k_neighbors = heapq.nlargest(self.k, neighbors, key=lambda t: t[0])

    # Compute average sim score of K neighbors weighted by user ratings
    simTotal = weightedSum = 0
    for (simScore, rating) in k_neighbors:
        if (simScore > 0):
            simTotal += simScore
            weightedSum += simScore * rating

    if (simTotal == 0):
        raise PredictionImpossible('No neighbors')

    predictedRating = weightedSum / simTotal

    return predictedRating
```

Nhận vào 1 người dùng u và 1 bộ phim i, dict neighbors gồm điểm tương đồng giữa i và tất cả bộ phim u đã đánh giá. Chọn k hàng xóm gần nhất (có điểm tương đồng cao nhất) với i và đánh giá dự đoán của i sẽ là trung bình đánh giá của k bộ phim đó.

Điểm tương đồng được tính bằng công thức:

```
self.similarities[thisRating, otherRating] = genreSimilarity * yearSimilarity
```

Chỉ số đánh giá:

```
Algorithm  RMSE       MAE        HR         cHR        ARHR       Coverage   Diversity  Novelty
ContentKNN 0.9375     0.7263     0.0030     0.0030     0.0017     0.9285     0.5700     4567.1964
Random     1.4385     1.1478     0.0089     0.0089     0.0015     1.0000     0.0719     557.8365
```

Mẫu gợi ý cho người dùng ID = 85:

```
We recommend:
Presidio, The (1988) 3.841314676872932
Femme Nikita, La (Nikita) (1990) 3.839613347087336
Wyatt Earp (1994) 3.8125061475551796
Shooter, The (1997) 3.8125061475551796
Bad Girls (1994) 3.8125061475551796
The Hateful Eight (2015) 3.812506147555179
True Grit (2010) 3.812506147555179
Open Range (2003) 3.812506147555179
Big Easy, The (1987) 3.7835412549266985
Point Break (1991) 3.764158410102279
```

5. Neighborhood-based Collaborative Filtering

Mô hình chỉ output gợi ý, không dự đoán đánh giá, không cần tập kiểm tra

```
trainSet = data.build_full_trainset()
```

Xây dựng ma trận tương đồng sử dụng thuật toán KNNBasic của thư viện Suprise, chọn công thức tính độ tương đồng là cosine

```
sim_options = {'name': 'cosine',
               'user_based': True
               }

model = KNNBasic(sim_options=sim_options)
model.fit(trainSet)
simsMatrix = model.compute_similarities()
```

Thu thập các cặp chứa đối tượng đang xét và tạo thành dict gồm các tuples chứa id đối tượng và điểm tương đồng với đối tượng đang xét, sau đó lọc ra danh sách KNN (k nearest neighbors)

```
testUserInnerID = trainSet.to_inner_uid(testSubject)
similarityRow = simsMatrix[testUserInnerID]

similarUsers = []
for innerID, score in enumerate(similarityRow):
    if (innerID != testUserInnerID):
        similarUsers.append( (innerID, score) )

#kNeighbors = heapq.nlargest(k, similarUsers, key=lambda t: t[1])
kNeighbors = []
for rating in similarUsers:
    if rating[1] > 0.95:
        kNeighbors.append(rating)
```

```
testUserInnerID = trainSet.to_inner_uid(testSubject)

# Get the top K items we rated
testUserRatings = trainSet.ur[testUserInnerID]
#kNeighbors = heapq.nlargest(k, testUserRatings, key=lambda t: t[1])
kNeighbors = []
for rating in testUserRatings:
    if rating[1] > 4.0:
        kNeighbors.append(rating)
```

Tạo danh sách candidates, từ các trường của đối tượng tương đồng, đưa ra đánh giá dựa vào đánh giá của đối tượng tương đồng và chỉ số tương đồng với đối tượng đang xét. Nếu như 1 candidate được xét nhiều lần bởi nhiều đối tượng, sẽ có hệ số cao hơn. Theo dõi số lần xét bằng defaultdict của Python và xét giá trị của các candidate ban đầu là 0.

```
# Get the stuff they rated, and add up ratings for each item, weighted by user similarity
candidates = defaultdict(float)
for similarUser in kNeighbors:
    innerID = similarUser[0]
    userSimilarityScore = similarUser[1]
    theirRatings = trainSet.ur[innerID]
    for rating in theirRatings:
        candidates[rating[0]] += (rating[1] / 5.0) * userSimilarityScore

# Build a dictionary of stuff the user has already seen
watched = {}
for itemID, rating in trainSet.ur[testUserInnerID]:
    watched[itemID] = 1

# Get top-rated items from similar users:
pos = 0
for itemID, ratingSum in sorted(candidates.items(), key=itemgetter(1), reverse=True):
    if not itemID in watched:
        movieID = trainSet.to_raw_iid(itemID)
        print(ml.getMovieName(int(movieID)), ratingSum)
        pos += 1
        if (pos > 10):
            break
```

```
# Get similar items to stuff we liked (weighted by rating)
candidates = defaultdict(float)
for itemID, rating in kNeighbors:
    similarityRow = simsMatrix[itemID]
    for innerID, score in enumerate(similarityRow):
        candidates[innerID] += score * (rating / 5.0)

# Build a dictionary of stuff the user has already seen
watched = {}
for itemID, rating in trainSet.ur[testUserInnerID]:
    watched[itemID] = 1

# Get top-rated items from similar users:
pos = 0
for itemID, ratingSum in sorted(candidates.items(), key=itemgetter(1), reverse=True):
    if not itemID in watched:
        movieID = trainSet.to_raw_iid(itemID)
        print(ml.getMovieName(int(movieID)), ratingSum)
        pos += 1
        if (pos > 10):
            break
```

Kết quả gợi ý:

```
In [6]: runfile('C:/Users/Admin/Desktop/HUST/GR1/CollaborativeFiltering/SimpleUserCF.py',
wdir='C:/Users/Admin/Desktop/HUST/GR1/CollaborativeFiltering')
Reloaded modules: MovieLens
Computing the cosine similarity matrix...
Done computing similarity matrix.
Computing the cosine similarity matrix...
Done computing similarity matrix.
Shawshank Redemption, The (1994) 229.96836194898867
Silence of the Lambs, The (1991) 205.43102781808292
Schindler's List (1993) 176.08290718596766
Jurassic Park (1993) 170.4933268092533
Fargo (1996) 155.93550957059838
Back to the Future (1985) 154.04871891085054
Usual Suspects, The (1995) 147.8810875119617
Godfather, The (1972) 145.92718848855256
Fugitive, The (1993) 139.1874367528664
Seven (a.k.a. Se7en) (1995) 136.39157573589637
Independence Day (a.k.a. ID4) (1996) 132.79149035042283
```

```
In [5]: runfile('C:/Users/Admin/Desktop/HUST/GR1/CollaborativeFiltering/SimpleItemCF.py',
wdir='C:/Users/Admin/Desktop/HUST/GR1/CollaborativeFiltering')
Reloaded modules: MovieLens, ContentKNNAlgorithm, EvaluationData, RecommenderMetrics,
EvaluatedAlgorithm, Evaluator
Computing the cosine similarity matrix...
Done computing similarity matrix.
Computing the cosine similarity matrix...
Done computing similarity matrix.
Machete (2010) 9.233960269854148
47 Ronin (2013) 9.230262169999088
Hairspray (2007) 9.213647604612948
Johnny English (2003) 9.197174512772522
13 Assassins (JÃ»san-nin no shikaku) (2010) 9.188221279130666
It's Kind of a Funny Story (2010) 9.187494372165327
For a Few Dollars More (Per qualche dollaro in piÃ¹) (1965) 9.176606337889563
Megamind (2010) 9.172178852622746
3:10 to Yuma (2007) 9.168695550350016
Shall We Dance? (2004) 9.164158350575471
Killing Them Softly (2012) 9.160745847603803
```

IV. Đánh giá và nhận xét thực tế

1. Content-based

- Ưu điểm: Đơn giản

- Nhược điểm: Chậm, kết quả không phù hợp với thực tế do thể loại và năm phát hành không thật sự chính xác dưới góc nhìn của người dùng

2. User-based Collaborative Filtering

- Ưu điểm: Nhanh, kết quả hợp lý so với thực tế

- Nhược điểm: Không thật sự chính xác do người dễ thay đổi theo thời gian, cần nhiều dữ liệu từ người dùng đang xét để hoạt động hiệu quả, cần bộ dữ liệu lớn và đầy đủ để hoạt động hiệu quả

3. Item-based Collaborative Filtering

- Ưu điểm: Nhanh, kết quả hợp lý so với thực tế, chính xác hơn trên mặt lý thuyết so với user-based vì sản phẩm thường mang tính cố định hơn

- Nhược điểm: Cần bộ dữ liệu lớn và đầy đủ để hoạt động hiệu quả.

V. Phân công công việc

Thành viên	Nhiệm vụ	Đóng góp
Phạm Việt Hải (Leader)	- Quản lý tiến độ công việc - Thu thập dữ liệu test - Code framework đánh giá mô hình - Tích hợp mô hình item-based collaborative filtering	25%
Trần Bình Minh	- Thu thập dữ liệu - Tiền xử lý dữ liệu - Code framework tích hợp dữ liệu	25%

Nguyễn Lê Sơn	- Unit test framework đánh giá mô hình - Tích hợp mô hình user-based collaborative filtering - Viết báo cáo	25%
Nguyễn Văn Dũng	- Tích hợp mô hình content based - Viết báo cáo - Làm slide	25%

VI. Phụ lục

1. Khó khăn trong phát triển

Chúng em đã gặp khó khăn trong giai đoạn tìm và nghiên cứu các mô hình gợi ý, vì việc đánh giá hệ gợi ý rất khó khi không có sự xác nhận của một số lượng người như trong thực tế.

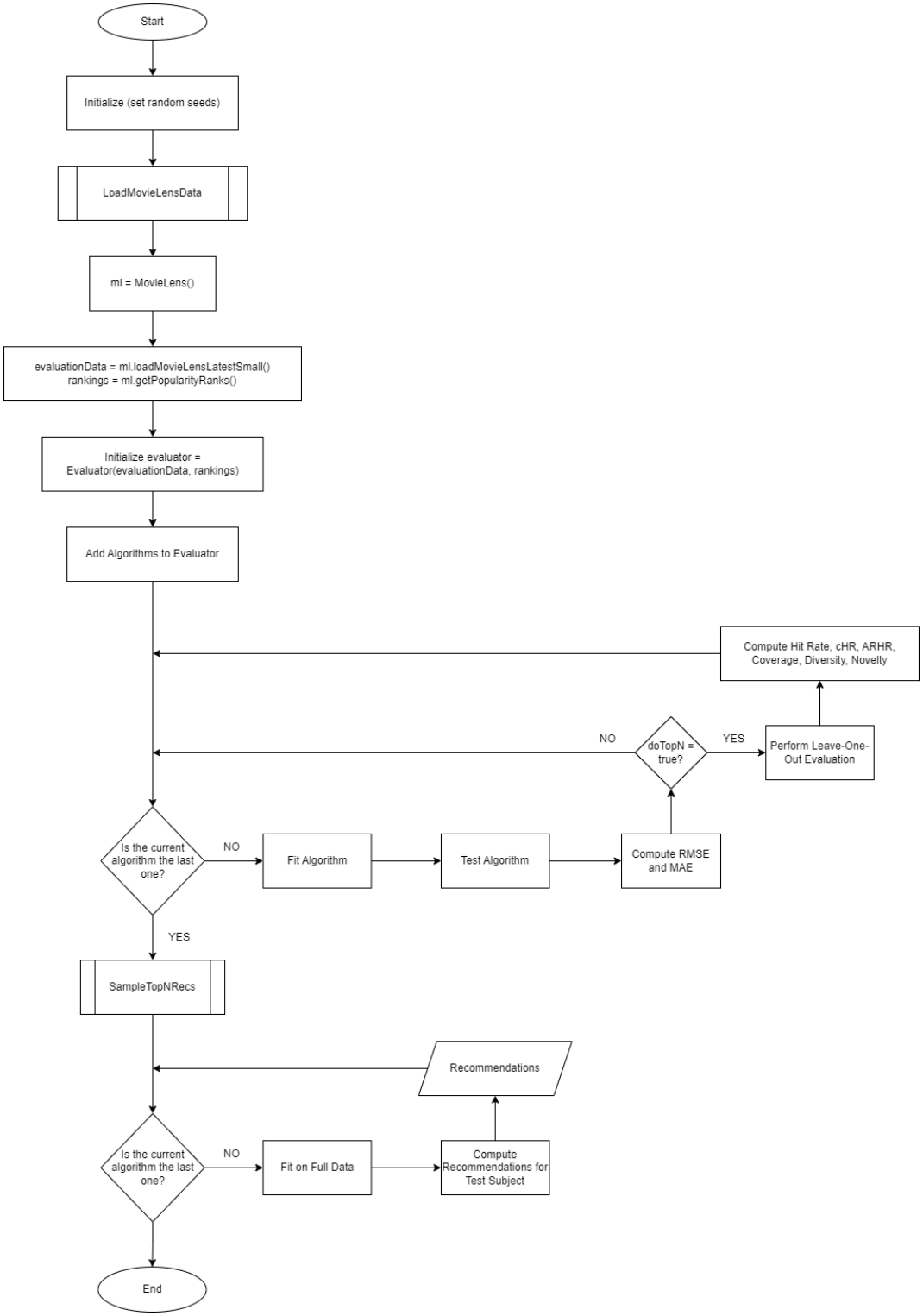
Giải quyết: Tham khảo ý kiến nhận xét từ thầy

2. Nguồn tham khảo

Document của thư viện: Surprise · A Python scikit for recommender systems. (surpriselib.com)

Cơ sở lý thuyết và xây dựng framework, mô hình: Building Recommender Systems with Machine Learning and AI: Getting Started - Sundog Education with Frank Kane (sundog-education.com)

3. Sơ đồ giải thuật cho hàm chạy chương trình gợi ý



4. Định hướng phát triển

- Thử và đánh giá các mô hình Matrix Factorization Methods, Deep Learning

- Xây dựng giao diện có khả năng đăng nhập, truy cập dữ liệu từ các nguồn hỗ trợ cho kiểm thử thực tế và sử dụng hệ thống