HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG

KHOA CÔNG NGHỆ THÔNG TIN

BÁO CÁO BÀI TẬP LỚN
NHẬP MÔN KHOA HỌC DỮ LIỆU
CHỦ ĐỀ : XÂY DỰNG HỆ THỐNG KHUYẾN NGHỊ XEM PHIM

Giảng viên : **Vũ Hoài Nam**
Nhóm lớp : **04**
Nhóm bài tập : **15**
Thành viên : **Nguyễn Văn Mạnh – B20DCCN428**
: **Phan Kế Vũ Hoàng – B20DCCN283**
: **Lâm Khánh Duy – B20DCCN151**

Hà Nội – 2023

LỜI CẢM ƠN

Đầu tiên, em xin gửi lời cảm ơn chân thành đến Học viện Công nghệ Bưu chính Viễn thông đã đưa môn học Nhập môn Khoa học dữ liệu vào chương trình giảng dạy. Đặc biệt, em xin gửi lời cảm ơn sâu sắc đến giảng viên bộ môn - Vũ Hoài Nam đã dạy dỗ, truyền đạt cho chúng em trong suốt thời gian học tập vừa qua để chúng em có đủ kiến thức, kinh nghiệm, có thể hoàn thành bài tập lớn này. Trong thời gian tham gia lớp học Nhập môn Khoa học dữ liệu của thầy, chúng em đã có thêm cho mình nhiều kiến thức bổ ích, tinh thần học tập hiệu quả, nghiêm túc.

Bộ môn Nhập môn Khoa học dữ liệu là môn học thú vị, vô cùng bổ ích và có tính thực tế cao. Đảm bảo cung cấp đủ kiến thức, gắn liền với nhu cầu thực tiễn của sinh viên đặc biệt trong thời đại bùng nổ công nghệ của trí tuệ nhân tạo ngày nay. Tuy nhiên, do vốn kiến thức còn nhiều hạn chế và việc áp dụng vào thực tế còn nhiều bỡ ngỡ. Mặc dù chúng em đã cố gắng hết sức nhưng chắc chắn bài báo cáo khó có thể tránh khỏi những thiếu sót, kính mong thầy xem xét và góp ý để hệ thống của chúng em được hoàn thiện hơn.

Chúng em xin chân thành cảm ơn!

Hà Nội, ngày 14 tháng 10 năm 2023

NHÓM TRƯỞNG

NGUYỄN VĂN MẠNH

MỤC LỤC

I. TỔNG QUAN VỀ HỆ THỐNG

1.1. Lý do chọn đề tài

- Cùng với sự phát triển bùng nổ của internet, các kênh tìm hiểu thông tin, giải trí, thương mại điện tử… cũng phát triển nhanh chóng và mạnh mẽ. Giờ đây, gần như chúng ta có thể tìm kiếm được mọi thứ trên internet, từ tài liệu, sách, truyện, phim, video đến mặt hàng, sản phẩm…

- Mỗi người, khi có một nhu cầu mua bán hoặc giải trí, sẽ có 2 cách để thực hiện. Thứ nhất, người đó có thể đến một địa điểm bán hàng hoặc vui chơi, nơi đó có những nhân viên có thể khuyến nghị về vấn đề của khách hàng hoặc khách hàng có thể thỏa thích xe kom qua những sản phẩm trên kệ hàng. Thứ 2, người có nhu cầu sử dụng internet để tìm kiếm.

- Người thích đọc sách, các trang web đọc sách online, mua bán sách sẽ là địa chỉ truy cập thường xuyên. Người dùng có thể tìm kiếm ra cuốn sách mình muốn đọc nhưng có thể, có những cuốn sách hay, phù hợp mà họ không biết đến.

- Người thích mua sắm có thể tìm kiếm những gì mình cần mua trên các trang bán hàng, nhưng có thể có những vật dụng khác mà nhất thời chưa nghĩ ra hoặc chưa được biết đến phù hợp với họ.

- Tương tự như vậy, người thích xem phim tìm kiếm các bộ phim ưa thích trên các trang xem phim trực tuyến, nhưng có thể có những bộ phim khác phù hợp với thể loại, sở thích của người dùng mà họ không biết đến.

- Điểm yếu so với cách thứ nhất của cách thứ 2 – sử dụng internet chính là ở các trang web truyền thống thiếu đi một nhân viên tư vấn cho khách hàng truy cập vào trang web của mình. Có nhiều giải pháp được đưa ra như lập một kênh trò chuyện trực tuyến giữa nhân viên bán hàng và người dùng, gọi điện thoại tư vấn. Như vậy, với một trang web có lượng truy cập lớn, số nhân viên trực cũng phải cần rất nhiều. Điều này đòi hỏi chi phí cao. Vì vậy, hệ thống khuyến nghị tự động ra đời, giải quyết được các vấn đề đó.

- Để người xem có thể chọn được một bộ phim ưng ý thì một sự gợi ý là rất quan trọng. Trong các rạp chiếu phim, những sự gợi ý từ các nhân viên bán vé sẽ làm tăng thêm nguồn thu nhập cho rạp chiếu phim. Do đó một hệ thống khuyến nghị trong một trang web xem phim online là vô cùng cần thiết

- Một hệ thống khuyến nghị tốt có thể đóng vai trò như một người trung gian hỗ trợ người xem chọn ra những bộ phim hay và đúng với yêu cầu người xem muốn. Bằng cách xác định mục đích và nhu cầu của người xem, hệ thống có thể đưa ra một danh sách các bộ phim gợi ý giúp cho người xem dễ dàng chọn lựa.

1.2. Giới thiệu về hệ thống khuyến nghị xem phim

1.2.1. Khái niệm về hệ thống khuyến nghị

- Hệ thống khuyến nghị (recommender system) là một dạng công cụ lọc thông tin (information filtering) cho phép suy diễn, dự đoán các sản phẩm, dịch vụ, nội dung mà người dùng có thể quan tâm dựa trên những thông tin thu thập được về người dùng, về các sản phẩm, dịch vụ, về các hoạt động, tương tác cũng như đánh giá của người dùng đối với các sản phẩm, dịch vụ trong quá khứ.

1.2.2. Lợi ích đem lại của hệ thống khuyến nghị

- Lý do cần có hệ thống khuyến nghị là bởi số lượng sản phẩm/dịch vụ/nội dung được cung cấp trực tuyến quá nhiều và người dùng khó tìm được thứ mình cần. Khi người dùng vào website cung cấp sản phẩm, hệ thống khuyến nghị sẽ trả về một danh sách ngắn các sản phẩm mà người dùng nhiều khả năng sẽ chọn, có thể bao gồm cả những thứ mà người đó không biết từ trước.

- Như vậy, hệ thống khuyến nghị giúp tiết kiệm thời gian, tăng tốc độ tìm kiếm và giúp người dùng truy cập tới nội dung họ quan tâm một cách dễ dàng hơn, đồng thời, gợi ý tới người dùng những đề xuất mới mà trước đây họ chưa từng biết đến. Với khả năng của hệ thống khuyến nghị, các doanh nghiệp sử dụng chúng để giới thiệu sản phẩm tới người tiêu dùng, giúp gia tăng doanh số nhờ các ưu đãi, sản phẩm, dịch vụ được khuyến nghị một cách cá nhân hóa, làm nâng cao trải nghiệm khách hàng. Điều này cải thiện lợi thế cạnh tranh của doanh nghiệp và giảm thiểu tỉ lệ khách hàng rời bỏ và đến với đối thủ cạnh tranh khi họ nhận thấy doanh nghiệp hiểu nhu cầu của họ và cung cấp cho họ những thứ họ muốn.

- Hệ thống khuyến nghị là thành phần không thể thiếu của các nền tảng trực tuyến cung cấp đa dạng các loại hình dịch vụ, từ các website thương mại điện tử tới nền tảng đào tạo trực tuyến. Theo McKinsey, 35% doanh thu của Amazon được tạo ra từ các tương tác với hệ thống khuyến nghị của hãng này. Một thống kê khác cũng cho thấy 75% thời lượng xem phim trên Netflix được thực hiện nhờ các khuyến nghị được cá nhân hoá

1.2.3. Các kỹ thuật chính trong hệ thống khuyến nghị

- Các hệ thống gợi ý thường sử dụng nhiều thuật toán khác nhau, về cơ bản, chúng ta có thể chia làm 2 nhóm lớn:

- Lọc dựa theo nội dung (Content-Based Filtering)
- Lọc cộng tác (Collaborative Filtering)

- Hệ thống gợi ý dựa trên nội dung (Content-Based Recommendation System) dựa vào thuộc tính của các sản phẩm, ví dụ như tên, nhà sản xuất, giá cả, chỉ số, mô tả... để đưa ra các sản phẩm tương tự nhau. Đặc điểm của phương pháp này

là việc xây dựng mô hình cho mỗi người dùng không phụ thuộc vào những người dùng khác mà thường phụ thuộc vào các đặc điểm của mặt hàng

- Hệ thống gợi ý dựa trên lọc cộng tác (Collaborative Filtering Recommendation System) dựa vào hành vi của những người dùng có xu hướng tương tự để gợi ý ra các sản phẩm cho người dùng. Phương pháp này có thể tận dụng được thông tin từ người dùng mà không phụ thuộc vào những đặc điểm của mặt hàng. Tuy nhiên cũng chính vì thế mà phương pháp này đòi hỏi một lượng lớn dữ liệu hiện có của người dùng để đưa ra những đề xuất chính xác.

1.3. Công nghệ sử dụng trong hệ thống

1.3.1. Ngôn ngữ lập trình Python

- Python là ngôn ngữ lập trình máy tính bậc cao thường được sử dụng để xây dựng trang web và phần mềm, tự động hóa các tác vụ và tiến hành phân tích dữ liệu. Python là ngôn ngữ có mục đích chung, nghĩa là nó có thể được sử dụng để tạo nhiều chương trình khác nhau và không chuyên biệt cho bất kỳ vấn đề cụ thể nào.

1.3.2. Ngôn ngữ lập trình JavaScript

JavaScript là ngôn ngữ lập trình được nhà phát triển sử dụng để tạo trang web tương tác. Từ làm mới bảng tin trên trang mạng xã hội đến hiển thị hình ảnh động và bản đồ tương tác, các chức năng của JavaScript có thể cải thiện trải nghiệm người dùng của trang web. Là ngôn ngữ kịch bản phía máy khách, JavaScript là một trong những công nghệ cốt lõi của World Wide Web. Ví dụ: khi duyệt internet, bất cứ khi nào bạn thấy quảng cáo quay vòng dạng hình ảnh, menu thả xuống nhấp để hiển thị hoặc màu sắc phần tử thay đổi động trên trang web cũng chính là lúc bạn thấy các hiệu ứng của JavaScript.

1.3.3. HTML

HTML là viết tắt của cụm từ Hypertext Markup Language (tạm dịch là Ngôn ngữ đánh dấu siêu văn bản). HTML được sử dụng để tạo và cấu trúc các thành phần trong trang web hoặc ứng dụng, phân chia các đoạn văn, heading, titles, blockquotes… và HTML không phải là ngôn ngữ lập trình.

1.3.4. CSS

CSS là chữ viết tắt của Cascading Style Sheets, nó là một ngôn ngữ được sử dụng để tìm và định dạng lại các phần tử được tạo ra bởi các ngôn ngữ đánh dấu (HTML). Nói ngắn gọn hơn là ngôn ngữ tạo phong cách cho trang web. Bạn có thể hiểu đơn giản rằng, nếu HTML đóng vai trò định dạng các phần tử trên website như việc tạo ra các đoạn văn bản, các tiêu đề, bảng,…thì CSS sẽ giúp

chúng ta có thể thêm style vào các phần tử HTML đó như đổi bố cục, màu sắc trang, đổi màu chữ, font chữ, thay đổi cấu trúc…

II. KIẾN THỨC CHUYÊN MÔN CỦA HỆ THỐNG

2.1. Giới thiệu về xử lý ngôn ngữ tự nhiên NLP

2.1.1. Khái niệm về xử lý ngôn ngữ tự nhiên NLP

- Xử lý ngôn ngữ tự nhiên (NLP) là một công nghệ máy học, cung cấp cho máy tính khả năng diễn giải, tương tác và hiểu được ngôn ngữ của con người. Các tổ chức ngày nay có khối lượng lớn dữ liệu thoại và văn bản từ nhiều kênh liên lạc khác nhau như email, tin nhắn văn bản, bảng tin trên mạng xã hội, tệp video, tệp âm thanh và nhiều hơn nữa. Họ sử dụng phần mềm NLP để tự động xử lý dữ liệu này, phân tích ý định hoặc cảm xúc trong tin nhắn và phản hồi bằng người thật theo thời gian thực.

2.1.2. Các bước trong xử lý ngôn ngữ tự nhiên NLP.

- Xử lý tự nhiên bao gồm 5 bước:

1. Lexical Analysis: Chúng ta phải phân tích cấu trúc của từ. Các tập hợp gồm các từ và cụm từ trong một ngôn ngữ được gọi là một bộ từ vựng của một ngôn ngữ.

2. Phân tích cú pháp (Syntactic Analysis): Chúng ta sẽ sử dụng phân tích cú pháp để phân tích các từ và cụm từ. Điều này đồng nghĩa với việc bạn phải sắp xếp các từ theo một cách cụ thể, trong đó, các từ phải có mối quan hệ với nhau.

3. Phân tích ngữ nghĩa: Bạn có thể hiểu đây là công đoạn mô tả nghĩa của một từ sao cho có nghĩa. Trong miền tác vụ, chúng sẽ ánh xạ cấu trúc cú pháp và đối tượng.

4. Tích hợp bài giảng: Trong bước này, nghĩa của một câu bất kỳ đều sẽ phụ thuộc vào nghĩa của câu trước.

5. Phân tích thực dụng: Trong bước này, dữ liệu được diễn giải dựa trên ý nghĩa thực sự của nó. Mặc dù vậy, chúng ta cần phải phân tích ngôn ngữ kỹ hơn theo thực tế, điều này đòi hỏi kiến thức trong thế giới thực.

2.1.3. Các ứng dụng của xử lý ngôn ngữ tự nhiên NLP.

2.1.3.1.Liên lạc

- Về cơ bản, máy tính là một phương tiện để giao tiếp với người dùng. Bên cạnh đó, để học một ngôn ngữ mới, chúng tôi không thể ép buộc người dùng (Mặc dù đối với người dùng bình thường thì nó khá quan trọng nhất). Ví dụ trong trường hợp của người quản lý và trẻ em, các quản lý viên không có thời gian để học các kỹ năng tương tác mới.

- Ngôn ngữ tự nhiên sở hữu một kho thông tin khổng lồ, và để xem các thông tin này, chúng ta phải truy cập thông qua máy tính. Hiện nay, con người đang tạo ra thông tin liên tục, dưới dạng sách, dữ liệu kinh doanh hoặc thậm chí là báo cáo từ Chính Phủ.

- Trong hệ thống xử lý ngôn ngữ tự nhiên, AI được tích hợp vào một cách rõ ràng

- Bất kỳ hệ thống hiểu ngôn ngữ tự nhiên cũng có 3 khía cạnh chính

2.3.1.2. Cú pháp

- Về cơ bản, chúng ta sẽ sử dụng cú pháp để mô tả hình thức của ngôn ngữ. Ngoài ra, ngữ pháp được sử dụng để chỉ định nó. Chúng ta sẽ sử dụng NLP cho các ngôn ngữ logic và chương trình máy tính AI. Ngôn ngữ này sẽ phức tạp hơn đôi chút so với các ngôn ngữ chính thức khác.

2.3.1.3. Ngữ nghĩa học

- Ý nghĩa của lời nói cũng khá tương đồng với ngữ nghĩa. Nếu chúng ta muốn xây dựng sự hiểu biết này cho hệ thống, ta có thể sử dụng các lý thuyết ngữ nghĩa chung đã có.

2.3.1.4. Ngữ dụng học

- Chúng ta sẽ sử dụng thành phần này để giải thích ý nghĩa của các lời nói trong thế giới thực tế.

2.2. Tổng quát về TF-IDF

2.2.1. Khái niệm về TF-IDF

- TF-IDF (viết tắt của term frequency – inverse document frequency) là một phương thức thống kê thường được sử dụng trong mảng truy xuất thông tin (information retrieval) và khai phá dữ liệu văn bản (text mining) để đánh giá mức độ quan trọng của một cụm từ đối với một tài liệu cụ thể trong một tập hợp bao gồm nhiều tài liệu. Khái niệm này đã xuất hiện từ rất sớm trong các lĩnh vực nghiên cứu khác nhau, chẳng hạn như ngôn ngữ học (linguistics) và cấu trúc thông tin (information architecture), nhờ vào khả năng hỗ trợ xử lý nhiều tập tài liệu với số lượng lớn trong một khoảng thời gian ngắn.

- Các máy tìm kiếm thường sử dụng các biến số khác nhau của thuật toán TF-IDF như là một phần trong cơ chế xếp hạng. Bằng cách gán cho các tài liệu một mức điểm số về độ liên quan (relevance score), chúng có thể đưa ra các kết quả tìm kiếm thích hợp chỉ trong phần triệu giây.

2.2.2. Cơ chế hình thành TF-IDF

- Đầu tiên, chúng ta cần biết rằng chỉ số TF-IDF có thể được tính toán dựa theo công thức: TF-IDF = TF x IDF

Trong đó:

- Đại lượng TF(Tần suất thuật ngữ) : Số lần một thuật ngữ xuất hiện trong mỗi tài liệu chia cho tổng số từ trong tài liệu
- Đại lượng IDF(Tần suất tài liệu nghịch đảo): Giá trị nhật ký của tần suất tài liệu. Tần suất tài liệu là tổng số tài liệu của một thuật ngữ

- Term Frequency (TF): Tần suất xuất hiện của một từ trong một văn bản

Hình thức đơn giản nhất của TF chính là đếm số lần sử dụng một từ khóa trên một page hoặc một tài liệu nào đó. Tuy nhiên, lúc này sẽ có một vấn đề xảy ra: giả sử có một trang sử dụng một từ khóa 10 lần thì sẽ được xem là có giá trị về độ liên quan hơn là một trang chỉ sử dụng từ khóa 1 lần. Điều này trên thực tế lại không đúng. Và vì lý do đó, chúng ta sẽ hạn chế sự sai lệch này lại, về mặt toán học thì giải pháp chính là sử dụng một hàm dưới tuyến tính (sublinear function) để phản ánh chính xác hơn, cụ thể ở đây là hàm lô-ga-rít (log).

$$TF = \frac{1 + \log(\text{Keyword Count})}{\log(\text{Word Count})}$$

- Inverse Document Frequency (IDF): tần suất nghịch của một cụm từ trong một tập hợp gồm nhiều tài liệu

Chỉ số này thể hiện giá trị thực của một từ khóa cụ thể. Nó đo lường tỷ lệ giữa tổng số lượng tài liệu với số lượng các tài liệu có chứa từ khóa đó. Công thức để tính IDF như sau (tương tự như TF, bạn cũng có thể tìm thấy một vài cách tính biến thể khác nhau của IDF, nhưng đều được xây dựng dựa trên bản chất của nó):

$$IDF = \log\left(1 + \frac{\text{Total Documents}}{\text{Documents with Keyword}}\right)$$

Như vậy có thể thấy, không giống như đại lượng mật độ từ khóa chỉ phản ánh mức độ "nhồi nhét" một từ khóa cụ thể vào trong văn bản, TF-IDF thể hiện vai trò là một chỉ số toàn diện và nâng cao hơn, phản ánh mức độ quan trọng của một từ khóa cụ thể đối với một trang cụ thể. Nó giúp làm giảm đi sự lấn át của những

từ và cụm từ không quan trọng, trong khi những cụm từ có ý nghĩa và xuất hiện không nhiều sẽ được nâng cao mức độ quan trọng hơn.

III. XÂY DỰNG HỆ TRI THỨC CỦA HỆ THỐNG

3.1. Dữ liệu sử dụng trong hệ thống

- Dataset bao gồm 10000 bộ phim hàng đầu từ IMDb được sản xuất từ năm 1915 đến 2023. Dữ liệu được lấy từ trang web của IMDb bằng kỹ thuật web scraping

- Thu thập dữ liệu:

- Import các thư viện cần thiết

```
import pandas as pd
import requests
from bs4 import BeautifulSoup
import numpy as np
from time import sleep
import re
import copy
```

- Tạo các biến phục vụ cho việc chứa dữ liệu

```
[ ]  Placeholder=None#To handle missing values when scraping
     Headers={'User-Agent':'Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:84.0) Gecko/20100101 Firefox/84.0'}

[ ]  movie_name=[] # tên bộ phim
     movie_year=[] # năm phát hành
     movie_rating=[] # xếp hạng phim
     movie_votes=[] # lượt vote
     movie_metascore=[]
     Gross_income=[] # tổng thu nhập
     Movie_Runtime=[] # thời lượng phim
     Genre=[] # thể loại
     Age_rated=[] # độ tuổi xếp hạng
     Directors=[] # đạo diễn
     Stars=[] # diễn viên
     description=[] # miêu tả
```

- Scraping

```
import time
start=time.time()
pages=np.arange(1,10000,50)
for page in pages:
    url=f'https://www.imdb.com/search/title/?title_type=feature&num_votes=10000,&sort=user_rating,desc&start={str(page)}'
    try:
        webpage = requests.get(url, headers=Headers).text
    except requests.exceptions.ConnectionError:
        print(f"Connection error occurred. Retrying after 5 sec")
        time.sleep(5)  # Đợi 5 giây trước khi thử lại
        continue  # Bỏ qua phần còn lại của vòng lặp và chuyển sang trang tiếp theo

    # webpage=requests.get(url,headers=Headers).text
    soup=BeautifulSoup(webpage,'lxml')
    movie_data=soup.find_all("div",class_="lister-item mode-advanced")
    print(f"Scraping URL: {url}")
```

```
    for movies in movie_data:
        # tên bộ phim
        movie_name.append(movies.h3.a.text.strip())

        # năm phát hành
        years=movies.h3.find("span",class_='lister-item-year text-muted unbold').text.strip("()")
        movie_year.append(years)



        # xếp hạng phim
        rate=movies.find('div', class_ = 'inline-block ratings-imdb-rating').text.replace('\n', '')
        movie_rating.append(rate)

        #Meta Score
        meta = movies.find('span', class_ = 'metascore').text.replace(' ', '') if movies.find('span', class_ = 'metascore') else Placeholder
        movie_metascore.append(meta)

        # tổng thu nhập và lượt vote

        value = movies.find_all('span', attrs = {'name': "nv"})

        vote = value[0].text
        gross = value[1].text if len(value)>1 else Placeholder#xử lý giá trị còn thiếu
        Gross_income.append(gross)
        movie_votes.append(vote)


        #Runtime,Genre and Certificate

        runtime_span = movies.find("span", class_="runtime")
        genre_span = movies.find("span", class_="genre")
        certificate_span = movies.find("span", class_="certificate")

            # Xử lý thông tin thời gian chạy bị thiếu
        if runtime_span:
            runtime_text = runtime_span.text.strip('min')
        else:
            runtime_text = Placeholder

            # Xử lý thông tin thể loại bị thiếu
        if genre_span:
            genre_text = genre_span.text.strip().split(',')
        else:
            genre_text = [Placeholder]
```

```
             # Xử lý thông tin chứng chỉ bị thiếu
        if certificate_span:
            certificate_text = certificate_span.text.strip()
        else:
            certificate_text = Placeholder


        #miêu tả

        Movie_Runtime.append(runtime_text)
        Genre.append(genre_text)
        Age_rated.append(certificate_text)

        describe = movies.find_all('p', class_ = 'text-muted')
        description_ = describe[1].text.replace('\n', '') if len(describe) >1 else Placeholder
        description.append(description_)


        #Director and Stars using Regex
        Cast = movies.find('p', class_='').text.strip().replace('\n', '')

             # Extract directors and stars using regex patterns
        directors_pattern = r"Directors?:(.*?)(?:\||Stars:|$)"
        stars_pattern = r"Stars:(.*?)(?:\||$)"

             # Extract directors using regex
        directors_match = re.search(directors_pattern, Cast, re.DOTALL)
        directors = [director.strip() for director in directors_match.group(1).split(',')] if directors_match else []

             # Extract stars using regex
        stars_match = re.search(stars_pattern, Cast, re.DOTALL)
        stars = [star.strip() for star in stars_match.group(1).split(',')] if stars_match else []

        Directors.append(directors)
        Stars.append(stars)

end=time.time()
total=end-start
print(total)
```

```
Scraping URL: https://www.imdb.com/search/title/?title_type=feature&num_votes=10000,&sort=user_rating,desc&start=1
Scraping URL: https://www.imdb.com/search/title/?title_type=feature&num_votes=10000,&sort=user_rating,desc&start=51
Scraping URL: https://www.imdb.com/search/title/?title_type=feature&num_votes=10000,&sort=user_rating,desc&start=101
Scraping URL: https://www.imdb.com/search/title/?title_type=feature&num_votes=10000,&sort=user_rating,desc&start=151
Scraping URL: https://www.imdb.com/search/title/?title_type=feature&num_votes=10000,&sort=user_rating,desc&start=201
Scraping URL: https://www.imdb.com/search/title/?title_type=feature&num_votes=10000,&sort=user_rating,desc&start=251
Scraping URL: https://www.imdb.com/search/title/?title_type=feature&num_votes=10000,&sort=user_rating,desc&start=301
Scraping URL: https://www.imdb.com/search/title/?title_type=feature&num_votes=10000,&sort=user_rating,desc&start=351
```

- Tạo dataframe chứa dữ liệu

```
[ ] Dataset = pd.DataFrame({
        "Movie Name": movie_name, "Year of Release" : movie_year, "Run Time in minutes": Movie_Runtime,"Movie Rating": movie_rating,"Votes":movie_votes,"MetaScore":movie_metascore,
        "Gross":Gross_income,"Genre":Genre,"Certification":Age_rated,"Director":Directors,"Stars":Stars,"Description":description
    })

[ ] data = pd.DataFrame(columns = Dataset.columns, data = copy.deepcopy(Dataset.values))

    data['Year of Release']=data['Year of Release'].str.replace(r'\D', '').astype(int)#Removes all the unwanted values with regex \D  and changes them into int
    data.Votes=data.Votes.str.replace(',','').astype(int)
    data['Movie Rating']=data['Movie Rating'].astype(float)
    data['Run Time in minutes']=data['Run Time in minutes'].astype(int)

    for index,value in enumerate(data.Gross):
      if value!=None:
        data.Gross[index]=value.replace('$','').replace('M','')
    data.Gross=pd.to_numeric(data.Gross)
    data.Gross=data.Gross*1000000

    data.MetaScore=pd.to_numeric(data.MetaScore)

    <ipython-input-311-497fe3c909c0>:1: FutureWarning: The default value of regex will change from True to False in a future version.
      data['Year of Release']=data['Year of Release'].str.replace(r'\D', '').astype(int)#Removes all the unwanted values with regex \D  and changes them into int
    <ipython-input-311-497fe3c909c0>:8: SettingWithCopyWarning:
    A value is trying to be set on a copy of a slice from a DataFrame

    See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
      data.Gross[index]=value.replace('$','').replace('M','')
```

- Thông tin dữ liệu sau khi thu thập

```
data.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 12 columns):
 #   Column               Non-Null Count  Dtype
---  ------               --------------  -----
 0   Movie Name           10000 non-null  object
 1   Year of Release      10000 non-null  int64
 2   Run Time in minutes  10000 non-null  int64
 3   Movie Rating         10000 non-null  float64
 4   Votes                10000 non-null  int64
 5   MetaScore            7974 non-null   float64
 6   Gross                7085 non-null   float64
 7   Genre                10000 non-null  object
 8   Certification        9631 non-null   object
 9   Director             10000 non-null  object
 10  Stars                10000 non-null  object
 11  Description          10000 non-null  object
dtypes: float64(3), int64(3), object(6)
memory usage: 937.6+ KB

[ ]

[ ] data['Description']=data['Description'].apply(lambda x:x.split())
```

```
[ ] data['Description']=data['Description'].apply(lambda x:x.split())

data
```

	Movie Name	Year of Release	Run Time in minutes	Movie Rating	Votes	MetaScore	Gross	Genre	Certification	Director	Stars	Description
0	The Shawshank Redemption	1994	142	9.3	2804443	82.0	28340000.0	[Drama]	R	[Frank Darabont]	[Tim Robbins, Morgan Freeman, Bob Gunton, Will...	[Over, the, course, of, several, years,, two, ...
1	The Godfather	1972	175	9.2	1954174	100.0	134970000.0	[Crime, Drama]	R	[Francis Ford Coppola]	[Marlon Brando, Al Pacino, James Caan, Diane K...	[Don, Vito, Corleone,, head, of, a, mafia, fam...
2	Ramayana: The Legend of Prince Rama	1993	135	9.2	12995	NaN	NaN	[Animation, Action, Adventure]	PG	[Ram Mohan, Yûgô Sakô, Koichi Saski]	[Arun Govil, Nikhil Kapoor, Edie Mirman, Rael ...	[An, anime, adaptation, of, the, Hindu, epic, ...
3	The Chaos Class	1975	87	9.2	42231	NaN	NaN	[Comedy, Drama]	None	[Ertem Egilmez]	[Kemal Sunal, Münir Özkul, Halit Akçatepe, Tar...	[Lazy,, uneducated, students, share, a, very, ...
4	The Dark Knight	2008	152	9.0	2786129	84.0	534860000.0	[Action, Crime, Drama]	PG-13	[Christopher Nolan]	[Christian Bale, Heath Ledger, Aaron Eckhart, ...	[When, the, menace, known, as, the, Joker, wre...
...	...	...	...	...	...	...	...	...	...	...	...	...
9995	Golmaal Again	2017	140	4.9	10183	NaN	1010000.0	[Action, Comedy, Fantasy]	Not Rated	[Rohit Shetty]	[Ajay Devgn, Arshad Warsi, Tabu, Shreyas Talpade]	[The, gang, encounters, with, some, spiritual,...
9996	Not to Forget	2021	84	4.9	11191	NaN	NaN	[Drama]	None	[Valerio Zanoli]	[Karen Grassle, Louis Gossett Jr., Tatum O'Nea...	[After, a, lifetime, of, scams,, a, self-cente...
9997	Housefull 3	2016	140	4.9	10308	NaN	1160000.0	[Action, Comedy, Romance]	Not Rated	[Sajid, Farhad Samji]	[Akshay Kumar, Abhishek Bachchan, Riteish Desh...	[A, father, doesn't, want, his, three, daughte...
9998	A.I. Rising	2018	85	4.9	11187	NaN	NaN	[Drama, Romance, Sci-Fi]	R	[Lazar Bodroza]	[Sebastian Cavazza, Stoya, Marusa Majer, Kirst...	[An, intimate, relationship, between, a, human...
9999	The Last Boy	2019	87	4.9	10324	NaN	NaN	[Fantasy, Mystery, Sci-Fi]	None	[Perry Bhandal]	[Luke Goss, Flynn Allen, Peter Guinness, Matil...	[With, the, world's, end, imminent,, a, dying,...

10000 rows × 12 columns

- Dữ liệu bao gồm các nội dung:

Movie Name: Tên phim

Year of Release: Năm phát hành của bộ phim

Run Time in minutes: Thời gian bộ phim diễn ra

Movie Rating: Đánh giá của người dùng IMDb.

Votes: Số phiếu bầu trên IMDb.

MetaScore: Đánh giá metascore.

Gross: Tổng thu nhập của bộ phim

Genre: (Các) Thể loại của phim.

Certification: Chứng nhận hoặc xếp hạng của bộ phim.

Director: (các) đạo diễn của bộ phim.

Stars: Dàn diễn viên chính hoặc các diễn viên của phim.

Description: Một bản tóm tắt hoặc cốt truyện ngắn gọn của bộ phim.

- Link dataset: https://www.kaggle.com/datasets/willianoliveiragibin/10000-data-about-movies-1915-2023/data

3.2. Độ tương tự Cosine (Cosine Similarity)

3.2.1. Khái niệm về độ tương tự Cosine.

Độ tương tự cosin là một cách đo độ tương tự (measure of similarity) giữa hai vectơ khác không của một không gian tích vô hướng. Độ tương tự này được định nghĩa bằng giá trị cosine của góc giữa hai vectơ, và cũng là tích vô hướng của cùng các vectơ đơn vị để cả hai đều có chiều dài 1. Giá trị cosine của 0° là 1, và bé hơn 1 với bất kỳ góc nào trong khoảng các radian (0, π].

3.2.2. Công thức tính độ tương tự Cosine

Cho hai vectơ chứa các thuộc tính, *A* và *B*, độ tương tự cosine, cos(θ), được thể hiện bằng tích vô hướng và độ lớn là

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

với Ai và Bi là thành phần của vector A và B tương ứng

Độ tương tự có giá trị −1 có nghĩa là trái nghĩa hoàn toàn, với giá trị 1 nghĩa là giống nhau hoàn toàn, với 0 có nghĩa là trực giao hay tương quan (decorrelation), trong khi các giá trị ở giữa biểu thị sự giống nhau hoặc không giống nhau ở mức trung gian.

3.2.3. Độ tương tự Cosine trong học máy

Độ tương tự cosine được sử dụng để tìm ra điểm tương đồng giữa hai tài liệu. Nó thực hiện điều này bằng cách tính điểm tương tự giữa các vectơ, được thực hiện bằng cách tìm các góc giữa chúng. Phạm vi tương đồng nằm trong khoảng từ 0 đến 1. Nếu giá trị điểm tương đồng giữa hai vectơ là 1, điều đó có nghĩa là có sự tương đồng lớn hơn giữa hai vectơ.

Mặt khác, nếu giá trị điểm tương đồng giữa hai vectơ là 0 thì có nghĩa là không có sự tương đồng giữa hai vectơ. Khi điểm tương đồng bằng 1 thì góc giữa hai vectơ là 0 và khi điểm tương đồng là 0 thì góc giữa hai vectơ là 90 độ.

Trong các ứng dụng học máy, kỹ thuật này chủ yếu được sử dụng trong các hệ thống khuyến nghị để tìm ra điểm tương đồng giữa mô tả của hai sản phẩm để chúng tôi có thể đề xuất sản phẩm giống nhất cho người dùng nhằm mang lại trải nghiệm người dùng tốt hơn.

3.2.4. Cosine Similarity trong Python

Để triển khai nó bằng Python, chúng ta có thể sử dụng phương thức "cosine_similarity" do scikit-Learn cung cấp.
Ý tưởng là tạo hai mảng và sau đó triển khai phương thức "cosine_similarity" được cung cấp trong thư viện Scikit-Learn để tìm ra điểm tương đồng giữa chúng. Dưới đây là cách tính Độ tương tự Cosine bằng Python:

```
import numpy as np

import pandas as pd

from sklearn.metrics.pairwise import cosine_similarity



a = np.array([10, 5, 15, 7, 5])

b = np.array([5, 10, 17, 5, 3])

cosine = cosine_similarity(a.reshape(1, -1), b.reshape(1, -1))

print(cosine)
```

Kết quả:

```
[[0.92925111]]
```

Vì vậy, điểm tương tự nhận được giữa hai mảng (a và b) là 0,92 (xấp xỉ), gần bằng 1. Vì vậy, chúng ta có thể nói rằng các mảng giống nhau ở một mức độ nào đó.

IV. XÂY DỰNG HỆ THỐNG

4.1. Xử lý dữ liệu

- Đọc dữ liệu:

```
import numpy as np
import pandas as pd
from sklearn.feature_extraction import text
from sklearn.metrics.pairwise import cosine_similarity

data = pd.read_csv("data.csv")
data.head()
```

✓ 6.0s Python

	Unnamed: 0	Movie Name	Year of Release	Run Time in minutes	Movie Rating	Votes	MetaScore	Gross	Genre	Certification	Director	Stars	Description
0	0	The Shawshank Redemption	1994	142	9.3	2804443	82.0	28340000.0	['Drama']	R	['Frank Darabont']	['Tim Robbins', 'Morgan Freeman', 'Bob Gunton'...	['Over', 'the', 'course', 'of', 'several', 'ye...
1	1	The Godfather	1972	175	9.2	1954174	100.0	134970000.0	['Crime', ' Drama']	R	['Francis Ford Coppola']	['Marlon Brando', 'Al Pacino', 'James Caan', '...	['Don', 'Vito', 'Corleone,', 'head', 'of', 'a'...
2	2	Ramayana: The Legend of Prince Rama	1993	135	9.2	12995	NaN	NaN	['Animation', ' Action', ' Adventure']	PG	['Ram Mohan', 'Yûgô Sakô', 'Koichi Saski']	['Arun Govil', 'Nikhil Kapoor', 'Edie Mirman',...	['An', 'anime', 'adaptation', 'of', 'the', 'Hi...
3	3	The Chaos Class	1975	87	9.2	42231	NaN	NaN	['Comedy', ' Drama']	NaN	['Ertem Egilmez']	['Kemal Sunal', 'Münir Özkul', 'Halit Akçatepe...	['Lazy,', 'uneducated', 'students', 'share', '...
4	4	The Dark Knight	2008	152	9.0	2786129	84.0	534860000.0	['Action', ' Crime', ' Drama']	PG-13	['Christopher Nolan']	['Christian Bale', 'Heath Ledger', 'Aaron Eckh...	['When', 'the', 'menace', 'known', 'as', 'the'...

- Chọn lọc dữ liệu:

Để sử dụng dữ liệu cho mục đích giới thiệu phim mới, chúng ta sẽ quan tâm đến 3 cột Moive Name, Description, Genre

```
data = data[["Movie Name", "Description", "Genre"]]
data.head()
```

✓ 0.0s

	Movie Name	Description	Genre
0	The Shawshank Redemption	['Over', 'the', 'course', 'of', 'several', 'ye...	['Drama']
1	The Godfather	['Don', 'Vito', 'Corleone,', 'head', 'of', 'a'...	['Crime', ' Drama']
2	Ramayana: The Legend of Prince Rama	['An', 'anime', 'adaptation', 'of', 'the', 'Hi...	['Animation', ' Action', ' Adventure']
3	The Chaos Class	['Lazy,', 'uneducated', 'students', 'share', '...	['Comedy', ' Drama']
4	The Dark Knight	['When', 'the', 'menace', 'known', 'as', 'the'...	['Action', ' Crime', ' Drama']

- Thông tin dữ liệu

```
data.info()
```

✓ 0.0s

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 3 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   Movie Name   10000 non-null  object
 1   Description  10000 non-null  object
 2   Genre        10000 non-null  object
dtypes: object(3)
memory usage: 234.5+ KB
```

Như vậy dữ liệu đã sẵn sàng để sử dụng

- Import thư viện để xử lý

```
import nltk
import re
nltk.download('stopwords')
stemmer = nltk.SnowballStemmer("english")
from nltk.corpus import stopwords
import string
stopword=set(stopwords.words('english'))
```

✓ 1.5s

```
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\ADMIN\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

- Chuyển dữ liệu cột “Genre” thành một danh sách (list).
- Tạo một đối tượng TfidfVectorizer từ thư viện text của scikit-learn. Vectorizer này sẽ chuyển đổi các mục văn bản thành ma trận TF-IDF, sử dụng danh sách từ ngữ dừng (stop words) tiếng Anh để loại bỏ những từ không mang ý nghĩa trong quá trình phân tích.
- Dùng fit_transform() để chuyển đổi dữ liệu văn bản trong cột "Genre" thành ma trận TF-IDF.
- similarity = cosine_similarity(tfidf_matrix): Sau khi có ma trận TF-IDF, dòng này sử dụng hàm cosine_similarity để tính độ tương đồng cosine giữa các văn bản. Kết quả là một ma trận tương đồng, trong đó mỗi phần tử (i, j) biểu thị độ tương đồng giữa mục i và mục j trong danh sách các thể loại ("Genre") dựa trên nội dung của chúng.

```
feature = data["Genre"].tolist()
tfidf = text.TfidfVectorizer(stop_words="english")
tfidf_matrix = tfidf.fit_transform(feature)
similarity = cosine_similarity(tfidf_matrix)
```

```
indices = pd.Series(data.index,
                    index=data['Movie Name']).drop_duplicates()
```

- Viết hàm

```python
# độ tương tự theo Tên
def movie_recommendation_byMovieName(title,data):
    # Reset index và loại bỏ cột index cũ
    data = data.reset_index(drop=True)
    feature = data["Movie Name"].tolist()
    tfidf = text.TfidfVectorizer(stop_words="english")
    tfidf_matrix = tfidf.fit_transform(feature)
    indices = pd.Series(data.index, index=data['Movie Name']).drop_duplicates()
    similarity = cosine_similarity(tfidf_matrix)
    index = indices[title]
    similarity_scores = list(enumerate(similarity[index]))
    similarity_scores = sorted(similarity_scores, key=lambda x: x[1], reverse=True)
    similarity_scores = similarity_scores[0:100]
    movieindices = [i[0] for i in similarity_scores]
    return data.iloc[movieindices].reset_index(drop=True)
```

88] ✓ 0.0s

```python
# độ tương tự theo thể loại
def movie_recommendation_byGenre(title,data):
    # Reset index và loại bỏ cột index cũ
    data = data.reset_index(drop=True)
    feature = data["Genre"].tolist()
    tfidf = text.TfidfVectorizer(stop_words="english")
    tfidf_matrix = tfidf.fit_transform(feature)
    indices = pd.Series(data.index, index=data['Movie Name']).drop_duplicates()
    similarity = cosine_similarity(tfidf_matrix)
    index = indices[title]
    similarity_scores = list(enumerate(similarity[index]))
    similarity_scores = sorted(similarity_scores, key=lambda x: x[1], reverse=True)
    similarity_scores = similarity_scores[0:100]
    movieindices = [i[0] for i in similarity_scores]
    return data.iloc[movieindices].reset_index(drop=True)
```

]

```python
def movie_recommendation(title,data):
    # độ tương tự theo Tên
    feature = data["Movie Name"].tolist()
    tfidf = text.TfidfVectorizer(stop_word  (parameter) data: Any
    tfidf_matrix = tfidf.fit_transform(fea
    indices = pd.Series(data.index, index=data['Movie Name']).drop_duplicates()
    similarity = cosine_similarity(tfidf_matrix)
    index = indices[title]
    similarity_scores = list(enumerate(similarity[index]))
    similarity_scores = sorted(similarity_scores, key=lambda x: x[1], reverse=True)
    similarity_scores = similarity_scores[0:100]
    movieindices = [i[0] for i in similarity_scores]
    data = data.iloc[movieindices].reset_index(drop=True)
    # độ tương tự theo thể loại
    feature = data["Genre"].tolist()
    tfidf = text.TfidfVectorizer(stop_words="english")
    tfidf_matrix = tfidf.fit_transform(feature)
    indices = pd.Series(data.index, index=data['Movie Name']).drop_duplicates()
    similarity = cosine_similarity(tfidf_matrix)
    index = indices[title]
    similarity_scores = list(enumerate(similarity[index]))
    similarity_scores = sorted(similarity_scores, key=lambda x: x[1], reverse=True)
    movieindices = [i[0] for i in similarity_scores[0:10]]
    return data.iloc[movieindices].reset_index(drop=True)
```

✓ 0.0s

```python
movie_recommendation("Iron Man",data)
```

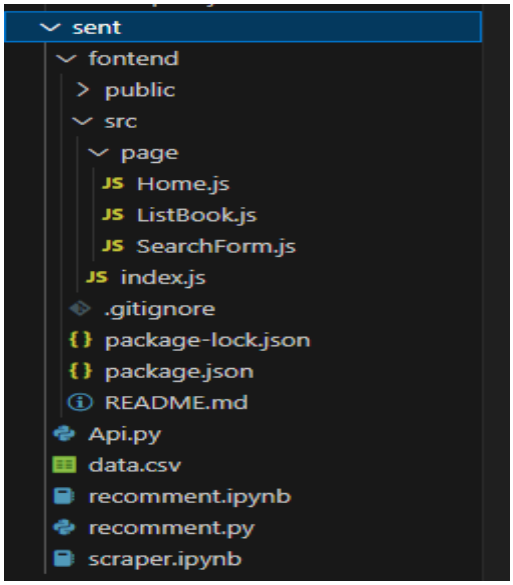
✓ 0.2s

693

	Movie Name	Description	Genre
0	Iron Man	['After', 'being', 'held', 'captive', 'in', 'a...	['Action', ' Adventure', ' Sci-Fi']
1	Iron Man Three	['When', 'Tony', "Stark's", 'world', 'is', 'to...	['Action', ' Adventure', ' Sci-Fi']
2	Spider-Man 2	['Peter', 'Parker', 'is', 'beset', 'with', 'tr...	['Action', ' Adventure', ' Sci-Fi']
3	Spider-Man	['After', 'being', 'bitten', 'by', 'a', 'genet...	['Action', ' Adventure', ' Sci-Fi']
4	Spider-Man 3	['A', 'strange', 'black', 'entity', 'from', 'a...	['Action', ' Adventure', ' Sci-Fi']
5	Man of Steel	['An', 'alien', 'child', 'is', 'evacuated', 'f...	['Action', ' Adventure', ' Sci-Fi']
6	Iron Man 2	['With', 'the', 'world', 'now', 'aware', 'of',...	['Action', ' Sci-Fi']
7	The Omega Man	['Dr', 'Robert', 'Neville', 'has', 'developed'...	['Action', ' Drama', ' Sci-Fi']
8	Ant-Man	['Armed', 'with', 'a', 'super-suit', 'with', '...	['Action', ' Comedy', ' Sci-Fi']
9	The Running Man	['In', 'a', 'dystopian', 'America,', 'a', 'fal...	['Action', ' Sci-Fi', ' Thriller']

4.2. Áp dụng vào hệ thống

Cấu trúc thư mục của dự án bao gồm hai phần chính là backend và frontend. Phần backend đảm nhận vai trò trong việc gửi các yêu cầu từ phía người dùng tới máy chủ và xử lý chúng, cung cấp dữ liệu và thông tin cho phần frontend. Mặt khác, là giao diện người dùng, hiển thị thông tin từ phía backend và tương tác trực tiếp với người dùng.



4.3. Các file quan trọng:

data.csv: chứa dữ liệu của hệ thống

api.py: cung cấp api

recomment.py: cung cấp phương thức xử lý yêu cầu hệ thống

4.4. Code

recomment.py | App.js | recomment.ipynb | data.csv

sent > recomment.py > ...

```python
import numpy as np
import pandas as pd
import nltk
nltk.download('stopwords')
import re
from sklearn.feature_extraction import text
from sklearn.metrics.pairwise import cosine_similarity
stemmer = nltk.SnowballStemmer("english")
from nltk.corpus import stopwords
import string
stopword=set(stopwords.words('english'))


# đọc dữ liệu
data = pd.read_csv("data.csv")
data = data[["Movie Name", "Description", "Genre"]]
data = data.dropna()

def movie_recommendation(title,data):
    data = data.reset_index(drop=True)
    feature = data["Movie Name"].tolist()
    tfidf = text.TfidfVectorizer(stop_words="english")
    tfidf_matrix = tfidf.fit_transform(feature)
    indices = pd.Series(data.index, index=data['Movie Name']).drop_duplicates()
    similarity = cosine_similarity(tfidf_matrix)
    index = indices[title]
    similarity_scores = list(enumerate(similarity[index]))
    similarity_scores = sorted(similarity_scores, key=lambda x: x[1], reverse=True)
    similarity_scores = similarity_scores[0:100]
    movieindices = [i[0] for i in similarity_scores]
    data = data.iloc[movieindices].reset_index(drop=True)

    feature = data["Genre"].tolist()
    tfidf = text.TfidfVectorizer(stop_words="english")
    tfidf_matrix = tfidf.fit_transform(feature)
    indices = pd.Series(data.index, index=data['Movie Name']).drop_duplicates()
    similarity = cosine_similarity(tfidf_matrix)
    index = indices[title]
    similarity_scores = list(enumerate(similarity[index]))
    similarity_scores = sorted(similarity_scores, key=lambda x: x[1], reverse=True)
    movieindices = [i[0] for i in similarity_scores[0:10]]
    df = data.iloc[movieindices].reset_index(drop=True).rename(columns={'Movie Name': 'name', 'Description': 'description', 'Genre': 'genre'})
    json_data = df.to_json(orient='records')
    return json_data

def getpage(page):
    page = int(page)
    df = data[page*10:(page+1)*10].rename(columns={'Movie Name': 'name', 'Description': 'description', 'Genre': 'genre'})
    json_data = df.to_json(orient='records')
    return json_data

```

```python
from recomment import netFlix_recommendation,getpage
from flask import Flask, render_template,request,jsonify,json
from flask_cors import CORS, cross_origin
import pandas as pd

app=Flask(__name__)
CORS(app, support_credentials=True)
@app.route("/recomment",methods=['GET','POST'])
def recomment():
    param_value = request.args.get('name')
    return netFlix_recommendation(f'{param_value}')

@app.route("/home",methods=['GET','POST'])
def home():
    param_value = request.args.get('page')
    return getpage(f'{param_value}')

if __name__ == '__main__':
    app.run(debug=True)
```

4.5. Giao diện hệ thống