TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT TP. HỒ CHÍ MINH
KHOA ĐÀO TẠO CHẤT LƯỢNG CAO
NGÀNH CÔNG NGHỆ THÔNG TIN

BÁO CÁO TIẾN ĐỘ -TUẦN 05
MÔN: ĐỒ ÁN 3
ĐỀ TÀI: TÌM HIỂU VỀ THUẬT TOÁN RECOMMENDATION

GVHD : Thầy Huỳnh Xuân Phụng
SVTH :
Nguyễn Thành Như **17110202**
Võ Ngọc Thuận **17110234**

TP. Hồ Chí Minh, tháng 11 năm 2020

1. Demo Recommendation System with Content-based Filtering

1.1. Code Demo

```python
import pandas as pd
import numpy as np
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.metrics.pairwise import cosine_similarity
###### helper functions. Use them when needed #######
def get_title_from_index(index):
    return df[df.index == index]["title"].values[0]

def get_index_from_title(title):
    return df[df.title == title]["index"].values[0]
##################################################

##Step 1: Read CSV File
df = pd.read_csv("movie_dataset.csv")
print (df.columns)

##Step 2: Select Features
features = ['keywords','cast','genres','director']

##Step 3: Create a column in DF which combines all selected features
for feature in features:
    df[feature] = df[feature].fillna('')

def combine_features(row):
    try:
        return row['keywords']+" "+row['cast']+" "+row['genres']+" "+row['director']
    except:
        print ("Error:", row)

df["combined_features"] = df.apply(combine_features,axis=1)
df.iloc[0].combined_features
##Step 4: Create count matrix from this new combined column
cv = CountVectorizer()
count_matrix = cv.fit_transform(df["combined_features"])

##Step 5: Compute the Cosine Similarity based on the count_matrix
cosine_sim = cosine_similarity(count_matrix)

```

```
## Step 6: Get index of this movie from its title
def get_title_from_index(index):
    return df[df.index == index]["title"].values[0]
def get_index_from_title(title):
    return df[df.title == title]["index"].values[0]
movie_user_likes = "Avatar"
movie_index = get_index_from_title(movie_user_likes)
similar_movies = list(enumerate(cosine_sim[movie_index]))

## Step 7: Get a list of similar movies in descending order of similarity score
sorted_similar_movies = sorted(similar_movies,key=lambda x:x[1],reverse=True)[1:]

## Step 8: Print titles of first 50 movies
i=0
print("Top 5 similar movies to "+movie_user_likes+" are:\n")
for element in sorted_similar_movies:
    print(get_title_from_index(element[0]))
    i=i+1
    if i>5:
        break

```

1.2. Giải thích code:

- **import pandas as pd** : chúng ta tiến hành import thư viện Pandas, nó là một thư viện Python cung cấp các cấu trúc dữ liệu nhanh, mạnh mẽ, linh hoạt và mang hàm ý. Chúng em dùng nó để đọc dữ liệu từ file csv.
- **from scipy import sparse** : scipy là một thư viện mã nguồn mở các thuật toán và các công cụ toán học cho Python, được xây dựng trên các đối tượng mảng NumPy tạo thành ngăn xếp NumPy bao gồm các công cụ như Pandas, SymPy và Matplotlib. SciPy cung cấp khá nhiều module tính toán từ đại số tuyến tính, tích phân, vi phân, nội suy đến xử lý ảnh, fourier transform... và cụ thể ở đây ta import gói ma trận sparse và các đoạn chương trình liên quan
- **from sklearn.metrics.pairwise import cosine_similarity** : cosine_similarity được lấy từ sklearn.metrics.pairwise để tính toán độ tương tự giữa 2 vectors
- **Step 1: df=pd.read_csv("movie_dataset.csv"):** gán biến df và đọc biến df với dữ liệu là file movie_dataset.csv bằng việc đọc vào một file .csv bằng cách sử dụng hàm read_csv và được trả về 1 dataframe.

```
##Step 1: Read CSV File
df = pd.read_csv("movie_dataset.csv")
print (df.columns)

```

- **Step 2: features = ['keywords','cast','genres','director']:** chọn 4 features 'keywords', 'cast', 'genres', 'director' làm bộ tính năng cho nội dung của movie_dataset

```
##Step 2: Select Features
features = ['keywords','cast','genres','director']
```

- **Step 3:** Nhiệm vụ tiếp theo của chúng ta là tạo một hàm để kết hợp các giá trị của các cột này thành một chuỗi duy nhất. chúng ta cần gọi hàm này trên mỗi hàng của khung dữ liệu. Tuy nhiên, trước khi làm điều đó, chúng ta cần clean và xử lý trước dữ liệu để sử dụng. Điền tất cả các giá trị NaN bằng chuỗi trống trong khung dữ liệu.

```
##Step 3: Create a column in DF which combines all selected features
for feature in features:
    df[feature] = df[feature].fillna('')

def combine_features(row):
    try:
        return row['keywords']+" "+row['cast']+" "+row['genres']+" "+row['director']
    except:
        print ("Error:", row)

df["combined_features"] = df.apply(combine_features,axis=1)
df.iloc[0].combined_features
```

- **Step 4: cv = CountVectorizer():** tạo object CountVectorizer().

  count_matrix = cv.fit_transform(df["combined_features"]): combine chuỗi string(movie contents) với object CountVectorizer().

```
##Step 4: Create count matrix from this new combined column
cv = CountVectorizer()
count_matrix = cv.fit_transform(df["combined_features"])
```

- **Step 5:** Chúng ta cần lấy ma trận tương tự cosine từ ma trận đếm.

```
##Step 5: Compute the Cosine Similarity based on the count_matrix
cosine_sim = cosine_similarity(count_matrix)
```

- **Step 6:** Xác định hai hàm trợ giúp để lấy title phim từ chỉ mục phim và ngược lại.

```
## Step 6: Get index of this movie from its title
def get_title_from_index(index):
    return df[df.index == index]["title"].values[0]
def get_index_from_title(title):
    return df[df.title == title]["index"].values[0]
movie_user_likes = "Avatar"
movie_index = get_index_from_title(movie_user_likes)
similar_movies = list(enumerate(cosine_sim[movie_index]))
```

Bước tiếp theo là lấy tên phim mà người dùng đang thích. Sau đó, tìm chỉ số của bộ phim đó. Sau đó, truy cập vào hàng tương ứng với bộ phim này trong ma trận tương tự. Như vậy, chúng ta sẽ nhận được điểm tương đồng của tất cả

các phim khác từ phim hiện tại. Sau đó, thống kê thông qua tất cả các điểm tương tự của bộ phim đó để tạo ra một bộ chỉ số phim và điểm tương tự. Điều này sẽ chuyển đổi một hàng điểm số tương tự như thế này- [1 0,5 0,2 0,9] thành hàng này- [(0, 1) (1, 0,5) (2, 0,2) (3, 0,9)].

```
45 movie_user_likes = "Avatar"
46 movie_index = get_index_from_title(movie_user_likes)
47 similar_movies = list(enumerate(cosine_sim[movie_index]))
```

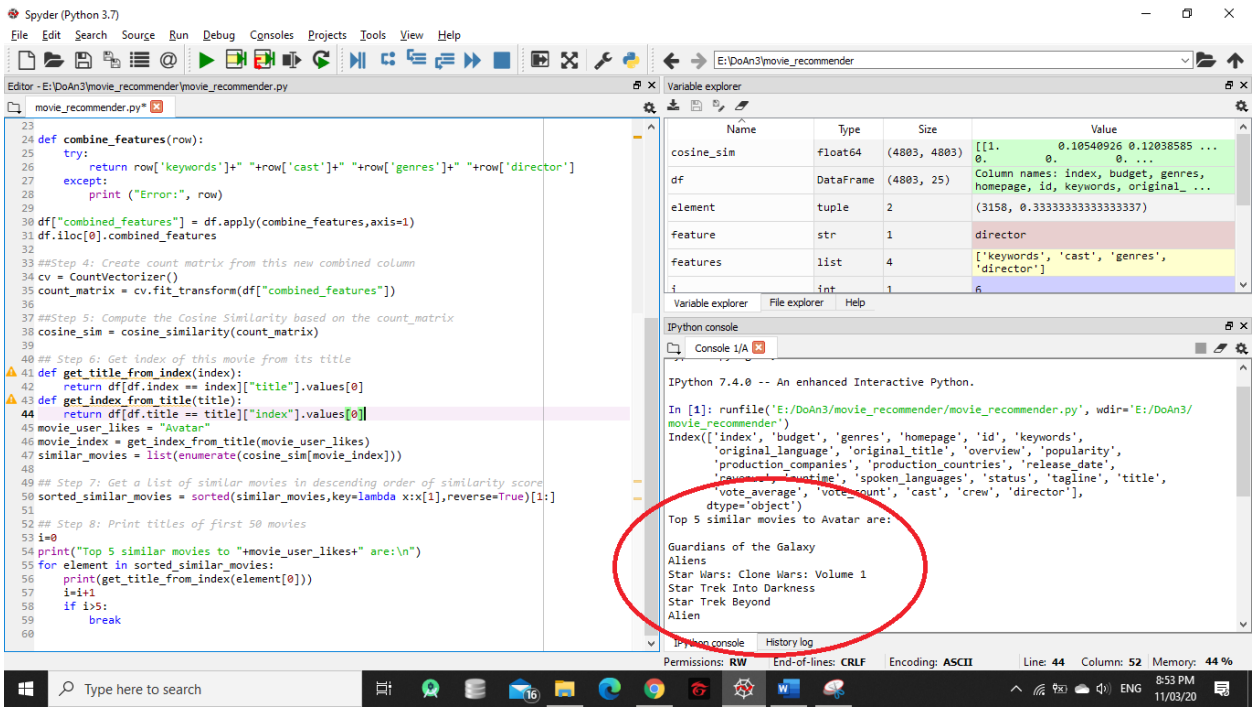
- **Step 7:** Sắp xếp danh sách similar_movies theo điểm tương tự theo thứ tự giảm dần. Vì phim tương tự nhất với một phim nhất định sẽ là chính nó, loại bỏ phần tử đầu tiên sau khi sắp xếp các phim.

```
49 ## Step 7: Get a list of similar movies in descending order of similarity score
50 sorted_similar_movies = sorted(similar_movies,key=lambda x:x[1],reverse=True)[1:]
```

- **Step 8:** Chạy vòng lặp để in 5 entries đầu tiên từ danh sách sorted_similar_movies.

```
49 ## Step 7: Get a list of similar movies in descending order of similarity score
50 sorted_similar_movies = sorted(similar_movies,key=lambda x:x[1],reverse=True)[1:]
51
52 ## Step 8: Print titles of first 50 movies
53 i=0
54 print("Top 5 similar movies to "+movie_user_likes+" are:\n")
55 for element in sorted_similar_movies:
56     print(get_title_from_index(element[0]))
57     i=i+1
58     if i>5:
59         break
```

Kết quả: Top 5 bộ phim liên quan, gần với phim Avatar nhất là:



Top 5 bộ phim liên quan, gần với bộ phim Avatar nhất là:

Guardians of the Galaxy
Aliens
Star Wars: Clone Wars: Volume 1
Star Trek Into Darkness
Star Trek Beyond
Alien

II. Công việc tuần sau:

- Demo dựa theo Clusting User.