Universidad
Internacional
Menéndez Pelayo

Máster Universitario en Investigación en Inteligencia Artificial

Curso 2020-2021

Sistemas de Recomendación

Recomendación para grupos en Python

30 de abril de 2021

Laura Rodríguez Navas
DNI: 43630508Z
e-mail: rodrigueznavas@posgrado.uimp.es

Índice

1. Introducción

2. Conjunto de datos

El conjunto de datos que se ha usado en esta práctica se encuentra disponible públicamente para su descarga en el siguiente enlace: https://www.kaggle.com/abhikjha/movielens-100k/download. Este conjunto de datos llamado *ml-latest-small*, describe las valoraciones (entre 1 y 5 estrellas) y la actividad del etiquetado de MovieLens [1], un servicio de recomendación de películas. Concretamente, el conjunto de datos contiene 100836 clasificaciones y 3683 etiquetas de 9742 películas. Los datos fueron creados por 610 usuarios que fueron seleccionados al azar. Cada usuario clasificó al menos 20 películas y está representado por una identificación.

Una vez descargado el conjunto de datos veremos que está contenido en los archivos *links.csv*, *movies.csv*, *ratings.csv* y *tags.csv*. Pero para el desarrollo de esta práctica solo se utilizan los archivos *movies.csv* y *ratings.csv*. A continuación, cargamos cada archivo dentro de su dataframe (ver Definición 1) con el uso de la librería pandas [2].

```
movies_df = pd.read_csv('dataset/movies.csv')
ratings_df = pd.read_csv('dataset/ratings.csv')
```

Observamos el contenido de *movies_df* para ver como ha quedado organizado:

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

Tabla 1: Contenido del dataframe *movies_df* inicialmente.

Cada película tiene un único identificador, un título con su año de estreno y diferentes géneros. Como los años contienen caracteres *unicode* y para que no haya problemas más adelante, los sacaremos de la columna de los títulos y los ubicaremos en su propia columna que nombraremos *year*. Para ello, primero utilizamos una expresión regular para encontrar los años guardados entre paréntesis, y con la función *extract* de la librería pandas los extraemos de la columna de los títulos para guardarlos en su propia columna. Después borramos los años de la columna de los títulos y con la función *strip* nos aseguramos de sacar los espacios finales que pudiera haber.

```
regular_expression = r'\((.*?)\)'
movies_df['year'] = movies_df.title.str.lower().str.extract(regular_expression)
movies_df['title'] = movies_df.title.str.replace(regular_expression, '', regex=True)
movies_df['title'] = movies_df['title'].apply(lambda x: x.strip())
```

Vemos el resultado:

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

Tabla 2: Contenido del dataframe *movies_df* con la columna *year*.

Eliminamos la columna de los géneros, ya que no los tendremos en cuenta para el sistema de recomendación.

```
movies_df = movies_df.drop('genres', 1)
```

Finalmente, así queda el dataframe *movies_df*:

movieId	title	year
1	Toy Story	1995
2	Jumanji	1995
3	Grumpier Old Men	1995
4	Waiting to Exhale	1995
5	Father of the Bride Part II	1995

Tabla 3: Contenido del dataframe *movies_df* final.

Definición 1. *Un DataFrame es una estructura de datos etiquetada bidimensional que acepta diferentes tipos datos de entrada organizados en columnas. Se puede pensar en un DataFrame como una hoja de cálculo o una tabla SQL.*

Ahora, veremos cómo se ha organizado el dataframe *ratings_df*. En este caso también hemos eliminado una columna, la columna *timestamp*, ya que tampoco se tendrá en cuenta en el sistema de recomendación.

```
ratings_df = ratings_df.drop('timestamp', 1)
```

Finalmente, así queda el dataframe *ratings_df*:

userId	movieId	rating
1	1	4.0
1	3	4.0
1	6	4.0
1	47	5.0
1	50	5.0

Tabla 4: Contenido del dataframe *ratings_df* final.

Cada fila del dataframe *ratings_df* tiene un identificador de usuario asociado con al menos una película, el identificador de la película y una valoración de esta.

3. Filtrado Colaborativo

En esta sección, describiremos el sistema de recomendación basado en usuarios que ha sido desarrollado en Python [3].

Como hemos comentado en la sección de Introducción (ver sección 1), la técnica que usaremos y que hemos desarrollado para el sistema de recomendación se llama Filtrado Colaborativo. El Filtrado Colaborativo también es conocido como Filtrado de Usuario a Usuario. Con esta técnica utilizaremos los usuarios del conjunto de datos para recomendar películas a un nuevo usuario que añadirá nuevos datos al sistema. Para ello, el sistema de recomendación intentará encontrar usuarios que tengan valoraciones parecidas a las del nuevo usuario, y entonces recomendarle al nuevo usuario películas acordes a sus valoraciones. Para encontrar usuarios parecidos existen varios métodos, en el caso de la práctica el método elegido se basa en el coeficiente de correlación de Pearson.

El proceso para crear el sistema de recomendación sigue los siguientes pasos:

- Crear un nuevo usuario con las películas del conjunto de datos que el usuario a mirado.
- Basado en su índice de selección de películas, encontrar a los primeros X vecinos.
- Obtener el identificador de las películas que miró el nuevo usuario por cada vecino.
- Calcular las similitudes entre los usuarios existentes y el nuevo usuario.
- Recomendar películas al nuevo usuario con las puntuaciones más altas.

Así, comenzamos creando un nuevo usuario a quien recomendar películas. Para ello, creamos el archivo *dataset/user_ratings.csv* que contiene 100 películas elegidas aleatoriamente del conjunto de datos, y a continuación las valoramos. En este caso, he valorado las películas según mi criterio. Este archivo se puede modificar como se desee para realizar tantas recomendaciones como se quiera, solo nos debemos asegurar de escribir los títulos de las películas en mayúsculas y si una película comienza con un "The", como "The Matrix" entonces escribirlo así: "Matrix, The". Para finalizar con la creación del nuevo usuario, el archivo *dataset/user_ratings.csv* se carga dentro de un nuevo dataframe:

```
user_df = pd.read_csv('dataset/user_ratings.csv')
```

Con los datos del nuevo usuario en el sistema de recomendación, extraeremos los identificadores de las películas del dataframe de películas, para agregarlos con los nuevos datos añadidos por el usuario. Para ello, primero se extraen los títulos de las películas que el usuario haya visto para luego juntarlas con los datos del usuario. En este punto también, vamos a aprovechar a eliminar información que no se utilizará más adelante en el sistema de recomendación, como la columna *year*, para ahorrar espacio de memoria.

```
titles = movies_df[movies_df['title'].isin(user_df['title'].tolist())]
user_df = pd.merge(titles, user_df)
user_df = user_df.drop('year', 1)
```

Vemos el resultado:

movieId	title	rating
1	Toy Story	5.0
2	Jumanji	5.0
3	Grumpier Old Men	1.0
4	Waiting to Exhale	1.0
5	Father of the Bride Part II	1.0

Tabla 5: Contenido del dataframe *user_df*.

Ahora, que hemos añadido los identificadores de las películas con los datos del nuevo usuario, podemos obtener el subconjunto de usuarios que han visto las mismas películas. Además, también agruparemos las filas por el identificador de usuario.

```
user_titles = ratings_df[ratings_df['movieId'].isin(user_df['movieId'].tolist())]
user_groups = user_titles.groupby(['userId'])
```

Observaremos a uno de los usuarios, por ejemplo, el usuario 525.

userId	movieId	rating
525	1	4.0
525	2	3.5
525	34	3.0
525	39	4.5
525	48	3.0

Tabla 6: Contenido del dataframe *user_df*.

Finalmente, ordenaremos el conjunto de datos de tal forma que los usuarios que compartan la mayor cantidad de películas tengan prioridad. Esto nos brindaré una mejor recomendación ya que no será necesario pasar por todos los usuarios.

```
user_groups = sorted(user_groups, key=lambda x: len(x[1]), reverse=True)
```

3.1. Similitud entre los usuarios y el nuevo usuario

En esta sección, buscamos las similitudes de todos los usuarios con el nuevo usuario para encontrar aquellos que se le parecen más. Tendremos que ver como se relacionan entre sí, y para ello usaremos el coeficiente de correlación de Pearson. La fórmula para calcular este coeficiente se puede ver a continuación:

$$r = \frac{\sum_1^n (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_1^n (x_i - \bar{x})^2}\ \sqrt{\sum_1^n (y_i - \bar{y})^2}} \tag{1}$$

Se ha decidido usar el coeficiente de correlación de Pearson por una de sus propiedades. La propiedad dice que, si se multiplican todos los elementos por una constante distinta a cero o si se agrega cualquier constante a todos los elementos del coeficiente, este no cambia con la escala. Por ejemplo, si tenemos dos vectores X e Y, entonces, $pearson(X, Y) = pearson(X, 2 \cdot Y + 3)$. Esta es una propiedad muy importante en los sistemas de recomendación porqué si dos usuarios valoran dos series de elementos de manera completamente diferente,

pero son usuarios muy parecidos (con ideas similares) contamos con valoraciones muy parecidas en escalas variadas.

Los valores brindados por la fórmula(1) pueden variar de $r = -1$ a $r = 1$, donde 1 se correlaciona directamente entre dos entidades (esto sería una correlación positiva perfecta) y -1 forma una correlación negativa perfecta. En el caso de la práctica, un 1 se refiere a que dos usuarios tendrán gustos parecidos, mientras que -1 será lo opuesto.

```
pearsonCorrelationDict = {}

for name, group in user_groups:      # For each user

# Sorting the current user in such a way that the values don't get mixed up later
user = group.sort_values(by='movieId')
movies = user_df.sort_values(by='movieId')

# Get the number of elements (N) for the formula
nRatings = len(user)

# Set ratings for movies in common in a list
temp_df = movies[movies['movieId'].isin(user['movieId'].tolist())]
tempRatingList = temp_df['rating'].tolist()

# Set user ratings in a list
tempGroupList = user['rating'].tolist()

# Calculate the Pearson Correlation between two users, x and y
Uxx = sum([i ** 2 for i in tempRatingList]) - pow(sum(tempRatingList), 2)
/ float(nRatings)
Uyy = sum([i ** 2 for i in tempGroupList]) - pow(sum(tempGroupList), 2)
/ float(nRatings)
Uxy = sum(i * j for i, j in zip(tempRatingList, tempGroupList)) - sum(tempRatingList)
* sum(tempGroupList) / float(nRatings)

# If the denominator is nonzero, then we divide, otherwise the correlation is 0
if Uxx != 0 and Uyy != 0:
pearsonCorrelationDict[name] = Uxy / sqrt(Uxx * Uyy)

else:
pearsonCorrelationDict[name] = 0
```

4. Conclusiones

Automatizar la creación del archivo *dataset/user_ratings.csv*.

El enlace al vídeo debéis incluirlo en la memoria. Trabajo práctico: la idea es que vayáis explicando el porqué de las decisiones que habéis tomado (qué me llevó a utilizar unos determinados hiperparámetros, por qué he utilizado esta manera de medir la calidad de la solución,...) y hagáis un razonamiento sobre los resultados obtenidos.

Ahora, calculemos el coeficiente de correlación de Pearson entre el nuevo usuario y el resto de los usuarios del conjunto de datos. El resultado lo vamos a almacenar en un diccionario, donde la clave es el identificador del usuario y el valor es el coeficiente.

Bibliografía

[1] F. Maxwell Harper and Joseph A. Konstan. The movielens datasets: History and context. *ACM Trans. Interact. Intell. Syst.*, 5(4), December 2015.

[2] Jeff Reback, Wes McKinney, jbrockmendel, Joris Van den Bossche, Tom Augspurger, Phillip Cloud, gfyoung, Sinhrks, Simon Hawkins, Matthew Roeschke, Adam Klein, Terji Petersen, Jeff Tratner, Chang She, William Ayd, Shahar Naveh, Marc Garcia, Jeremy Schendel, Andy Hayden, Daniel Saxton, Vytautas Jancauskas, Ali McMaster, Pietro Battiston, Skipper Seabold, patrick, Kaiqi Dong, chris b1, h vetinari, Stephan Hoyer, and Marco Gorelli. pandas-dev/pandas: Pandas 1.1.5, December 2020.

[3] Laura Rodríguez-Navas. Recomendación para grupos. https://github.com/lrodrin/masterAI/tree/master/A13, 2021.