1_book_recommends

November 4, 2024

1 Recomendación de libros con KNN

Alumno: Rodolfo Luthe Narváez

1.1 Justificación

Soy un lector ávido, regularmente leo libros de ficción y tengo la necesidad de conseguir nuevas recomendaciones constantemente. Como la clase de Machine Learning necesita que usemos un algoritmo de aprendizaje y lo entrenemos con un buen volumen de datos pienso que este proyecto es apropiado para presentar este semestre para resolver mi necesidad de tener tema para el proyecto y de conseguir mas recomendaciones para mi.

1.2 Objetivos

Tener un algoritmo entrenado con críticas de usuarios en internet capaz de recomendar libros basandose en la similaridad de mis criticas a las de los demás usuarios

1.3 Adquisición de Datos y Preprocesamiento

Los datos vendrán de dos fuentes: - Kaggle - Goodreads

Kaggle nos servirá para conseguir el gran volumen de datos necesario para entrenar el modelo, y viene separado en tres archivos - Books.csv - Users.csv - Ratings.csv

1.3.1 Kaggle

El dataset contiene 278,858 users (anonimos pero con informacion demografica) que escribieron 1,149,780 ratings sobre 271,379 librosß. #### Books.csv Cada uno de los libros en Books.csv tiene los siguientes elementos: - ISBN - Book-Title - Book-Author - Year-Of-Publication - Publisher - Image-URL-S - Image-URL-M - Image-URL-L

Solamente usaremos el ISBN del libro para identificar las criticas que le corresponden. El ISBN es el ID del libro, identifica el volumen con un codigo de 10 digitos. El titulo y autor solamente nos servira para identificar el libro ante el usuario, pero por claridad se conservara en este notebook.

Users.csv Lus usuarios tienen tres datos en el .csv: - User-ID - Location - Age

Primariamente nos interesa el ID para asociarlo con las criticas, no tanto la ubicacion o la edad

Ratings.csv Este es el archivo que nos interesa mas para el proyecto, contiene criticas de los libros que usaremos para calcular cuales titulos recomendar en funcion de los libros que al usuariomle han gustado: - User-ID - ISBN - Book-Rating

1.4 Imports

[223]:
```
import numpy as np
import pandas as pd
from scipy.sparse import csr_matrix
from sklearn.neighbors import NearestNeighbors
import matplotlib.pyplot as plt
import warnings
import os, sys
import re

warnings.filterwarnings('ignore')
pd.set_option('display.max_colwidth', None)
```

1.4.1 Load Books

[224]:
```
df_books = pd.read_csv('../data/raw/Books.csv')
df_books.head()
```

[224]:
```
         ISBN  \
0  0195153448
1  0002005018
2  0060973129
3  0374157065
4  0393045218

           Book-Title  \
0
Classical Mythology
1
Clara Callan
2
Decision in Normandy
3  Flu: The Story of the Great Influenza Pandemic of 1918 and the Search for the
Virus That Caused It
4
The Mummies of Urumchi

            Book-Author Year-Of-Publication                   Publisher  \
0    Mark P. O. Morford                2002     Oxford University Press
1  Richard Bruce Wright                2001       HarperFlamingo Canada
2          Carlo D'Este                1991             HarperPerennial
3      Gina Bari Kolata                1999        Farrar Straus Giroux
```

```
4       E. J. W. Barber                1999  W. W. Norton &amp;amp; Company

                                                    Image-URL-S  \
0  http://images.amazon.com/images/P/0195153448.01.THUMBZZZ.jpg
1  http://images.amazon.com/images/P/0002005018.01.THUMBZZZ.jpg
2  http://images.amazon.com/images/P/0060973129.01.THUMBZZZ.jpg
3  http://images.amazon.com/images/P/0374157065.01.THUMBZZZ.jpg
4  http://images.amazon.com/images/P/0393045218.01.THUMBZZZ.jpg

                                                    Image-URL-M  \
0  http://images.amazon.com/images/P/0195153448.01.MZZZZZZZ.jpg
1  http://images.amazon.com/images/P/0002005018.01.MZZZZZZZ.jpg
2  http://images.amazon.com/images/P/0060973129.01.MZZZZZZZ.jpg
3  http://images.amazon.com/images/P/0374157065.01.MZZZZZZZ.jpg
4  http://images.amazon.com/images/P/0393045218.01.MZZZZZZZ.jpg

                                                    Image-URL-L
0  http://images.amazon.com/images/P/0195153448.01.LZZZZZZZ.jpg
1  http://images.amazon.com/images/P/0002005018.01.LZZZZZZZ.jpg
2  http://images.amazon.com/images/P/0060973129.01.LZZZZZZZ.jpg
3  http://images.amazon.com/images/P/0374157065.01.LZZZZZZZ.jpg
4  http://images.amazon.com/images/P/0393045218.01.LZZZZZZZ.jpg
```

Drop unwanted columns

[225]:
```
df_books = df_books[['ISBN', 'Book-Title', 'Book-Author']]
df_books.head()
```

[225]:
```
          ISBN  \
0  0195153448
1  0002005018
2  0060973129
3  0374157065
4  0393045218

            Book-Title  \
0
Classical Mythology
1
Clara Callan
2
Decision in Normandy
3  Flu: The Story of the Great Influenza Pandemic of 1918 and the Search for the
Virus That Caused It
4
The Mummies of Urumchi

             Book-Author
```

```
0    Mark P. O. Morford
1  Richard Bruce Wright
2          Carlo D'Este
3      Gina Bari Kolata
4       E. J. W. Barber
```

1.4.2 Load ratings

[226]:
```
df_ratings = pd.read_csv('../data/raw/Ratings.csv')
df_ratings.head()
```

[226]:
```
   User-ID        ISBN  Book-Rating
0   276725  034545104X            0
1   276726  0155061224            5
2   276727  0446520802            0
3   276729  052165615X            3
4   276729  0521795028            6
```

1.5 Clean Nulls

[227]:
```
df_books.isnull().sum()
```

[227]:
```
ISBN           0
Book-Title     0
Book-Author    2
dtype: int64
```

[228]:
```
df_ratings.isnull().sum()
```

[228]:
```
User-ID        0
ISBN           0
Book-Rating    0
dtype: int64
```

[229]:
```
df_books.dropna(inplace=True)
df_books.isnull().sum()
```

[229]:
```
ISBN           0
Book-Title     0
Book-Author    0
dtype: int64
```

[230]:
```
df_books.shape
```

[230]:
```
(271358, 3)
```

[231]:
```
df_ratings.shape
```

[231]: (1149780, 3)

1.6 Procesar ratings

Analizamos cuantos ratings han escrito los usuarios con el mayor numero de entradas. Podemos ver que los usuarios mas prolificos tienen miles de ratings a su nombre

[232]:
```
ratings = df_ratings['User-ID'].value_counts()
ratings.sort_values(ascending=False).head()
```

[232]:
```
User-ID
11676     13602
198711     7550
153662     6109
98391      5891
35859      5850
Name: count, dtype: int64
```

Hacemos la operacion inversa y vemos que hay usuarios con tan solo un rating. Vamos a desechar a los usuarios con menos de un numero criticas para asegurarnos que solo la gente con cierto nivel de experiencia pueda contribuir al entrenamiento de nuestro recomendador

[233]:
```
ratings.sort_values(ascending=True).head()
```

[233]:
```
User-ID
119628    1
205542    1
205545    1
205552    1
205555    1
Name: count, dtype: int64
```

Veremos cuantos usuarios hay con menos de x ratings para ver cual es nuestro numero minimo de criticas

[234]:
```
print("Total de usuarios:                ",df_ratings.shape[0])

print("Usuarios con menos de 5 ratings:  ", len(ratings[ratings < 5]))
print("Usuarios con menos de 10 ratings: ", len(ratings[ratings < 10]))
print("Usuarios con menos de 50 ratings: ", len(ratings[ratings < 50]))
print("Usuarios con menos de 100 ratings:", len(ratings[ratings < 100]))
print("Usuarios con menos de 150 ratings:", len(ratings[ratings < 150]))
print("Usuarios con menos de 200 ratings:", len(ratings[ratings < 200]))
```

```
Total de usuarios:                1149780
Usuarios con menos de 5 ratings:  82467
Usuarios con menos de 10 ratings: 92186
Usuarios con menos de 50 ratings: 101856
Usuarios con menos de 100 ratings: 103436
```

```
Usuarios con menos de 150 ratings: 104049
Usuarios con menos de 200 ratings: 104378
```

Vemos que no hay tanta diferencia despues de quitar a los usuarios con menos de 5 ratings en el volumen total de datos. En el resto del proyecto quitaremos los usuarios con menos de 10 ratings

[235]:
```
df_ratings['User-ID'].isin(ratings[ratings < 10].index).sum()
```

[235]:
```
np.int64(184067)
```

[236]:
```
df_ratings_rm = df_ratings[
  ~df_ratings['User-ID'].isin(ratings[ratings < 10].index)
]
df_ratings_rm.shape
```

[236]:
```
(965713, 3)
```

Hacer lo mismo para libros con menos de 100 ratings

[237]:
```
ratings = df_ratings['ISBN'].value_counts()
ratings.sort_values(ascending=False).head()
```

[237]:
```
ISBN
0971880107    2502
0316666343    1295
0385504209     883
0060928336     732
0312195516     723
Name: count, dtype: int64
```

[238]:
```
ratings.sort_values(ascending=True).head()
```

[238]:
```
ISBN
0671883917    1
0060164220    1
0316185922    1
1931333246    1
0140260676    1
Name: count, dtype: int64
```

[239]:
```
print("Libros de usuarios:             ",ratings.shape[0])

print("Libros con menos de 20 ratings: ", len(ratings[ratings < 20]))
print("Libros con menos de 50 ratings: ", len(ratings[ratings < 50]))
print("Libros con menos de 100 ratings:", len(ratings[ratings < 100]))
```

```
Libros de usuarios:              340556
Libros con menos de 20 ratings:  333066
Libros con menos de 50 ratings:  338371
Libros con menos de 100 ratings: 339825
```

[240]:
```
df_ratings_rm = df_ratings_rm[
  ~df_ratings_rm['ISBN'].isin(ratings[ratings < 100].index)
]
df_ratings_rm.shape
```

[240]:
```
(108113, 3)
```

Tenemos nuestro dataset de ratings con user, ISBN y calificacion

[241]:
```
df_ratings_rm.head()
```

[241]:

	User-ID	ISBN	Book-Rating
34	276762	0451167317	0
133	276822	0060096195	10
145	276822	0786817070	10
173	276847	0446364193	0
413	276925	002542730X	10

Generamos la matriz de usuario-libro con el valor de la celda siendo la calificacion. Si no ha dado un rating para ese libro llenamos con 0

[242]:
```
df = df_ratings_rm.
  ↪pivot_table(index=['User-ID'],columns=['ISBN'],values='Book-Rating').
  ↪fillna(0).T
df_isbn = df_books.copy()
df.head()
```

[242]:

User-ID	99	242	243	254	383	384	388	408	\
ISBN									
002542730X	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
0060008032	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
0060096195	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
006016848X	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
0060173289	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	

User-ID	424	446	…	278418	278506	278522	278535	278536	\
ISBN			…						
002542730X	0.0	0.0	…	0.0	0.0	0.0	0.0	0.0	
0060008032	0.0	0.0	…	0.0	0.0	0.0	0.0	0.0	
0060096195	0.0	0.0	…	0.0	0.0	0.0	0.0	0.0	
006016848X	0.0	0.0	…	0.0	0.0	0.0	0.0	0.0	
0060173289	0.0	0.0	…	0.0	0.0	0.0	0.0	0.0	

User-ID	278554	278582	278633	278771	278843
ISBN					
002542730X	0.0	0.0	0.0	0.0	0.0
0060008032	0.0	0.0	0.0	0.0	0.0
0060096195	9.0	0.0	0.0	0.0	0.0

```
006016848X     0.0     0.0     0.0     0.0     0.0
0060173289     0.0     0.0     0.0     0.0     9.0

[5 rows x 10164 columns]
```

Indexamos la matriz con el titulo del libro para buscar el libro por titulo en vez del ISBN

[243]:
```
df.index = df.join(df_books.set_index('ISBN'))['Book-Title']
df = df.sort_index()
df.head()
```

[243]:
```
User-ID              99      242     243     254     383     384     388     \
Book-Title
1984                    0.0     0.0     0.0     9.0     0.0     0.0     0.0
1st to Die: A Novel     0.0     0.0     0.0     0.0     0.0     0.0     0.0
1st to Die: A Novel     0.0     0.0     0.0     0.0     0.0     0.0     0.0
2nd Chance              0.0     0.0     0.0     0.0     0.0     0.0     0.0
2nd Chance              0.0     0.0     0.0     0.0     0.0     0.0     0.0

User-ID              408     424     446     …  278418  278506  278522  \
Book-Title                                   …
1984                    0.0     0.0     0.0  …     0.0     0.0     0.0
1st to Die: A Novel     0.0     0.0     0.0  …     0.0     0.0     0.0
1st to Die: A Novel     0.0     0.0     0.0  …     0.0     0.0     0.0
2nd Chance              0.0     0.0     0.0  …     0.0     0.0     0.0
2nd Chance              0.0     0.0     0.0  …     0.0     0.0     0.0

User-ID              278535  278536  278554  278582  278633  278771  278843
Book-Title
1984                    0.0     0.0     0.0     0.0     0.0     0.0     0.0
1st to Die: A Novel     0.0     0.0     0.0     0.0     0.0     0.0     0.0
1st to Die: A Novel    10.0     0.0     0.0     0.0     0.0     0.0     0.0
2nd Chance              0.0     0.0     0.0     0.0     0.0     0.0     0.0
2nd Chance              0.0     0.0     0.0     0.0     0.0     0.0     0.0

[5 rows x 10164 columns]
```

Probamos que exista un libro, en este caso Lord of the Rings. Usaremos el titulo que usa el dataset para el libro `16 Lighthouse Road` para probar el modelo ya que lo tengamos entrenado

[244]:
```
df.loc["The Fellowship of the Ring (The Lord of the Rings, Part 1)"][:5]
```

[244]:
```
User-ID
99     0.0
242    0.0
243    0.0
254    0.0
383    0.0
```

```
Name: The Fellowship of the Ring (The Lord of the Rings, Part 1), dtype: float64
```

1.7 Modelo

1.7.1 Creacion del modelo

Usaremos un modelo de KNN para hacer las recomendaciones. La distancia sera conseguida en base la similaridad de las criticas de ese libro con las criticas de otros libros. Si los mismos usuarios le dieron calificaciones similares a dos libros diremos que ese segundo libro deberia ser recomendado a los usuarios que les gusto el primero

[245]:
```
model = NearestNeighbors(metric='cosine')
model.fit(df.values)
```

[245]:
```
NearestNeighbors(metric='cosine')
```

[246]:
```
title = 'The Fellowship of the Ring (The Lord of the Rings, Part 1)'
df.loc[title].shape
```

[246]:
```
(10164,)
```

1.8 Conseguir recomendaciones

Usaremos un K de 6 para esta prueba

[247]:
```
distance, indice = model.kneighbors([df.loc[title].values], n_neighbors=6)

print(distance)
print(indice)
```

```
[[0.         0.55566748 0.58906449 0.73726605 0.90290779 0.90773266]]
[[530 624 657 551 297 183]]
```

[248]:
```
df.iloc[indice[0]].index.values
```

[248]:
```
array(['The Fellowship of the Ring (The Lord of the Rings, Part 1)',
       'The Return of the King (The Lord of the Rings, Part 3)',
       'The Two Towers (The Lord of the Rings, Part 2)',
       'The Hobbit : The Enchanting Prelude to The Lord of the Rings',
       'Lasher: Lives of the Mayfair Witches (Lives of the Mayfair Witches)',
       "Ender's Game (Ender Wiggins Saga (Paperback))"], dtype=object)
```

Metemos las recomendaciones a un DataFrame de pandas con el titulo recomendado y la distancia del libro original

[249]:
```
pd.DataFrame({
    'title'   : df.iloc[indice[0]].index.values,
    'distance': distance[0]
}) \
.sort_values(by='distance', ascending=False)
```

[249]:
```
                                                                title  \
5                          Ender's Game (Ender Wiggins Saga (Paperback))
4  Lasher: Lives of the Mayfair Witches (Lives of the Mayfair Witches)
3         The Hobbit : The Enchanting Prelude to The Lord of the Rings
2                       The Two Towers (The Lord of the Rings, Part 2)
1                The Return of the King (The Lord of the Rings, Part 3)
0            The Fellowship of the Ring (The Lord of the Rings, Part 1)

   distance
5  0.907733
4  0.902908
3  0.737266
2  0.589064
1  0.555667
0  0.000000
```

Creamos una funcion para simplificar el proceso de conseguir `k` recomendaciones para un titulo

[250]:
```
def get_recommends(title = "", k = 10):
  try:
    book = df.loc[title]
  except KeyError as e:
    return None

  distance, indice = model.kneighbors([book.values], n_neighbors=k)

  recommended_books = pd.DataFrame({
      'title'   : df.iloc[indice[0]].index.values,
      'distance': distance[0]
    }) \
    .sort_values(by='distance', ascending=False) \
    .head(5).values

  return [title, recommended_books]
```

1.9 Revisar predicciones

[251]:
```
books = get_recommends("Ender's Game (Ender Wiggins Saga (Paperback))")
print(books)
```

```
["Ender's Game (Ender Wiggins Saga (Paperback))", array([['Stranger in a Strange
Land (Remembering Tomorrow)',
        0.915838990066355],
       ['Watership Down', 0.9156888070393626],
       ['Life of Pi', 0.9155732309313093],
       ['Dune (Remembering Tomorrow)', 0.9151667349457187],
       ['The Stand: Complete and Uncut', 0.91451746093396]], dtype=object)]
```

[252]:
```
books = get_recommends("Dune (Remembering Tomorrow)")
print(books)
```

```
['Dune (Remembering Tomorrow)', array([['Siddhartha', 0.9222937976819585],
       ['The Hobbit : The Enchanting Prelude to The Lord of the Rings',
        0.9206880350806068],
       ["The Handmaid's Tale", 0.9204704161591308],
       ['Dreamcatcher', 0.9198527770815851],
       ['A Confederacy of Dunces (Evergreen Book)', 0.9194667585125214]],
      dtype=object)]
```

[253]:
```
books = get_recommends("American Gods")
print(books)
```

```
['American Gods', array([["The Bonesetter's Daughter", 0.9317207157574144],
       ['A Prayer for Owen Meany', 0.9296579620815202],
       ["Lamb : The Gospel According to Biff, Christ's Childhood Pal",
        0.9238304529803233],
       ['Love in the Time of Cholera (Penguin Great Books of the 20th Century)',
        0.9222952032805442],
       ["Slaughterhouse Five or the Children's Crusade: A Duty Dance With
Death",
        0.9215509071275683]], dtype=object)]
```

[254]:
```
books = get_recommends("The Fellowship of the Ring (The Lord of the Rings, Part␣
↪1)")
print(books)
```

```
['The Fellowship of the Ring (The Lord of the Rings, Part 1)', array([['Harry
Potter and the Chamber of Secrets (Book 2)',
        0.9193624743457263],
       ['Harry Potter and the Goblet of Fire (Book 4)',
        0.9179158381435988],
       ["Harry Potter and the Sorcerer's Stone (Harry Potter (Paperback))",
        0.913950465507235],
       ['The Tale of the Body Thief (Vampire Chronicles (Paperback))',
        0.9108658383837844],
       ["Ender's Game (Ender Wiggins Saga (Paperback))",
        0.9077326601838696]], dtype=object)]
```

1.10 Evaluaciones

La manera de evaluar este proyecto será de acuerdo a dos criterios: - Existen recomendaciones para mis libros favoritos - Me son útiles las recomendaciones Usamos estos dos criterios porque la meta de este proyecto es darme recomendaciones basadas en mis libros favoritos.

1.10.1 Métricas

Usaremos un export de mi librería de Goodreads que entre otras columnas tiene las siguientes: - ISBN - Título - Bookshelves Las primeras dos columnas se usarán para buscar libros con el modelo KNN, la tercer columna de Bookshelves tiene las categorías que yo les he dado. Una de estas categorías es la de "favorites", la cual indica que yo personalmente marqué este libro como uno de mis favoritos. Usaremos estos libros para ver cuantos de ellos tienen recomendaciones.

[255]:
```
df_books = pd.read_csv('../data/raw/goodreads_library_export.csv')
df_books.head()
```

[255]:

	Book Id	Title	Author	Author l-f	Additional Authors	ISBN	ISBN13	My Rating	Average Rating	Publisher	…	Date Read	Date Added	Bookshelves	Bookshelves with positions	Exclusive Shelf	My Review	Spoiler	Private Notes	Read Count	Owned Copies
0	62015945	Yours Truly (Part of Your World, #2)	Abby Jimenez	Jimenez, Abby	NaN	="1538704412"	="9781538704417"	0	4.32	Forever	…	2024/11/03	2024/10/04	NaN	NaN	read	NaN	NaN	NaN	1	0
1	195834342	Just for the Summer	Abby Jimenez	Jimenez, Abby	NaN	=""	=""	0	4.39	Forever	…	NaN	2024/11/03	currently-reading	currently-reading (#22)	currently-reading	NaN	NaN	NaN	1	0
2	220422953	Fighting Past Pain (Heavenly Chaos Book 2)	Daniel Schinhofen	Schinhofen, Daniel	NaN	=""	=""	0	4.73	NaN	…	NaN	2024/11/02	currently-reading	currently-reading (#21)	currently-reading	NaN	NaN	NaN	1	0
3	216035586	Saviors (Quest Academy #3)	Brian J. Nordon	Nordon, Brian J.	NaN	=""	=""	5	4.72	The Legion Publishers ltd	…	2024/11/02	2024/11/01	NaN	NaN	read	NaN	NaN	NaN	1	0
4	203274611	Scavengers (Quest Academy #2)	Brian J. Nordon	Nordon, Brian J.	NaN	=""	=""	5	4.53	The Legion Publishers Ltd	…	2024/11/01	2024/10/31	NaN	NaN	read	NaN	NaN	NaN	1	0

5 rows x 24 columns

[256]:
```
df_books = df_books[['ISBN', 'Title', 'Bookshelves']]
df_books.head()
```

[256]:
```
            ISBN                                        Title  \
0  ="1538704412"        Yours Truly (Part of Your World, #2)
1            =""                            Just for the Summer
2            =""  Fighting Past Pain (Heavenly Chaos Book 2)
3            =""                   Saviors (Quest Academy #3)
4            =""                Scavengers (Quest Academy #2)

         Bookshelves
0                NaN
1  currently-reading
2  currently-reading
3                NaN
4                NaN
```

[257]:
```
df_books.dropna(inplace=True)
df_books = df_books[df_books['Bookshelves'].str.contains('favorites')]
df_books['ISBN'] = df_books['ISBN'].str.replace('="', '').str.replace('"', '')
df_books.head()
```

[257]:
```
           ISBN  \
30   0451465105
63
210
215
222

                                                    Title  \
30                  The Thousand Names (The Shadow Campaigns, #1)
63                                           Dreamer's Throne 2
210                            Sex, Death, and Money: Season 1
215            The Path of Ascension 3 (The Path of Ascension #3)
222  Homicidal Aliens are Invading and All I Got is This Stat Menu

                   Bookshelves
30                   favorites
63                   favorites
210                  favorites
215          litrpg, favorites
222  favorites, supers, litrpg
```

[258]:
```
df_books['Recommendations'] = df_books['Title'].apply(lambda x:␣
 ↪get_recommends(x, k=10))
df_books.groupby('Recommendations').count()
```

[258]:
```
Empty DataFrame
Columns: [ISBN, Title, Bookshelves]
Index: []
```

1.11 Resultados

No conseguimos ninguna recomendación para los libros que tengo marcados como mis favoritos. Esto nos deja con dos conclusiones: 1. El dataset no tiene suficiente variedad de libros 2. Mis gustos lectores son demasiado de nicho como para usar este modelo 3. Debemos de consolidar los títulos de las diferentes ediciones en un sólo índice para tener más críticas por libro