movie-recomand

June 28, 2023

1 Movie Recomendation system

Importing the packages

[60]:
```
import pandas as pd
import matplotlib.pyplot as plt
import math
import numpy as np
import difflib
from sklearn.metrics.pairwise import cosine_similarity
from sklearn.feature_extraction.text import TfidfVectorizer
```

Reading the file

[61]:
```
data=pd.read_csv("movies.csv")
```

Details of File

[62]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4803 entries, 0 to 4802
Data columns (total 24 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   index                 4803 non-null   int64
 1   budget                4803 non-null   int64
 2   genres                4775 non-null   object
 3   homepage              1712 non-null   object
 4   id                    4803 non-null   int64
 5   keywords              4391 non-null   object
 6   original_language     4803 non-null   object
 7   original_title        4803 non-null   object
 8   overview              4800 non-null   object
 9   popularity            4803 non-null   float64
 10  production_companies  4803 non-null   object
 11  production_countries  4803 non-null   object
 12  release_date          4802 non-null   object
 13  revenue               4803 non-null   int64
```

```
 14  runtime               4801 non-null   float64
 15  spoken_languages      4803 non-null   object
 16  status                4803 non-null   object
 17  tagline               3959 non-null   object
 18  title                 4803 non-null   object
 19  vote_average          4803 non-null   float64
 20  vote_count            4803 non-null   int64
 21  cast                  4760 non-null   object
 22  crew                  4803 non-null   object
 23  director              4773 non-null   object
dtypes: float64(3), int64(5), object(16)
memory usage: 900.7+ KB
```

Getting the req_data from data set

[63]:
```
req_data=data.iloc[:,[2,5,8,9,12,14,15,19,21,23]]
```

Converting all data to string format and filling the null values

[64]:
```
for i in req_data.keys():
  req_data[i]=req_data[i].astype("string")
  req_data[i]=req_data[i].fillna("")
```

```
<ipython-input-64-5eab926dcb66>:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  req_data[i]=req_data[i].astype("string")
<ipython-input-64-5eab926dcb66>:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  req_data[i]=req_data[i].fillna("")
```

[65]:
```
req_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4803 entries, 0 to 4802
Data columns (total 10 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   genres            4803 non-null   string
 1   keywords          4803 non-null   string
 2   overview          4803 non-null   string
 3   popularity        4803 non-null   string
```

```
 4   release_date      4803 non-null   string
 5   runtime           4803 non-null   string
 6   spoken_languages  4803 non-null   string
 7   vote_average      4803 non-null   string
 8   cast              4803 non-null   string
 9   director          4803 non-null   string
dtypes: string(10)
memory usage: 375.4 KB
```

converting the secondary factors into a single attribute of string

[66]:
```
#movie recomdations based on genres,release_date,cast,director,and remain
string=req_data['keywords']+" "+req_data['director']+" "+req_data['overview']+"␣
 ↪"+req_data['popularity']+" "+req_data['runtime']+"␣
 ↪"+req_data['spoken_languages']+" "+req_data['vote_average']
```

- Creation of individual normalised vector for primary factors
- similarly common normalised vector for secondary factors

[67]:
```
vector1=TfidfVectorizer().fit_transform(req_data["genres"])
vector2=TfidfVectorizer().fit_transform(req_data["release_date"]+"␣
 ↪"+req_data["popularity"])
vector3=TfidfVectorizer().fit_transform(req_data["cast"])
vector4=TfidfVectorizer().fit_transform(string)
```

Generation of similarity scores for those vectors

[68]:
```
similar_genres=cosine_similarity(vector1)
similar_release_date=cosine_similarity(vector2)
similar_cast=cosine_similarity(vector3)
similar_other_factors=cosine_similarity(vector4)
```

Reading input from user and then matching it with the closest ones

[69]:
```
while (True):
  movie=input("Enter Movie Name:")
  movie_name=difflib.get_close_matches(movie,data["title"])
  if(len(movie_name)==0):
    print("No recomendations found")
    print("Try with other name")
  else:
    break
val=data[data.title==movie_name[0]]["index"].values[0]
movie=data.loc[[int(val)],["title"]].values[0][0]
genres=data.loc[[int(val)],["genres"]].values[0][0]
cast=data.loc[[int(val)],["cast"]].values[0][0]
release_date=data.loc[[int(val)],["release_date"]].values[0][0]
print("Movie Name:",movie)
print("genres:",genres)
```

```
print("cast:",cast)
print("release date:",release_date)
```

```
Enter Movie Name:avtar
Movie Name: Avatar
genres: Action Adventure Fantasy Science Fiction
cast: Sam Worthington Zoe Saldana Sigourney Weaver Stephen Lang Michelle
Rodriguez
release date: 2009-12-10
```

Getting similarity scores for the given movie

[70]:
```
genres_score=list(enumerate(similar_genres[val]))
cast_score=list(enumerate(similar_cast[val]))
release_score=list(enumerate(similar_release_date[val]))
others_score=list(enumerate(similar_other_factors[val]))
```

Recomending movies based on the User Zone

[71]:
```
count=0
recomanded_movies=[]
sorted_recomandations=sorted(genres_score,key=lambda x:x[1],reverse=True)
for i in sorted_recomandations:
  index=i[0]
  recomanded_movies.append(data[data.index==index]['title'].values[0])
  count+=1
  if(count>(10)):
    break
for i in recomanded_movies:
  va=data[data.title==i]["homepage"].values[0]
  if pd.isnull(va):
    print(i)
  else:
    print(i,va)
```

```
Avatar http://www.avatarmovie.com/
Superman Returns http://www.superman.com
Man of Steel http://www.manofsteel.com/
X-Men: Days of Future Past http://www.x-menmovies.com/
Jupiter Ascending http://www.jupiterascending.com
The Wolverine http://www.thewolverinemovie.com
Superman
Superman II
Beastmaster 2: Through the Portal of Time
Teenage Mutant Ninja Turtles http://www.teenagemutantninjaturtlesmovie.com
Mystery Men
```

Recomending movies based on user Favorite characters

[73]:
```
count=0
recomanded_movies=[]
sorted_recomandations=sorted(cast_score,key=lambda x:x[1],reverse=True)
for i in sorted_recomandations:
  index=i[0]
  recomanded_movies.append(data[data.index==index]['title'].values[0])
  count+=1
  if(count>(10)):
    break
for i in recomanded_movies:
  va=data[data.title==i]["homepage"].values[0]
  if pd.isnull(va):
    print(i)
  else:
    print(i,va)
```

```
Avatar http://www.avatarmovie.com/
Gettysburg
Out of the Furnace
Galaxy Quest
Imaginary Heroes http://www.sonypictures.com/classics/imaginary/site.html
Snow White: A Tale of Terror
The Words http://www.thewordsmovie.com/
Everest http://www.everestmovie.com/
Drumline
Get Over It
Vantage Point http://www.vantagepoint-movie.com/index.php
```

Recomending movies based on user interested time-zone

[74]:
```
count=0
recomanded_movies=[]
sorted_recomandations=sorted(release_score,key=lambda x:x[1],reverse=True)
for i in sorted_recomandations:
  index=i[0]
  recomanded_movies.append(data[data.index==index]['title'].values[0])
  count+=1
  if(count>(10)):
    break
for i in recomanded_movies:
  va=data[data.title==i]["homepage"].values[0]
  if pd.isnull(va):
    print(i)
  else:
    print(i,va)
```

```
Avatar http://www.avatarmovie.com/
A Shine of Rainbows
```

```
Life During Wartime http://www.ifcfilms.com/films/life-during-wartime-2
Chicago Overcoat
Invictus http://invictusmovie.warnerbros.com
Observe and Report
Oceans http://oceans-lefilm.com/
2012 http://www.sonypictures.com/movies/2012
Defendor
A Woman, a Gun and a Noodle Shop
Rocket Singh: Salesman of the Year
http://www.yashrajfilms.com/microsites/rocketsingh/rswebsite.html
```

Recomending movies that adds on interest

[75]:
```
count=0
recomanded_movies=[]
sorted_recomandations=sorted(others_score,key=lambda x:x[1],reverse=True)
for i in sorted_recomandations:
  index=i[0]
  recomanded_movies.append(data[data.index==index]['title'].values[0])
  count+=1
  if(count>10):
    break
for i in recomanded_movies:
  va=data[data.title==i]["homepage"].values[0]
  if pd.isnull(va):
    print(i)
  else:
    print(i,va)
```

```
Avatar http://www.avatarmovie.com/
Lifeforce
Moonraker http://www.mgm.com/view/movie/1292/Moonraker/
Gattaca
Gravity http://gravitymovie.warnerbros.com/
Cargo http://www.cargoderfilm.ch http://cargothemovie.com
Space Chimps http://www.spacechimpspower.com/
Apollo 18 http://apollo18movie.net/
Starship Troopers
Deep Impact
Alien https://www.facebook.com/alienanthology/
```