system-leveraging-machine-learning

July 17, 2024

0.1 Designing an Intelligent Movie Recommendation System Leveraging Machine Learning - Vignesh Prabhu

In this project, we developed a sophisticated movie recommendation system utilizing machine learning techniques. By leveraging collaborative filtering, content-based filtering, and hybrid approaches, our system intelligently predicts and suggests movies tailored to users' preferences. This enhances user experience and engagement by providing personalized movie recommendations.

Import dependencies

[1]:
```
import pandas as pd
import numpy as np
import difflib
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import cosine_similarity
```

Data Collection And Preprocessing

[2]:
```
#Loading The data to DataFrame
movies_data = pd.read_csv('/content/movies.csv')
```

[3]:
```
#To Print First 5 datas in dataset
movies_data.head()
```

[3]:
```
   index     budget                                    genres  \
0      0  237000000  Action Adventure Fantasy Science Fiction
1      1  300000000                  Adventure Fantasy Action
2      2  245000000                    Action Adventure Crime
3      3  250000000               Action Crime Drama Thriller
4      4  260000000          Action Adventure Science Fiction

                                       homepage      id  \
0                   http://www.avatarmovie.com/   19995
1  http://disney.go.com/disneypictures/pirates/     285
2   http://www.sonypictures.com/movies/spectre/  206647
3           http://www.thedarkknightrises.com/   49026
4         http://movies.disney.com/john-carter   49529
```

```
                                            keywords original_language  \
0  culture clash future space war space colony so…                en
1  ocean drug abuse exotic island east india trad…                en
2         spy based on novel secret agent sequel mi6                en
3  dc comics crime fighter terrorist secret ident…                en
4  based on novel mars medallion space travel pri…                en

                             original_title  \
0                                    Avatar
1  Pirates of the Caribbean: At World's End
2                                   Spectre
3                     The Dark Knight Rises
4                               John Carter

                                            overview  popularity  … runtime  \
0  In the 22nd century, a paraplegic Marine is di…  150.437577  …   162.0
1  Captain Barbossa, long believed to be dead, ha…  139.082615  …   169.0
2  A cryptic message from Bond’s past sends him o…  107.376788  …   148.0
3  Following the death of District Attorney Harve…  112.312950  …   165.0
4  John Carter is a war-weary, former military ca…   43.926995  …   132.0

                                    spoken_languages    status  \
0  [{"iso_639_1": "en", "name": "English"}, {"iso…  Released
1           [{"iso_639_1": "en", "name": "English"}]  Released
2  [{"iso_639_1": "fr", "name": "Fran\u00e7ais"},…  Released
3           [{"iso_639_1": "en", "name": "English"}]  Released
4           [{"iso_639_1": "en", "name": "English"}]  Released

                                          tagline  \
0                     Enter the World of Pandora.
1  At the end of the world, the adventure begins.
2                            A Plan No One Escapes
3                                  The Legend Ends
4            Lost in our world, found in another.

                                      title vote_average vote_count  \
0                                    Avatar          7.2      11800
1  Pirates of the Caribbean: At World's End          6.9       4500
2                                   Spectre          6.3       4466
3                     The Dark Knight Rises          7.6       9106
4                               John Carter          6.1       2124

                                                cast  \
0  Sam Worthington Zoe Saldana Sigourney Weaver S…
1  Johnny Depp Orlando Bloom Keira Knightley Stel…
2  Daniel Craig Christoph Waltz L\u00e9a Seydoux …
```

```
3   Christian Bale Michael Caine Gary Oldman Anne …
4   Taylor Kitsch Lynn Collins Samantha Morton Wil…

                                                crew          director
0   [{'name': 'Stephen E. Rivkin', 'gender': 0, 'd…      James Cameron
1   [{'name': 'Dariusz Wolski', 'gender': 2, 'depa…     Gore Verbinski
2   [{'name': 'Thomas Newman', 'gender': 2, 'depar…         Sam Mendes
3   [{'name': 'Hans Zimmer', 'gender': 2, 'departm…  Christopher Nolan
4   [{'name': 'Andrew Stanton', 'gender': 2, 'depa…     Andrew Stanton

[5 rows x 24 columns]
```

[4]:
```
#To check Rows and Columns
movies_data.shape
```

[4]:
```
(4803, 24)
```

Feature Selection

[5]:
```
#Relevent Features for Recommendation
selected_features = ['genres','keywords','tagline','cast','director']
print(selected_features)
```

```
['genres', 'keywords', 'tagline', 'cast', 'director']
```

[6]:
```
#Replacing Null Values With Null String
for feature in selected_features:
  movies_data[feature] = movies_data[feature].fillna('')
```

[7]:
```
#Combining Selected Features
combined_features = movies_data['genres']+' '+movies_data['keywords']+'␣
↪'+movies_data['tagline']+' '+movies_data['cast']+' '+movies_data['director']
```

[8]:
```
print(combined_features)
```

```
0       Action Adventure Fantasy Science Fiction cultu…
1       Adventure Fantasy Action ocean drug abuse exot…
2       Action Adventure Crime spy based on novel secr…
3       Action Crime Drama Thriller dc comics crime fi…
4       Action Adventure Science Fiction based on nove…
                              …
4798    Action Crime Thriller united states\u2013mexic…
4799    Comedy Romance  A newlywed couple's honeymoon …
4800    Comedy Drama Romance TV Movie date love at fir…
4801      A New Yorker in Shanghai Daniel Henney Eliza…
4802    Documentary obsession camcorder crush dream gi…
Length: 4803, dtype: object
```

Converting Text Data to feature Vectors

[9]:
```
vectorizer= TfidfVectorizer()
```

[10]:
```
feature_vectors = vectorizer.fit_transform(combined_features)
```

[11]:
```
print(feature_vectors)
```

```
  (0, 2432)	0.17272411194153
  (0, 7755)	0.1128035714854756
  (0, 13024)	0.1942362060108871
  (0, 10229)	0.16058685400095302
  (0, 8756)	0.22709015857011816
  (0, 14608)	0.15150672398763912
  (0, 16668)	0.19843263965100372
  (0, 14064)	0.20596090415084142
  (0, 13319)	0.2177470539412484
  (0, 17290)	0.20197912553916567
  (0, 17007)	0.23643326319898797
  (0, 13349)	0.15021264094167086
  (0, 11503)	0.27211310056983656
  (0, 11192)	0.09049319826481456
  (0, 16998)	0.1282126322850579
  (0, 15261)	0.07095833561276566
  (0, 4945)	0.24025852494110758
  (0, 14271)	0.21392179219912877
  (0, 3225)	0.24960162956997736
  (0, 16587)	0.12549432354918996
  (0, 14378)	0.33962752210959823
  (0, 5836)	0.1646750903586285
  (0, 3065)	0.22208377802661425
  (0, 3678)	0.21392179219912877
  (0, 5437)	0.1036413987316636
  :	:
  (4801, 17266)	0.2886098184932947
  (4801, 4835)	0.24713765026963996
  (4801, 403)	0.17727585190343226
  (4801, 6935)	0.2886098184932947
  (4801, 11663)	0.21557500762727902
  (4801, 1672)	0.1564793427630879
  (4801, 10929)	0.13504166990041588
  (4801, 7474)	0.11307961713172225
  (4801, 3796)	0.3342808988877418
  (4802, 6996)	0.5700048226105303
  (4802, 5367)	0.22969114490410403
  (4802, 3654)	0.262512960498006
  (4802, 2425)	0.24002350969074696
  (4802, 4608)	0.24002350969074696
  (4802, 6417)	0.21753405888348784
```

```
  (4802, 4371)	0.1538239182675544
  (4802, 12989)	0.1696476532191718
  (4802, 1316)	0.1960747079005741
  (4802, 4528)	0.19504460807622875
  (4802, 3436)	0.21753405888348784
  (4802, 6155)	0.18056463596934083
  (4802, 4980)	0.16078053641367315
  (4802, 2129)	0.3099656128577656
  (4802, 4518)	0.16784466610624255
  (4802, 11161)	0.17867407682173203
```

Cosine similarity

[12]:
```python
cosine_similarity= cosine_similarity(feature_vectors)
```

[13]:
```python
print(cosine_similarity)
```

```
[[1.         0.07219487 0.037733   … 0.         0.         0.        ]
 [0.07219487 1.         0.03281499 … 0.03575545 0.         0.        ]
 [0.037733   0.03281499 1.         … 0.         0.05389661 0.        ]
 …
 [0.         0.03575545 0.         … 1.         0.         0.02651502]
 [0.         0.         0.05389661 … 0.         1.         0.        ]
 [0.         0.         0.         … 0.02651502 0.         1.        ]]
```

[14]:
```python
cosine_similarity.shape
```

[14]:
```
(4803, 4803)
```

[15]:
```python
#Getting Input
movie_name = input('Enter Your Favourite Movie Name : ')
```

```
Enter Your Favourite Movie Name : Bat Man
```

[ ]:
```python
#Creating  a list with all movie names given in the dataset
list_of_all_titles = movies_data['title'].tolist()
print(list_of_all_titles)
```

[17]:
```python
#Finding The close match for user Input
find_close_match = difflib.get_close_matches(movie_name, list_of_all_titles)
print(find_close_match)
```

```
['Batman', 'Batman', 'Rain Man']
```

[18]:
```python
close_match = find_close_match[0]
print(close_match)
```

```
Batman
```

[19]:
```
#Finding The Index of the  movie with title
index_of_the_movie = movies_data[movies_data.title == close_match]['index'].
 ↪values[0]
print(index_of_the_movie)
```

```
1359
```

[ ]:
```
#Getting Similar Movies
similarity_score = list(enumerate(cosine_similarity[index_of_the_movie]))
print(similarity_score)
```

[21]:
```
len(similarity_score)
```

[21]:
```
4803
```

[ ]:
```
#sorting the movies based on their similarty score
Sorted_similar_movies = sorted(similarity_score, key = lambda x:x[1], reverse =␣
 ↪True)
print(Sorted_similar_movies)
```

[23]:
```
#Print the name of similar movies based on index
print('Movies suggested for you : \n')

i = 1
for movie in Sorted_similar_movies:
  index = movie[0]
  title_from_index = movies_data[movies_data.index==index]['title'].values[0]
  if (i<10):
    print(i, '.',title_from_index)
    i+=1
```

```
Movies suggested for you :

1 . Batman
2 . Batman Returns
3 . Batman & Robin
4 . The Dark Knight Rises
5 . Batman Begins
6 . The Dark Knight
7 . A History of Violence
8 . Superman
9 . Beetlejuice
```

Building Movie Recommendation System

[24]:
```
#Getting input
movie_name = input('Enter Your Favourite Movie Name : ')
```

```
list_of_all_titles = movies_data['title'].tolist()

find_close_match = difflib.get_close_matches(movie_name, list_of_all_titles)

close_match = find_close_match[0]

index_of_the_movie = movies_data[movies_data.title == close_match]['index'].
  ↪values[0]

similarity_score = list(enumerate(cosine_similarity[index_of_the_movie]))

Sorted_similar_movies = sorted(similarity_score, key = lambda x:x[1], reverse =␣
  ↪True)

print('Movies suggested for you : \n')

i = 1
for movie in Sorted_similar_movies:
  index = movie[0]
  title_from_index = movies_data[movies_data.index==index]['title'].values[0]
  if (i<10):
    print(i, '.',title_from_index)
    i+=1
```

```
Enter Your Favourite Movie Name : Captain America
Movies suggested for you :

1 . Captain America: Civil War
2 . Captain America: The Winter Soldier
3 . Avengers: Age of Ultron
4 . The Avengers
5 . Iron Man 2
6 . Captain America: The First Avenger
7 . Iron Man 3
8 . Iron Man
9 . Thor: The Dark World
```

Our movie recommendation system successfully employs advanced machine learning techniques to deliver personalized movie suggestions. By integrating various filtering methods, it enhances user satisfaction and engagement, demonstrating the effectiveness of machine learning in improving recommendation accuracy.

0.2 Thank You!