Crimson Publishers
Wings to the Research

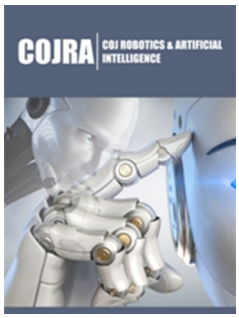
Mini Review

Building a Popularity Recommender for Movie Lens Dataset Using Python

Marappan R*

Senior Assistant Professor, India

ISSN: 2832-4463



***Corresponding author:** Marappan R, School of Computing, SASTRA Deemed University, Thanjavur, India

Submission: June 14, 2022
Published: October 17, 2022

Volume 2 - Issue 3

How to cite this article: Marappan R*. Building a Popularity Recommender for Movie Lens Dataset Using Python. COJ Rob Artificial Intel. 2(3). COJRA. 000540. 2022. DOI: 10.31031/COJRA.2022.02.000540

Abstract

There are a lot of recommendation systems developed for different artificial intelligence and machine learning applications. The recommendation systems should provide a better recommendation with minimal computing time. This research focuses on how to build the popularity-based recommender system for the MovieLens dataset using Python with its analysis. The proposed recommender model obtains the top n recommendations in less computing time (less than 5 seconds). The accuracy of the proposed model lies in between 90% to 97% and the expected absolute error lies between 3% to 12% compared to the other models.

Keywords: Recommendation systems; Recommender systems; Artificial intelligence; Machine learning; Popularity recommender

Introduction

Recently many recommender systems are developed for information filtering [1]. The learners or users are expected to get a better recommendation for their interests in various applications. This research focuses on developing a popularity-based recommender using Python.

Dataset Structure

For the implementation purpose, the Movie Lens dataset is considered with ratings.csv and movies.csv files [2-4]. The fields in movies.csv are movieId, title, and genre. The unique id for each movie is defined in movieId. The name of the movie is defined in the title field. The genre of the movie is defined in the genre field. The fields defined in ratings.csv are userId, movieId, rating, and timestamp. The users who are rated movies are defined in the unique userId. The movie ratings of the user are defined in the rating field. The time of rating a movie is defined in the timestamp field.

Python Model

This section focuses on the Python recommendation model for the Movie Lens dataset. The structure of ratings.csv and movies.csv are sketched in Figure 1 & 2 respectively. The combined structure of these files is shown in Figure 3. The complete model is defined as follows:

```
#import all necessary libraries: os, numpy, pandas, matplotlib.pyplot
plt.style.use('seaborn-bright')
%Matplotlib inline
#Change directory to the folder where data files are present
#This step is not necessary if the data files and jupyter notebook are in same folder
```

```
os.chdir("E:\MovieLens")
#Import ratings file in a pandas data frame
ratings_data=pd.read_csv("ratings.csv")
ratings_data.head()
movie_names = pd.read_csv("movies.csv")
movie_names.head()
movie_data=pd.merge(ratings_data,movie_names,on='movieId')
movie_data.head()
```

	userId	movieId	rating	timestamp
0	1	1	4.0	964982703
1	1	3	4.0	964981247
2	1	6	4.0	964982224
3	1	47	5.0	964983815
4	1	50	5.0	964982931

Figure 1: Structure of ratings csv.

	movieId	title	genres
0	1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy
1	2	Jumanji (1995)	Adventure|Children|Fantasy
2	3	Grumpier Old Men (1995)	Comedy|Romance
3	4	Waiting to Exhale (1995)	Comedy|Drama|Romance
4	5	Father of the Bride Part II (1995)	Comedy

Figure 2: Structure of movies csv.

	userId	movieId	rating	timestamp	title	genres
0	1	1	4.0	964982703	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy
1	5	1	4.0	847434962	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy
2	7	1	4.5	1106635946	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy
3	15	1	2.5	1510577970	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy
4	17	1	4.5	1305696483	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy

Figure 3: Combined structure of ratings.csv & movies.csv with movieId as the primary key.

Model Analysis

This section focuses on the analysis of the constructed model. The ratings and the total ratings are shown in figure 4. The ratings with the number of movies are sketched in figure 5. The proposed recommender model obtains the top n recommendations in less computing time (less than 5 seconds). The accuracy of the proposed model lies in between 90% to 97% and the expected absolute error lies between 3% to 12% compared to the other models. The steps in Python model development for popularity-based recommendation are as follows:

1. Define the packages: os, numpy and pandas
2. The working directory is to be changed to the dataset folder.
3. Read the information from the ratings.
4. Read the information from the movies file.
5. Merge ratings_data & movie_names using the pandas built-in function.
6. Construct the data frame 'movie_data' and print it.
7. Plot a horizontal bar graph using matplotlib library to get an overview of data.
8. Plot a bar graph to sketch the total number of reviews for each movie individually.
9. Arrange the titles in the order to recommend top-rating movies.

title	rating	total number of ratings
'71 (2014)	4.0	1
'Hellboy': The Seeds of Creation (2004)	4.0	1
'Round Midnight (1986)	3.5	2
'Salem's Lot (2004)	5.0	1
'Til There Was You (1997)	4.0	2

Figure 4: Rating and total ratings.

```
plt.figure(figsize =(10, 4))
ax=plt.barh(trend['rating'].round(),trend['total number of ratings'],color='b')
plt.show()
```

Figure 5

Conclusions & Future Work

More users may review and rate some movies. To have a better recommendation, new rules should be added for better popularity prediction of a movie. In addition, the newer ones may be better than the existing ones. In these situations, more weight will be included in the rate of newer movies to bring to the recommendation list.

In the future, new recommenders will be developed using hybrid, cluster-based and specific features based methods further to reduce the computing time of the proposed recommender [5-7]. The recommendation heuristics and strategies may also be mixed with the evolutionary operators such as the genetic operators with the advanced local search procedures and particle swarm optimization strategies to recommend the better items based on the user interests and profile characteristics [8-10]. The recommender system may also be modified using graph and input based features in the future to provide better suggestions [11-12] (Figures 6 to 8).

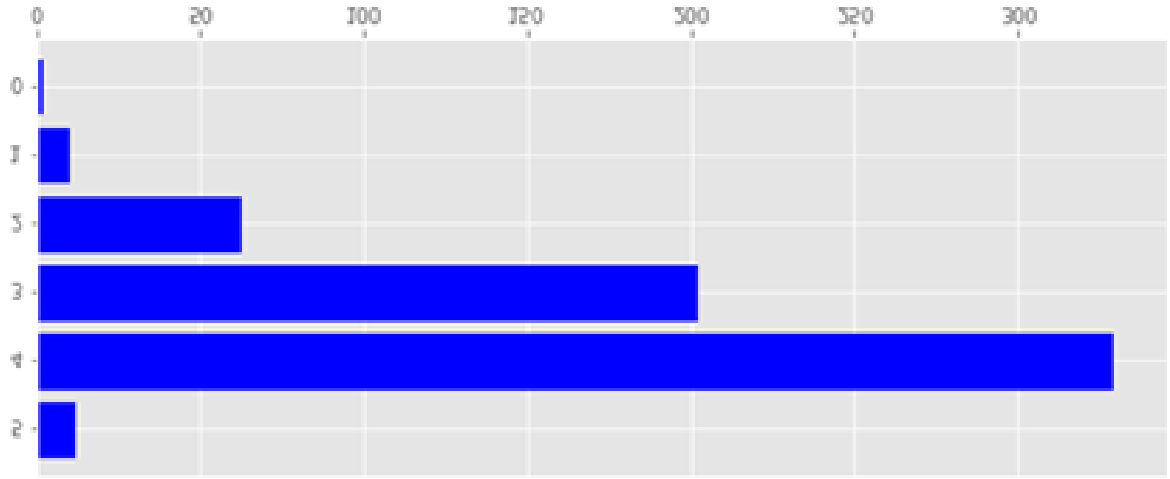


Figure 6: Ratings with number of movies.

```
plt.figure(figsize =(10, 4))
ax=plt.subplot()
ax.bar(trend.head(25).index,trend['total number of ratings'].head(25),color='b')
ax.set_xticklabels(trend.index,rotation=40,fontsize='12',horizontalalignment="right")
ax.set_title("Total Number of reviews for each movie")
plt.show()
```

Figure 7

title	
Karlson Returns (1970)	5.0
Winter in Prostokvashino (1984)	5.0
My Love (2006)	5.0
Sorority House Massacre II (1990)	5.0
Winnie the Pooh and the Day of Concern (1972)	5.0
Sorority House Massacre (1986)	5.0
Bill Hicks: Revelations (1993)	5.0
My Man Godfrey (1957)	5.0
Hellbenders (2012)	5.0
In the blue sea, in the white foam. (1984)	5.0
Won't You Be My Neighbor? (2018)	5.0
Red Sorghum (Hong gao liang) (1987)	5.0
Love Exposure (Ai No Mukidashi) (2008)	5.0
My Sassy Girl (Yeopgijeogin geunyeo) (2001)	5.0
The Love Bug (1997)	5.0
Ballad of Narayama, The (Narayama bushiko) (1983)	5.0
Heidi Fleiss: Hollywood Madam (1995)	5.0
Louis Theroux: Law & Disorder (2008)	5.0
Winnie the Pooh Goes Visiting (1971)	5.0
In the Realm of the Senses (Ai no corrida) (1976)	5.0
Winnie Pooh (1969)	5.0
Ex Drummer (2007)	5.0
Tom Segura: Mostly Stories (2016)	5.0
Tom and Jerry: A Nutcracker Tale (2007)	5.0
A Plasticine Crow (1981)	5.0
Tom and Jerry: Shiver Me Whiskers (2006)	5.0
Cosmic Scrat-tastrophe (2015)	5.0
Delirium (2014)	5.0
Lumberjack Man (2015)	5.0
Loving Vincent (2017)	5.0

Figure 8

References

1. Bertani RM, Bianchi AC, Reali Costa AH (2020) Combining novelty and popularity on personalised recommendations via user profile learning. Expert Systems With Applications 146: p. 113149.
2. Zhang F, Gong T, Lee VE, Zhao G, Rong C, et al. (2016) Fast algorithms to evaluate collaborative filtering recommender systems. Knowl Based Syst 96: 96-103.
3. Syed MA, Rakesh KL, Gopal KN, Rabindra KB (2018) Movie recommendation system using genome tags and content-based filtering. In Advances in Data and Information Sciences 38: 85-94.
4. Harper FM, Konstan JA (2015) The movielens datasets: History and context. ACM Trans Interact Intell Syst 5(4): 1-19.
5. Bhaskaran S, Marappan R (2021) Design and analysis of an efficient machine learning based hybrid recommendation system with enhanced density-based spatial clustering for digital e-learning applications. Complex Intell Syst.
6. Bhaskaran S, Marappan R, Santhi B (2021) Design and analysis of a cluster-based intelligent hybrid recommendation system for e-learning applications. Mathematics 9(2): 197.
7. Bhaskaran S, Marappan R, Santhi B (2020) Design and comparative analysis of new personalized recommender algorithms with specific features for large scale datasets. Mathematics 8(7): 1106.
8. Marappan R, Sethumadhavan G (2021) Solving graph coloring problem using divide and conquer-based turbulent particle swarm optimization. Arab J Sci Eng 47: 9695-9712.
9. Marappan R, Sethumadhavan G (2020) Complexity analysis and stochastic convergence of some well-known evolutionary operators for solving graph coloring problem. Mathematics 8(3): 303.
10. Marappan R, Sethumadhavan G (2018) Solution to graph coloring using genetic and tabu search procedures. Arab J Sci Eng 43: 525-542.
11. Lee C, Han D, Han K, Yi M, (2022) Improving graph-based movie recommender system using cinematic experience. Appl Sci 12(3): 1493.
12. An H, Kim D, Lee K, Moon N (2021) MovieDIRec: Drafted-input-based recommendation system for movies. Appl Sci 11(21): 10412.