

**Bishop’s University**

Winter 2023

CS 504 - Programming Languages for Data Analysis

Building Movie Recommender Systems Using Machine Learning Techniques on MovieLens Dataset

|  |  |
| --- | --- |
| Name | Student ID |
| Henry Oscar Perez Guzman | 002 28 0127 |
| Shahriar Minaei Jalil | 002 30 1723 |

## Abstract

The aim of this project is to build a movie recommender system using the Movielens dataset, which consists of ratings and reviews of movies by users. The system will use machine learning algorithms such as KNN, SVD, and deep learning models that incorporate NLP techniques and NN architecture to suggest movies to users based on similar users and for queries specific to genre, user, movie, rating, and popularity. The Pandas library in Python will be used to manipulate and analyze the data.

## Introduction

In today's world, people have access to a vast amount of content, and it can be challenging to decide what to watch or listen to next. Recommender systems are a way to solve this problem by suggesting content to users based on their past behavior or preferences. One such example is the movie recommender system, which suggests movies to users based on their preferences or behavior.

In this project, we will be building a movie recommender system using the Movielens dataset, which contains over 100,000 movie ratings and reviews by users. We will be using machine learning algorithms like KNN, SVD, and deep learning models that incorporate NLP techniques and NN architecture to suggest movies to users based on their preferences. The system will also allow users to make queries specific to genre, user, movie, rating, and popularity, making the movie selection process more personalized and efficient.

The Pandas library in Python will be used to manipulate and analyze to understand the data better. The project aims to provide an overview of how machine learning algorithms and NLP techniques and other techniques can be used to build a personalized movie recommender system that can enhance user experience and help them discover new movies that match their preferences.

## Recommender System Overview

A recommender system is a subclass of information filtering system that seeks to predict the "rating" or "preference" a user would give to an item. Recommender systems are utilized in a variety of areas including movies, music, news, social tags, and products in general. Recommender systems typically produce a list of recommendations and there are few ways in which it can be done. Two of the most popular ways are through collaborative filtering or through content-based filtering.

Most internet products we use today are powered by recommender systems. YouTube, Netflix, Amazon, Pinterest, and long list of other internet products all rely on recommender systems to filter millions of contents and make personalized recommendations to their users. Recommender systems are well-studied and proven to provide tremendous values to internet businesses and their consumers.

There are majorly three types of recommender systems which work primarily in the Media and Entertainment industry:

* Collaborative Recommender system
* Content-based recommender system
* Knowledge based recommender system

Recommender System is a vast concept rooted from a base idea of giving out suggestions to the users. There are wide range of algorithms are used to build a recommender system and the type of recommender system used is mostly dictated by the type of data available. In this project, first three of the above recommender systems were built.

### Content based recommender system

This approach utilizes a series of discrete characteristics of an item in order to recommend additional items with similar properties. Content-based filtering methods are based on a description of the item and a profile of the user's preferences. To keep it simple, it will suggest your similar movies based on the movie we give (movie name would be the input) or based on all of the movies watched by a user (user is the input). It extracts features of an item and it can also look at the user's history to make the suggestions.

### Collaborative recommender system

Collaborative filtering is based on the assumption that people who agreed in the past will agree in the future, and that they will like similar kinds of items as they liked in the past. The system generates recommendations using only information about rating profiles for different users or items. By locating peer users/items with a rating history similar to the current user or item, they generate recommendations using this neighborhood. This approach builds a model from a user’s past behaviors (items previously purchased or selected and/or numerical ratings given to those items) as well as similar decisions made by other users. This model is then used to predict items (or ratings for items) that the user may have an interest in. Collaborative filtering methods are classified as memory-based and model-based.

## MovieLens 100K Dataset

The MovieLens dataset is a popular dataset commonly used in research and industry to develop and evaluate recommendation systems. The MovieLens 100k dataset contains movie ratings data collected from the MovieLens website. Specifically, the dataset includes ratings provided by users on a scale of 1 to 5 stars for approximately 1,000 movies. The dataset also includes demographic information for the users, such as age, gender, and occupation.

The data is divided into three files: u.data, u.item, and u.user. The u.data file contains the movie ratings data, with each row representing a single rating provided by a user for a movie. The u.item file contains information about each movie in the dataset, including the movie title, release date, and genre. The u.user file contains demographic information for each user in the dataset.

The dataset is often used to evaluate the performance of recommendation algorithms. Researchers typically split the data into training and testing sets and use the training set to train the recommendation algorithm and the testing set to evaluate its performance.

## Data preprocessing and model creation

The following preprocessing steps are performed on the MovieLens 100k dataset.

#### **Knowledge-based recommender system on the MovieLens 100k dataset**

Unlike collaborative filtering methods, which rely on historical user ratings data to make recommendations, knowledge-based recommender systems make recommendations based on the attributes of the items themselves.

1. The MovieLens 100k dataset is loaded from the u.item file into a Pandas DataFrame. The DataFrame has 24 columns representing the item ID, the title of the movie, the release date, and various genres.
2. The release date column is split into separate columns representing the year, month, and day of the release date.
3. The genres column is transformed into a binary matrix, with each row representing a single movie and each column representing a genre. The elements in the matrix represent whether or not the movie belongs to the corresponding genre.
4. The user is prompted to enter their preferred genres, and the input is converted into a binary vector, with each element representing whether or not the user is interested in the corresponding genre.
5. The binary vector representing the user's preferred genres is multiplied by the binary matrix representing the genres of all the movies in the dataset to obtain a binary vector representing the movies that match the user's preferred genres.
6. The user is then prompted to enter the release year range they are interested in, and the input is used to filter the movies that match the user's preferred genres based on their release date.
7. Finally, the top 10 movies that match the user's preferred genres and release year range are recommended to the user.

These are the top movies that can be naively suggested to the new users for the requested movie genre: Action. Recommendations based on top average ratings.

|  |  |
| --- | --- |
| movie title | rating |
| Star Wars (1977) | 4.358491 |
| Godfather, The (1972) | 4.283293 |
| Raiders of the Lost Ark (1981) | 4.252381 |
| Titanic (1997) | 4.245714 |
| Empire Strikes Back, The (1980) | 4.204360 |
| Boot, Das (1981) | 4.203980 |
| Godfather: Part II, The (1974) | 4.186603 |
| African Queen, The (1951) | 4.184211 |
| Princess Bride, The (1987) | 4.172840 |
| Braveheart (1995) | 4.151515 |

These are the most popular movies which can be recommended to a new user in Action genre. Recommendations based on Popularity

|  |  |
| --- | --- |
| movie title | Number of Users watched |
| Star Wars (1977) |  |
| Return of the Jedi (1983) | 507 |
| Air Force One (1997) | 431 |
| Independence Day (ID4) (1996) | 429 |
| Raiders of the Lost Ark (1981) | 420 |
| Godfather, The (1972) | 413 |
| Rock, The (1996) | 378 |
| Empire Strikes Back, The (1980) | 367 |
| Star Trek: First Contact (1996) | 365 |
| Titanic (1997) | 350 |

#### **Item-based collaborative filtering recommender system using k-nearest neighbors (KNN) algorithm on the MovieLens 100k dataset**

The following are the data preprocessing steps that are performed for this part:

1. The MovieLens 100k dataset is loaded from the u.data file into a Pandas DataFrame. The DataFrame has four columns representing the user ID, the item ID (movie ID), the rating provided by the user for the item, and the timestamp of the rating.
2. The DataFrame is then transformed into a sparse matrix using the csr\_matrix() function from the SciPy library. The sparse matrix has the item IDs as the rows and the user IDs as the columns. The elements in the matrix represent the rating provided by each user for each item.
3. The mean rating of each item is calculated by taking the average of the non-zero elements in each row of the sparse matrix. The mean rating is then subtracted from each non-zero element in the row to obtain the item-centered matrix. This step is performed to normalize the ratings and remove the bias caused by different users having different rating scales.
4. The similarity between items is calculated using the cosine similarity measure on the item-centered matrix. The cosine similarity measure calculates the angle between two vectors, in this case, the vectors representing the ratings of two items by all the users. The resulting similarity matrix has the item IDs as both the rows and columns.
5. Finally, the KNN algorithm is applied on the similarity matrix to find the top k most similar items to a given item. The predicted rating for the target item is then calculated as the weighted average of the ratings provided by the k most similar items, weighted by their similarity scores.

**Movie Recommender System for a User**

Movie Recommendation using KNN with Input as User id, Number of similar users should the model pick and Number of movies you want to get recommended. Results for a User id: 307; number of similar users to be considered: 15; Enter number of movies to be recommended: 15 are:

|  |  |
| --- | --- |
| User | separated by distance of |
| 70 | 0.4560883724650484 |
| 738 | 0.4846662001127756 |
| 922 | 0.503221313979523 |
| 407 | 0.5038250337403114 |
| 514 | 0.5060750098353226 |
| 44 | 0.5160506271876224 |
| 660 | 0.5165826487301209 |
| 5 | 0.5211146313938015 |
| 457 | 0.5309167131718452 |
| 23 | 0.5316197783536492 |
| 843 | 0.5324703658288387 |
| 64 | 0.53318921205275 |
| 198 | 0.535682894616484 |
| 815 | 0.5416036160331636 |
| 95 | 0.5468066886836396 |

Movies recommended based on similar users are:

["Schindler's List (1993)",

'Liar Liar (1997)',

'When Harry Met Sally... (1989)',

'Leaving Las Vegas (1995)',

'Silence of the Lambs, The (1991)',

'Dead Man Walking (1995)',

'Trainspotting (1996)',

'Forrest Gump (1994)',

'Scream (1996)',

'Twelve Monkeys (1995)',

'Jerry Maguire (1996)',

'Raising Arizona (1987)',

'Godfather, The (1972)',

'Rock, The (1996)',

'Fugitive, The (1993)']

**Movie Recommender System Using Movie Name**

Movie Recommendation using KNN with Input as Movie Name and Number of movies you want to get recommended. Top 10 movies which are very much similar to the Movie- 101 Dalmatians (1996) are:

Jack (1996)

Twister (1996)

Willy Wonka and the Chocolate Factory (1971)

Independence Day (ID4) (1996)

Toy Story (1995)

Father of the Bride Part II (1995)

Hunchback of Notre Dame, The (1996)

Lion King, The (1994)

Mrs. Doubtfire (1993)

Jungle Book, The (1994)

#### **Matrix factorization-based collaborative filtering recommender system using singular value decomposition (SVD) on the MovieLens 100k dataset**

1. The MovieLens 100k dataset is loaded from the u.data file into a Pandas DataFrame. The DataFrame has four columns representing the user ID, the item ID (movie ID), the rating provided by the user for the item, and the timestamp of the rating.
2. The DataFrame is then split into a training set and a test set using the train\_test\_split() function from the Scikit-learn library. The training set contains 80% of the data, and the test set contains 20% of the data.
3. The user-item matrix is constructed using the pivot\_table() function from the Pandas library. The matrix has the user IDs as the rows and the item IDs as the columns. The elements in the matrix represent the rating provided by each user for each item.
4. The matrix is then normalized by subtracting the mean rating of each user from all the ratings provided by that user. This step is performed to normalize the ratings and remove the bias caused by different users having different rating scales.
5. The SVD algorithm is applied on the normalized user-item matrix to factorize it into three matrices: a user-feature matrix, a feature-feature matrix, and an item-feature matrix. The user-feature matrix has the users as the rows and the latent factors as the columns. The item-feature matrix has the items as the rows and the latent factors as the columns. The feature-feature matrix contains the weights that connect the user-feature and item-feature matrices.
6. The predicted rating for a target item by a target user is calculated as the dot product of the target user's row in the user-feature matrix and the target item's row in the item-feature matrix.
7. The model's performance is evaluated on the test set using the mean squared error (MSE) and the root mean squared error (RMSE) metrics.

Movie Recommendations using SVD giving a movie name as input:

Enter the Movie name: dal

Entered Movie name is not matching with any movie from the dataset . Please check the below suggestions :

['101 Dalmatians (1996)', 'Mrs. Dalloway (1997)']

Enter the Movie name: 101 Dalmatians (1996)

Enter Number of movie recommendations needed: 10

Top 10 movies which are very much similar to the Movie- 101 Dalmatians (1996) are:

Black Beauty (1994)

Free Willy 2: The Adventure Home (1995)

Evening Star, The (1996)

Robin Hood: Men in Tights (1993)

Cool Runnings (1993)

Turbo: A Power Rangers Movie (1997)

Remains of the Day, The (1993)

City Hall (1996)

Children of the Corn: The Gathering (1996)

**Deep neural network-based collaborative filtering recommender system using the softmax activation function on the MovieLens 100k dataset**

1. The MovieLens 100k dataset is loaded from the u.data file into a Pandas DataFrame. The DataFrame has four columns representing the user ID, the item ID (movie ID), the rating provided by the user for the item, and the timestamp of the rating.
2. The DataFrame is then split into a training set and a test set using the train\_test\_split() function from the Scikit-learn library. The training set contains 90% of the data, and the test set contains 10% of the data.
3. The user-item matrix is constructed using the pivot\_table() function from the Pandas library. The matrix has the user IDs as the rows and the item IDs as the columns. The elements in the matrix represent the rating provided by each user for each item.
4. The matrix is then normalized by subtracting the mean rating of each user from all the ratings provided by that user. This step is performed to normalize the ratings and remove the bias caused by different users having different rating scales.
5. The user-item matrix is then converted into a one-hot encoded matrix, where each row represents a user and each column represents an item. The elements in the matrix represent whether or not the user has rated the corresponding item.
6. The one-hot encoded matrix is split into a training set and a validation set using the train\_test\_split() function from the Scikit-learn library. The training set contains 80% of the data, and the validation set contains 20% of the data.
7. The neural network model is constructed using the Keras library. The model has three layers: an input layer, a hidden layer, and an output layer. The input layer takes in the one-hot encoded matrix as input, and the output layer predicts the probability of each user rating each item. The softmax activation function is used in the output layer to ensure that the predicted probabilities sum up to 1.
8. The model is trained on the training set using the categorical cross-entropy loss function and the Adam optimizer. The model is evaluated on the validation set using the accuracy metric.
9. Finally, the model's performance is evaluated on the test set using the accuracy metric.

**Movie Recommendations**

The user provides their user ID as input, and the available data frame is used to extract the movie IDs that the user has not already seen. The deep neural network (DNN) model takes two inputs: user IDs and corresponding movie IDs, and predicts the user's rating for a given movie. By inputting the user ID and previously unseen movie ID, the DNN model predicts the rating that the user would give the movie. Ratings are normalized to a standardized scale and are not rescaled to a 0-5 rating scale. The DNN model is utilized to predict the ratings of previously unseen movies.

**Predicted Ratings**

[[6.28711879e-01 3.71125787e-01 1.93846718e-05 ... 2.48171236e-05

2.07571484e-05 3.11595759e-05]

[5.16196430e-01 4.83636826e-01 2.06636632e-05 ... 2.40022491e-05

2.14833890e-05 3.09596326e-05]

...

[6.53564811e-01 3.46285373e-01 1.90432311e-05 ... 2.25746426e-05

1.92296520e-05 2.91551714e-05]]

Output is of shape (1628, 9). We got probability of each possible rating from 1 to 5. We can extract specific rating which user would have given to a movie but it is not useful for these recommendations now.

array([0.6287119, 0.5161964, 0.8921049, ..., 0.6535648, 0.577208 ,

0.6869574], dtype=float32)

These predicted pseudo-ratings of the user for the unseen movies are sorted with highest ratings in the first and these labels are inverse transformed to get desired number of Movie names. As an final example, given user id as input:

Enter user id

307

Enter number of movies to be recommended:

15

Movie seen by the User:

['12 Angry Men (1957)',

'2001: A Space Odyssey (1968)',

'Abyss, The (1989)',

'Alien (1979)',

'Apollo 13 (1995)',

'Boot, Das (1981)',

'Brady Bunch Movie, The (1995)',

'Braveheart (1995)',

'Brazil (1985)',

'Casablanca (1942)',

'Fargo (1996)',

...

...

'Star Wars (1977)',

'Top Gun (1986)',

'Toy Story (1995)',

'Wallace & Gromit: The Best of Aardman Animation (1996)',

'Wizard of Oz, The (1939)',

'Wrong Trousers, The (1993)']

Top 15 Movie recommendations for the User 307 are:

['Speed 2: Cruise Control (1997)',

'Houseguest (1994)',

'Batman & Robin (1997)',

'Magic Hour, The (1998)',

"Devil's Advocate, The (1997)",

'Gone with the Wind (1939)',

'Cobb (1994)',

'Cool Runnings (1993)',

'Independence Day (ID4) (1996)',

'Smoke (1995)',

'Once Were Warriors (1994)',

'True Romance (1993)',

'Red Rock West (1992)',

'Third Man, The (1949)',

'MatchMaker, The (1997)']

## References

* Han, J.; Kamber, M. & Pei, J. (2012), 'Data mining concepts and techniques, third edition' , Morgan Kaufmann Publishers , Waltham, Mass. .
* McKinney, W. (2013), *Python for Data Analysis: Data Wrangling with Pandas, NumPy, and IPython* , O'Reilly Media .
* <https://www.kaggle.com/datasets/kanametov/movies-recomendation-system>
* <https://github.com/rposhala/Recommender-System-on-MovieLens-dataset>