GRADUATE PROJECT

CSI 6900 Winter 2020

Movie Recommendation System Using MovieLens Dataset

Supervised By:
Dr. Diana Inkpen
University of Ottawa
School of Electrical Engineering & Computer Science (SEECS)

Submitted By:
Deepshi Mediratta (300086013)

Movie Recommendation System Using MovieLens Dataset

Deepshi Mediratta (300086013)

Faculty of Engineering, University Of Ottawa, Ottawa, Canada
dmedi018@uottawa.ca

Abstract. Users tend to get overwhelmed with the amount of content that is offered to them which can come in the way of making informed choices. The aim of this project is to recommend personalized movies based on the user's past preference to improve user experience. The dataset used to build the recommendation system is the MovieLens dataset. We performed some exploratory data analysis to understand the underlying patterns in the data and prepared the data so that it can be used with the models. We train the recommender models to come up with movie recommendations for users. The models discussed include matrix factorization using Single Value Decomposition, Clustering using K-nearest neighbors and TF-IDF using cosine similarities. The models are used to recommend movies for users and the recommendations are analyzed in a user case study. The models are also evaluated using the evaluation measures and the results are presented.

Keywords: rating, movie, movie recommender, collaborative filtering, content based, SVD

1 Introduction

Over the years, there has been a rapid increase in online content which makes it difficult for a user to make informed choices. A recommendation system filters the data using different filtering and clustering algorithms and recommends the most relevant items to the user. The need to build a recommendation system is extremely important given the huge demand for personalized content by the modern users. The major application for a recommendation system is in the field of social media to advertise content, recommend movies, songs, books, or clothing items.

In this project, the MovieLens dataset is used to build a recommendation system. The idea is to predict movie preferences based on the user's past preference or finding another user profile with similar preferences. As the user keeps on using the system, it will understand the user better and provide curated movie recommendations. We have discussed two popular approaches of recommender systems- collaborative filtering and content-based filtering.

2 The MovieLens Dataset

2.1 Data Description

The dataset used in this project is the MovieLens dataset [1]. GroupLens Research collected the data and made it available in various formats on the MovieLens web site [2]. For the purpose of this project, (ml-latest-small) version of the dataset has been used [3]. It consists of 100836 ratings across 9742 movies by 610 users. It was collected between March 29, 1996 and September 24, 2018. The users were selected at random for inclusion and the selected users had rated a minimum of 20 and a maximum of 2698 movies. Each user is represented by a unique identifier and no other demographic information is included. Only movies with at least one rating are included in the dataset.

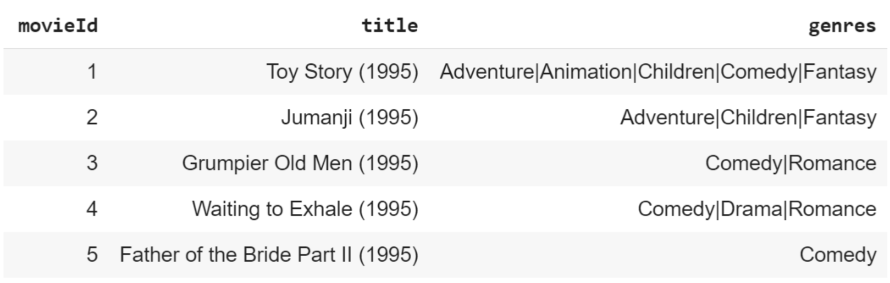
The dataset files used are 'ratings.csv' and 'movies.csv'. The ratings file contains the ratings provided by the uses for the movies. The movies file contains the movie id to movie title and genre mapping. The features have been described in detail in Table 1 and Table 2.

Table 1. The *movies* dataset feature description.

Feature	Feature Description
movieId	Unique Identifier representing the movies.
title	The movie title, also includes the movie release year in parentheses.
genres	Tab separated list of genres selected from list of 19 genres.

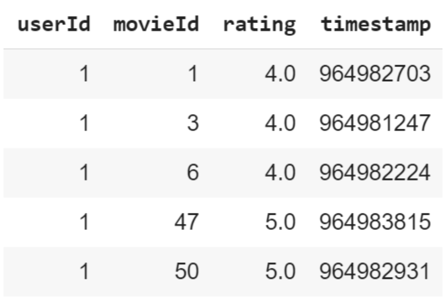
Table 2. The *ratings* dataset feature description.

Feature	Feature Description
userId	Unique identifier representing the user.
movieId	Unique Identifier representing the movies.
rating	Ratings made on a 5-star scale, with half-star increments (0.5 stars – 5.0 stars)
timestamp	Represents seconds since midnight coordinated universal time(UTC) of January 1, 1970



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

6. 1. Sample *movies* data.



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

Fig. 2. Sample *ratings* data.

3 Methodology

3.1 Data Exploration and Analysis

The number of unique users is 610 and the number of unique movies is 9724. However, we have only 100836 movie ratings, this means not all users have rated all movies. We observed that the minimum number of movies that a user rated was 20 and maximum was 2698 movies. On an average, the number of movies rated by a user is 165. The *user* × *movie* matrix had a sparsity of 0.983

$$Sparsity = 1 - \frac{\text{number of ratings}}{\text{number of users} * \text{number of movies}}$$

The ratings distribution was analysed and it was observed that the distribution was normal with most ratings centered at 3-4 stars. The minimum rating was 0.5 stars and the maximum rating was 5 stars. Average rating was 3.5 stars.

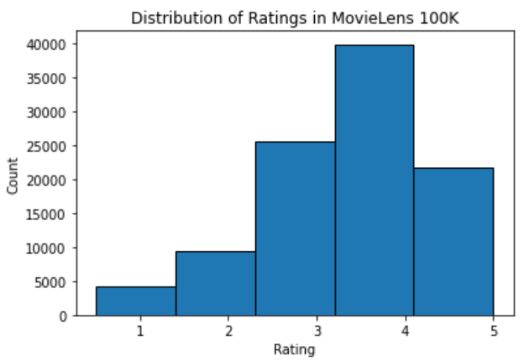


Fig. 3. Word Cloud of the most frequent genres in *movies* dataset.

When we plot the Word frequency distribution chart (Figure 4) for the movies genres, we observed that the most frequent genres were Comedy, Drama, Thriller, Action and Romance.

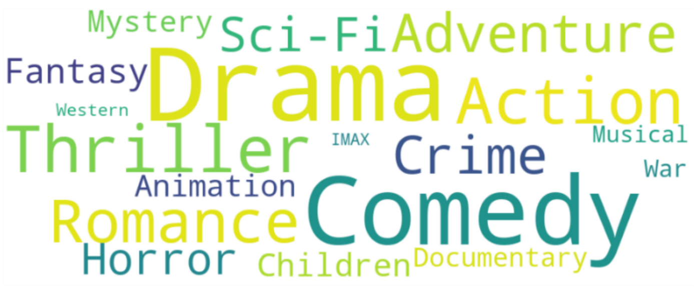


Fig. 4. Word Cloud of the most frequent genres in *movies* dataset.

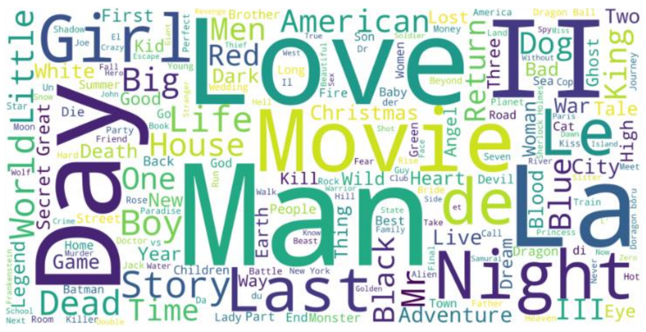


Fig. 5. Word Cloud of the most frequent words in movie titles.

When we plot the Word frequency distribution chart (Figure 5) for the movie titles, we observed that the most frequent words used in the movie title were Love, Man, Movie, Day, Girl. The appearance of the words Love, Man and Girl in the top most frequent words used in movie titles aligns with the fact that Romance is in top 5 movie genres.

3.2 Preparing the data

The *ratings* dataset is transformed into a *movie* × *user* 2D matrix where the y-axis would represent the users and the x-axis would represent the movies. The element at (*i*,*j*) position in the 2D matrix will be the rating that the user *i* gave for the movie *j*. This matrix is a sparse matrix with sparsity 0.983 because not all users have rated most movies. Figure 6 shows the 2D matrix for training data represented as a Sparse Matrix with movieId on x-axis, userId on y-axis and ratings in the values. A similar Matrix is generated for the test dataset also and the missing values in the test 2D matrix is determined using the recommendation models.

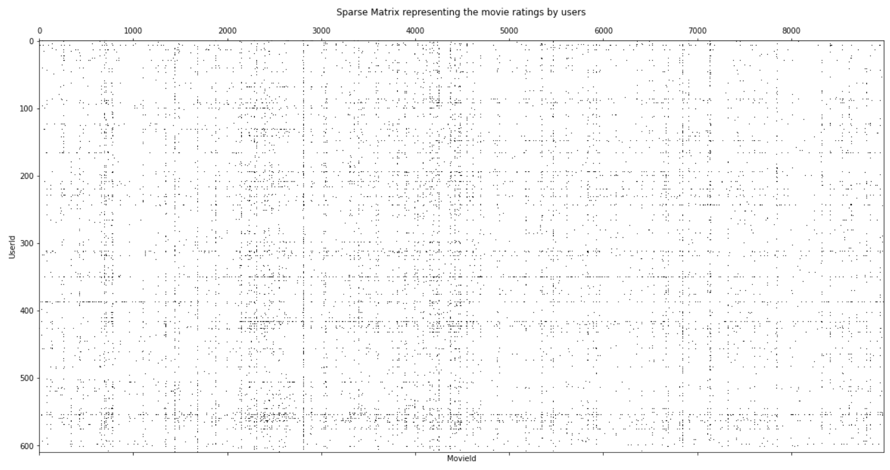


Fig. 6. Training data represented as a Sparse Matrix with movieId on x-axis, userId on y-axis and ratings in the values.

4 Recommendation System Models

4.1 Collaborative Filtering

Collaborative Filtering is a recommendation approach that is based on the assumption that people with similar preference will like the same type of movies. The user preference can be drawn from the movies they have already watched or the ratings given by the user for the movies. The relationship between the users and the movies is used to predict a movie or rating for a movie which could be relevant to the user. Example: If both person A and person B like movie X, and person A likes movie Y, then it is most likely that person B will also like movie Y. We implemented Matrix factorization using single valued decomposition and clustering using K-NN model as a collaborative filtering approach in recommendation system.

4.1.1 Matrix Factorization using Single Valued Decomposition

In Collaborative filtering, matrix factorization is the state of the art solution for sparse data problem. Matrix factorization is a method in linear algebra that reduces a matrix into a product of multiple matrices to make complex matrix computations easier. Single Value Decomposition or SVD is widely used and is a stable matrix factorization method.

We implemented matrix factorization using SVD to predict user ratings for movies and recommend movies based on the predicted ratings. SVD uses a *user* × *movie* 2D matrix where each row represents a user, each column represents a movie and the elements in the matrix are the ratings given by the user for that movie. It decomposes the 2D matrix into three other matrices as given below:

$$R = U.\sum.V^T$$

Where, R = 2D matrix of size ($m \times n$) that we want to decompose

U = The user-feature matrix of size ($m \times k$)

$\sum$ = Diagonal matrix of singular values (weights) of size ($k \times k$)

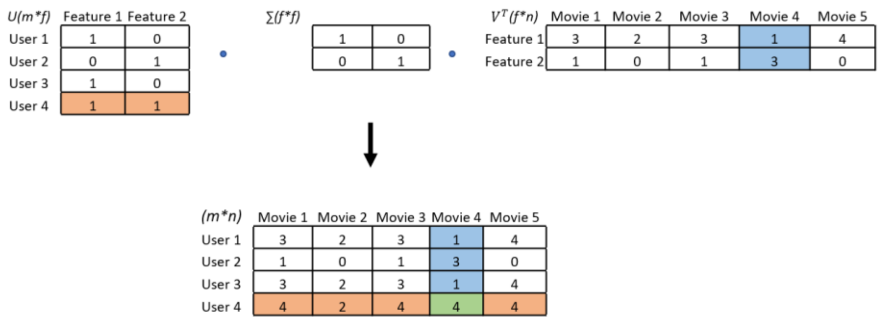
V^T = The movie feature matrix of size ($k \times n$)

m = number of users

n = number of movies

k = number of features

The SVD is calculated using the svd() function that takes the 2D training matrix as input and returns the U, sigma and V^T. The predicted ratings can be then reconstructed by finding the dot product of the three matrices. The number of latent features (f) are optimized by minimizing the root mean square error. For movies, the value for f between 20 and 100 have been found to be the best at generalizing to unseen data [4].



U(m*f)	Feature 1	Feature 2
User 1	1	0
User 2	0	1
User 3	1	0
User 4	1	1

Σ(f*f)	
1	0
0	1

V^T(f*n)	Movie 1	Movie 2	Movie 3	Movie 4	Movie 5
Feature 1	3	2	3	1	4
Feature 2	1	0	1	3	0

(m*n)	Movie 1	Movie 2	Movie 3	Movie 4	Movie 5
User 1	3	2	3	1	4
User 2	1	0	1	3	0
User 3	3	2	3	1	4
User 4	4	2	4	4	4

Fig. 7. Example showing SVD Decomposition with diagonal matrix where weights are 1 for simplicity.

Figure 7 shows an example of SVD for movie rating prediction. The user matrix *U* represents the relationship between users and latent features and matrix *V* represents the similarity between movies and latent features. The diagonal matrix representing the weight of each latent feature is of size 2*2 and the weights are assumed to be 1 for simplicity in the example shown. The dot product of U sigma and V results into the predicted ratings matrix. For instance, we can predict that user 4 will give a rating 4 to movie 4. Based on these ratings, movies can be recommended to the users. Along with predicting ratings for the movies, we can also uncover latent patterns from these matrices, for example, we can observe that user 1 and user 3 have similar feature interests which leads to similar movie rating predictions.

4.1.2 Clustering using K-Nearest Neighbors

K-Nearest Neighbors is a machine learning algorithm that can be used to find similar users by creating a measure of similarity(Cosine similarity in this case) between the users based on the movie ratings. We use the *user* × *movie* 2D matrix to form clusters of similar users, then to make a recommendation, we calculate the distance between the movie that a user has given the highest rating to and all other movies. The distances are then sorted in descending order and the top k nearest movies are recommended to the user. To predict the rating for the movie, the average from the ratings of the top k nearest movies are taken.

4.2 Content Based

The content based recommendation approach focuses on the description of the movies and user's behavior towards those movies. The movies being recommended to the user will be personalized based on the past or current preference, interests, relevance etc. We use the pre tagged genres of movies to recommend similar movies to a user. This type of approach is best suitable when we do not have detailed information about a user's preference but we have information about the movies.

4.2.1 TF-IDF with Cosine Similarity

Term Frequency and Inverse Document Frequency or TF-IDF, is a method that helps in evaluating the importance of a word in a document. The TF-IDF weighting negates the effect of high frequency words in determining the importance of an item. İt converts a collection of raw documents to a matrix of TF-IDF features [5]. For recommending movies, we consider each movie as a document and each genre as a word in the document.

TF: represents frequency of a genre in a movie.

IDF: represents inverse of movie frequency among the whole corpus of movies.

The following formula calculates the TF-IDF score. Log in TF-IDF is used to dampen the effect of high frequency words.

$$tfidf_{i,j} = tf_{i,j} \times \log \frac{N}{df_i}$$

Where, $tf_{i,j}$ = total number of times genre ‘i’ occured in a movie ‘j’

df_i = number of movies containing the genre ‘i’

N = total number of movies

After calculating the TF-IDF score, we want to find out which movies are closer to each other in the vector space model. The vector space model computes the proximity of the movies based on the angle between the movie vectors. We store all movies as their vector representations and determine the cosine similarity between the vectors.

Cosine Similarity:

$$A.B = ||A||\ ||B||\ Cos\Theta$$

$$Cos\Theta = \frac{A.B}{||A||\ ||B||}$$

After finding the cosine similarity, we can determine which movies are similar and recommend the similar movies.

5 Experiments

5.1 Training and Test datasets

The dataset split into training data and test data in 80:20 ratio, 80% for the training and 20% for testing. The training data consisted of 80557 user-movie ratings and the test data consisted of 20279 ratings after dividing the original dataset.

Partitioning the dataset also helped in addressing the cold-start problem in collaborative filtering as the instances of users or movies in the test set must have remaining instances in the training set.

5.2 Evaluation Metrics

The following evaluation metrics are used to evaluate the recommendation systems.

Root Mean Squared Error (RMSE) is the square root of the mean of the squared errors (MSE).

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - y_i')^2}$$

Mean Absolute Error (MAE) is the average of the absolute difference between the predicted values and observed values. It is a linear score which means that all the differences are weighed equally in average.

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - y_i'|$$

Precision and Recall are used to evaluate binary classification models. In order to use these measures to evaluate a numerical problem, we convert the user ratings to relevant or non-relevant labels. Any rating above a threshold is considered as a relevant item and any item below the threshold is considered as non-relevant.

Relevant Item	Actual user rating >= Threshold
Non-Relevant Item	Actual user rating < Threshold
Recommended Item	Predicted user rating >= Threshold
Non-Recommended Item	Predicted user rating < Threshold

After converting the user ratings to binary problem, we calculate the precision and recall for the top k items as we are interested in recommending the top k items to the user.

Precision@k is the precision evaluated for the top *k* predictions. It is defined as the fraction of recommended movies at top *k* that are relevant. For the special case when there are no movies recommended i.e, all movies recommended by the model have the *predicted rating < threshold*, we can not compute *Precision@k* since we can not divide by zero. In such scenarios, we set *Precision@k* as one [1] [2] [3].

$$Precision@k = \frac{\#\ of\ recommended\ movies\ at\ k\ that\ are\ relevant}{\#\ of\ recommended\ movies\ at\ k}$$

Recall@k is the recall evaluated for the top *k* predictions. It is defined as the fraction of relevant movies found in the top *k* recommendations. For the special case where there are no movies that are relevant i.e, all movies rated by the user have *rating < threshold*, we can not compute *Recall@k* since we can not divide by zero. In such scenarios, we set *Recall@k* as one [1] [2] [3].

$$Precision@k = \frac{\#\ of\ recommended\ movies\ at\ k\ that\ are\ relevant}{total\ \#\ of\ relevant\ movies}$$

F1@k is the harmonic mean of Precision@k and Recall@k. It gives equal weight to both Precision@k and Recall@k. For the special case where both Precision@k and Recall@k are zero, then F1@k is also assumed to be zero.

$$F1@k = \frac{2 * Precision@k * Recall@k}{Precision@K + Recall@k}$$

6 Case Study

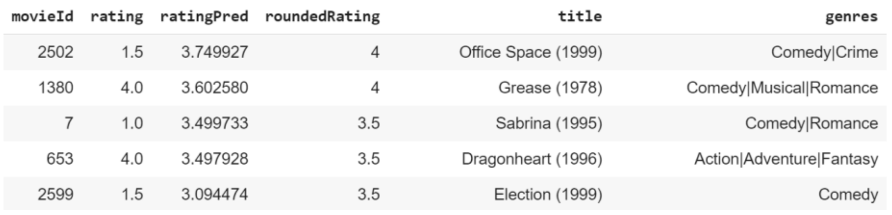
We performed a case study to analyze the movie recommendations made using the different models. These users were selected based on the number of movies they had rated in the original dataset. Table 3 gives an overview of the users. In this case study, we assumed that the threshold used to identify relevant and recommended movies is 3.5 stars. Any movie that has a rating of 3.5 stars or above is considered a relevant movie and any movie that has a predicted rating of 3.5 stars or above is considered as a recommended movie.

Table 3. Evaluation for some users (SVD)

User	userId	Number of movies user rated
User 1	257	20
User 2	143	71
User 3	414	2698

User 1

User 1 had rated only 20 movies which is the least amount of movies rated by a user. Figure 8 shows the actual rating, predicted rating and the rounded rating generated by the SVD model for User 1 for the top recommended movies. We can observe that only 2 movies that were recommended were actually relevant to the user. Both of these movies had very different genres and release years.



movieId	rating	ratingPred	roundedRating	title	genres
2502	1.5	3.749927	4	Office Space (1999)	Comedy|Crime
1380	4.0	3.602580	4	Grease (1978)	Comedy|Musical|Romance
7	1.0	3.499733	3.5	Sabrina (1995)	Comedy|Romance
653	4.0	3.497928	3.5	Dragonheart (1996)	Action|Adventure|Fantasy
2599	1.5	3.094474	3.5	Election (1999)	Comedy

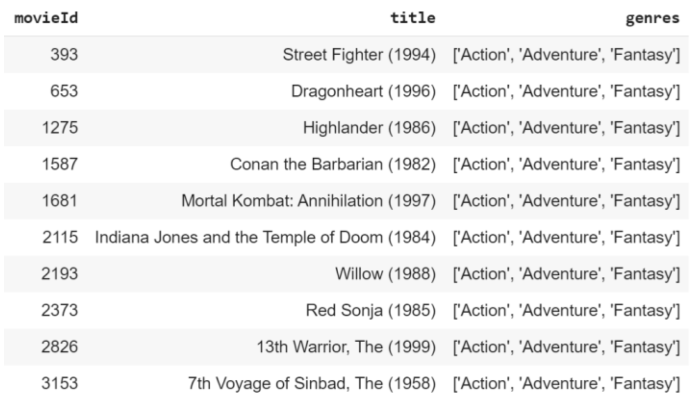
Fig. 8. Movie Recommendations from SVD model for user 1

‘Grease (1978)’ and ‘Dragonheart (1996)’ were the top 2 movies that were both relevant to the user and were recommended by the SVD model. We used knn to find movies similar to ‘Dragonheart (1996)’ and the top movies with their similarity are shown in Table 4.

Table 4. Movies similar to ‘Dragonheart (1996)’ generated using knn

Movie Id	Title	Genres	Similarity Score
143969	Er ist wieder da (2015)|	Comedy	0.18
105585	Machete Kills (Machete 2) (2013)	Action|Crime|Thriller	0.14
940183	Battleship (2012)|	Action|Sci-Fi|Thriller|IMAX	0.12
91842	Contraband (2012)	Action|Crime|Drama|Thriller	0.10
160565	The Purge: Election Year (2016)	Action|Horror|Sci-Fi	0.02
26249	They Call Me Trinity (1971)	Comedy|Western	0.10
79251	Serbian Film, A (Srpski film) (2010)	Horror|Thriller	0.007

We also used TF-IDF to generate genre based movies similar to the movie under observation. For ‘Dragonheart (1996)’ we have listed the top 10 movies predicted by TF-IDF in figure 9. We can see that all movies predicted belong to the same genres – Action, Adventure and Fantasy. We also observe that all movies have close release dates.

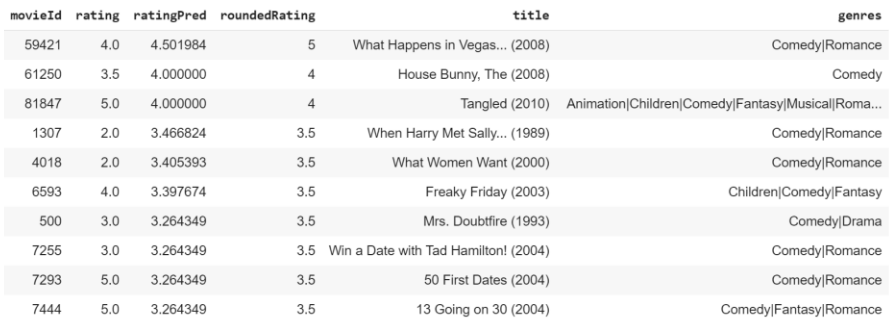


movieId	title	genres
393	Street Fighter (1994)	['Action', 'Adventure', 'Fantasy']
653	Dragonheart (1996)	['Action', 'Adventure', 'Fantasy']
1275	Highlander (1986)	['Action', 'Adventure', 'Fantasy']
1587	Conan the Barbarian (1982)	['Action', 'Adventure', 'Fantasy']
1681	Mortal Kombat: Annihilation (1997)	['Action', 'Adventure', 'Fantasy']
2115	Indiana Jones and the Temple of Doom (1984)	['Action', 'Adventure', 'Fantasy']
2193	Willow (1988)	['Action', 'Adventure', 'Fantasy']
2373	Red Sonja (1985)	['Action', 'Adventure', 'Fantasy']
2826	13th Warrior, The (1999)	['Action', 'Adventure', 'Fantasy']
3153	7th Voyage of Sinbad, The (1958)	['Action', 'Adventure', 'Fantasy']

Fig. 9. Movies similar to 'Dragonheart (1996)' generated using TF-IDF

User 2

User 2 had rated 71 movies which is the average amount of movies rated by a user. Figure 10 shows the actual rating, predicted rating and the rounded rating generated by the SVD model for User 2 for the top recommended movies. We can observe that out of the 10 movies that were recommended, 8 movies were relevant to the user. This is a much better prediction when compared to user 1. It is also analysed that most of the movies belong to the genres comedy and romance.



movieId	rating	ratingPred	roundedRating	title	genres
59421	4.0	4.501984	5	What Happens in Vegas... (2008)	Comedy|Romance
61250	3.5	4.000000	4	House Bunny, The (2008)	Comedy
81847	5.0	4.000000	4	Tangled (2010)	Animation|Children|Comedy|Fantasy|Musical|Roma...
1307	2.0	3.466824	3.5	When Harry Met Sally... (1989)	Comedy|Romance
4018	2.0	3.405393	3.5	What Women Want (2000)	Comedy|Romance
6593	4.0	3.397674	3.5	Freaky Friday (2003)	Children|Comedy|Fantasy
500	3.0	3.264349	3.5	Mrs. Doubtfire (1993)	Comedy|Drama
7255	3.0	3.264349	3.5	Win a Date with Tad Hamilton! (2004)	Comedy|Romance
7293	5.0	3.264349	3.5	50 First Dates (2004)	Comedy|Romance
7444	5.0	3.264349	3.5	13 Going on 30 (2004)	Comedy|Fantasy|Romance

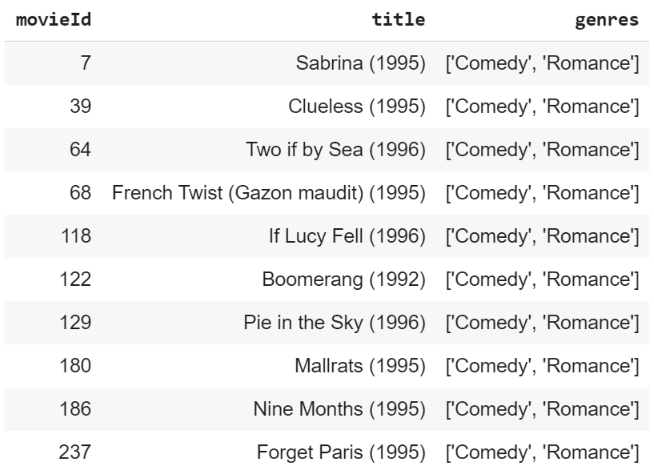
Fig. 10. Movie Recommendations from SVD model for user 2

'What Happens in Vegas... (2008)' is the movie that is highly recommended by SVD model. We used knn to find movies similar to 'What Happens in Vegas... (2008)' and the top movies with their similarity are shown in Table 5.

Table 5. Movies similar to 'What Happens in Vegas... (2008)' generated using knn

Movie Id	Title	Genres	Similarity Score
5853	Scanners (1981)	Horror|Sci-Fi|Thriller	0.0
50800	Messengers, The (2007)	Drama|Horror|Thriller	0.007
6405	Treasure Island (1950)	Adventure|Children	0.01
8906	Cannibal Holocaust (1980)	Horror	0.09
7024	Salo, or The 120 Days of Sodom (Salò o le 120 giornate di Sodoma) (1976)	Drama	0.23
66427	My Name Is Bruce (2007)	Comedy|Horror	0.24

Looking at the movies predicted by TF-IDF that are like 'What Happens in Vegas... (2008)' in Figure 11, we can see that all movies predicted belong to the same genres – Comedy and Romance.

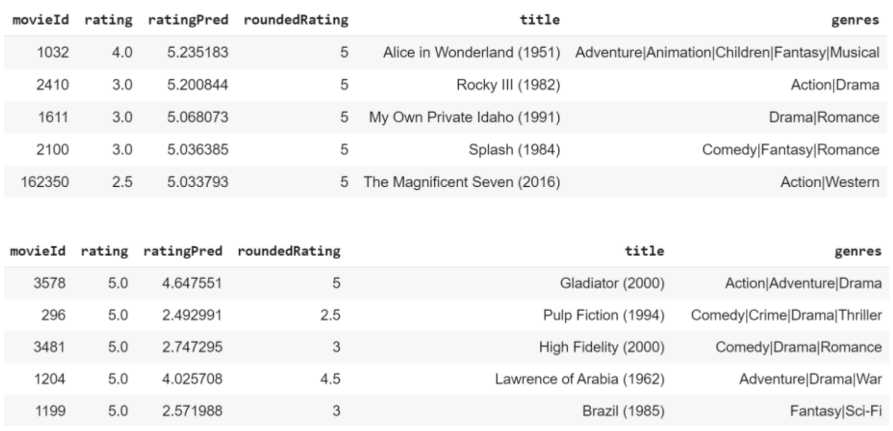


movieId	title	genres
7	Sabrina (1995)	['Comedy', 'Romance']
39	Clueless (1995)	['Comedy', 'Romance']
64	Two if by Sea (1996)	['Comedy', 'Romance']
68	French Twist (Gazon maudit) (1995)	['Comedy', 'Romance']
118	If Lucy Fell (1996)	['Comedy', 'Romance']
122	Boomerang (1992)	['Comedy', 'Romance']
129	Pie in the Sky (1996)	['Comedy', 'Romance']
180	Mallrats (1995)	['Comedy', 'Romance']
186	Nine Months (1995)	['Comedy', 'Romance']
237	Forget Paris (1995)	['Comedy', 'Romance']

Fig. 11. Movies similar to 'What Happens in Vegas... (2008)' generated using TF-IDF

User 3

User 3 had rated 2698 movies which is the highest number of movies rated by a user. Figure 12 shows the actual rating, predicted rating and the rounded rating generated by the SVD model for User 3 for the top recommended movies. We observed that there were a lot of movies that were given the rating 5.



movieId	rating	ratingPred	roundedRating	title	genres
1032	4.0	5.235183	5	Alice in Wonderland (1951)	Adventure|Animation|Children|Fantasy|Musical
2410	3.0	5.200844	5	Rocky III (1982)	Action|Drama
1611	3.0	5.068073	5	My Own Private Idaho (1991)	Drama|Romance
2100	3.0	5.036385	5	Splash (1984)	Comedy|Fantasy|Romance
162350	2.5	5.033793	5	The Magnificent Seven (2016)	Action|Western

movieId	rating	ratingPred	roundedRating	title	genres
3578	5.0	4.647551	5	Gladiator (2000)	Action|Adventure|Drama
296	5.0	2.492991	2.5	Pulp Fiction (1994)	Comedy|Crime|Drama|Thriller
3481	5.0	2.747295	3	High Fidelity (2000)	Comedy|Drama|Romance
1204	5.0	4.025708	4.5	Lawrence of Arabia (1962)	Adventure|Drama|War
1199	5.0	2.571988	3	Brazil (1985)	Fantasy|Sci-Fi

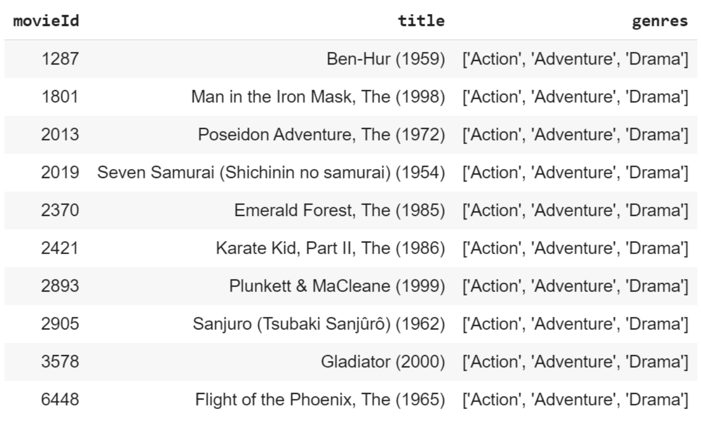
Fig. 12. Movie Recommendations from SVD model for user 3

'Gladiator (2000)' is the movie that is highly recommended by SVD model and the user also gave a 5 star rating to this movie. We used knn to find movies similar to 'Gladiator (2000)' and the top movies with their similarity are shown in Table 6. We observed that the movies close to 'Gladiator (2000)' belonged to the similar genres (Action, Adventure and Drama)

Table 6. Movies similar to 'Gladiator (2000)' generated using knn

Movie Id	Title	Genres	Similarity Score
2232	Cube (1997)	Horror|Mystery|Sci-Fi|Thriller	0
8810	AVP: Alien vs. Predator (2004)	Action|Horror|Sci-Fi|Thriller	0.52
7022	Battle Royale (Batoru rowaiaru) (2000)	Action|Drama|Horror|Thriller	0.53
80219	Machete (2010)	Action|Adventure|Comedy|Crime|Thriller	0.54
3275	Boondock Saints	Action|Crime|Drama|Thriller	0.54

We also used TF-IDF to generate genre based movies similar to 'Gladiator (2000)'. These movies are listed in figure 13. We can see that all movies predicted belong to the same genres – Action, Adventure and Drama.



movieId	title	genres
1287	Ben-Hur (1959)	['Action', 'Adventure', 'Drama']
1801	Man in the Iron Mask, The (1998)	['Action', 'Adventure', 'Drama']
2013	Poseidon Adventure, The (1972)	['Action', 'Adventure', 'Drama']
2019	Seven Samurai (Shichinin no samurai) (1954)	['Action', 'Adventure', 'Drama']
2370	Emerald Forest, The (1985)	['Action', 'Adventure', 'Drama']
2421	Karate Kid, Part II, The (1986)	['Action', 'Adventure', 'Drama']
2893	Plunkett & MaCleane (1999)	['Action', 'Adventure', 'Drama']
2905	Sanjuro (Tsubaki Sanjûrô) (1962)	['Action', 'Adventure', 'Drama']
3578	Gladiator (2000)	['Action', 'Adventure', 'Drama']
6448	Flight of the Phoenix, The (1965)	['Action', 'Adventure', 'Drama']

Fig. 13. Movies similar to 'Gladiator (2000)' generated using TF-IDF

7 Results

Some part of the training data was used for validation to find the optimal value for the number of latent features *f* by minimizing the root mean square error. Table 7 shows the RMSE and MAE for the various number of latent features, the value 23 was selected to be optimal. Using *f=23* the final SVD model was generated and evaluated using the test data. The overall results from the evaluation- Mean Precision@k = 10, Mean Recall@k = 10, Mean F1@K = 10, root mean square error and mean absolute error are shown in Table 8. The evaluation measures for the three users discussed in the case study are shown in Table 9.

Table 7. RMSE & MAE change when number of features are changes on training set. (SVD)

Number of latent Features(*f*)	RMSE	MAE
5	0.938607	0.745241
15	0.936306	0.744082
20	0.936148	0.744781
23	0.936380	0.744428
25	0.936555	0.744103
30	0.939364	0.745582
35	0.938507	0.744897
40	0.940543	0.745282
45	0.941018	0.745002
50	0.942433	0.745888

Table 8. Evaluation for all users (SVD)

Mean Precision@10	0.6576
Mean Recall@10	0.3341
Mean F1@10	0.4431
RMSE	1.556
MAE	0.994

Table 9. Evaluation for some users (SVD)

User	userId	Number of movies user rated	Precision@10	Recall@10	F1@10
1	257	20	0.5	0.25	0.33
2	143	71	1.0	0.27	0.42
3	414	2698	0.3	0.009	0.018

From the overall mean precision@10 and mean recall@10 (Table 8) we can say that 65% of the recommendations made by the system were relevant to the user and 33% of the relevant items appear in the top 10 results. Looking at the precision@10 and recall@10 (Table 9) for individual users from the case study, we can say that the model made the most relevant recommendations for user 2 who rated an average number of movies. For user 1 who rated the least movies, the model was able to recommend relevant movies only 50% of the time. It is interesting to see that the user who rated the highest number of movies recieved the worst recommendations. This is probably because the spectrum of ratings and geners related to this user we very wide and versatile so the recommender is suggesting random movies.

From the rating distribution in the test dataset (figure 14), we can see that both actual and predicted ratings have normal distribution with most ratings centered between 3-4. However, the number of movies with 5 rating is very low in the predicted ratings as compared to the actual ratings.

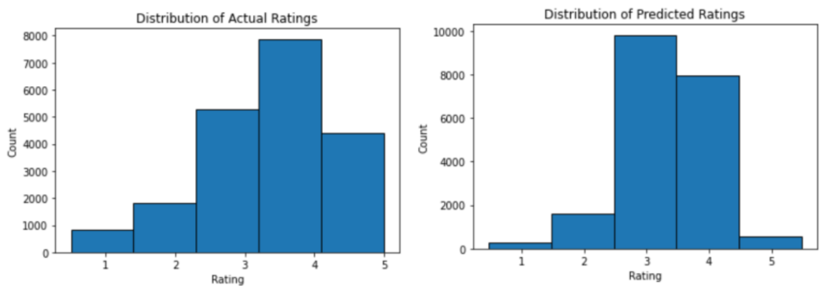


Fig. 14. Distribution of Actual (Left) and Predicted Ratings (Right) (SVD)

8 Discussion

Looking at the model evaluation, the recommendation system seems impressive but if we look at movies recommended to different users in the case study, we can observe that it is not able to predict ratings for the users who have not watched or rated many movies. Similarly if a movie has not been rated by many users, it is difficult to extract latent features for such movies which leads to poor recommendation of those movies. The model is highly influenced by the amount of movies a user rates and the amount of times a movie has been rated. We also observe that the distribution of the rating remains consistent in the actual and predicted ratings. The approach however does not seem to be scalable because as the number of users and movies increase, the space required to store the 2D matrix also increases. A lot of memory is wasted because most of the user-movie ratings are empty. Despite the matrix sparsity issue, SVD is able to predict the user ratings. Knn and TF-IDF both were used to recommend similar movies. However, TF-IDF was able to better capture the movie genre as compared to Knn but the Knn results were closer to the user's behavior. If a user is looking to explore new movies based on the genre then TF-IDF can be used, otherwise if the user wants similar rated movies then Knn could be used.

9 Conclusion

We implemented a recommender system based on three approaches – Single valued decomposition, Knn based clustering and TF-IDF with cosine similarity on the MovieLens dataset to generate more personalized movie recommendation for the users. We started off with analyzing the dataset and preparing the data in a usable 2D matrix format. We trained the matrix on the three models and recommended movies on the test set. We conducted a case study on users who were selected based on the number of movies they have rated and analyzed their movie recommendations.

10 Future Work

The dataset provided lacks ratings for a lot of movies which resulted in a very sparse 2D matrix from which deriving patterns could be difficult. It is understood that not all users want to provide feedback on the movies. We could improve the results by incorporating implicit behavior such as the number of times a user has watched a movie, whether the user finished watching a movie, if the user binge-watched a movie series or television show.

The dataset also lacks information about the movies, it only provides the genre tags that the movie falls under which might not be sufficient to describe the contents of the movie. Some users are inclined towards watching movies that cast their favorite actor, actress or producer. Some users prefer to watch movies with certain topics that are addressed by the movie. Including such information can help improve the model’s recommendation and user experience.

References

[1] F. M. a. K. J. A. Harper, "The MovieLens Datasets: History and Context," *ACM Trans. Interact. Intell. Syst.,* vol. 5, no. 2160-6455, 2015.

[2] [Online]. Available: http://movielens.org.

[3] [Online]. Available: http://files.grouplens.org/datasets/movielens/ml-latest-small-README.html.

[4] N. Becker, "Matrix Factorization for Movie Recommendations in Python," 10 November 2016. [Online]. Available: https://beckernick.github.io/matrix-factorization-recommender/.

[5] J. Le. [Online]. Available: https://nbviewer.jupyter.org/github/khanhnamle1994/movielens/blob/master/Content_Based_and_Collaborative_Filtering_Models.ipynb.

[6] J. L. a. K. J. A. a. T. L. G. a. R. J. T. Herlocker, "Evaluating Collaborative Filtering Recommender Systems," *ACM Trans. Inf. Syst.,* vol. 22, p. 5–53, January 2004.

[7] M. Malaeb, "Recall and Precision at k for Recommender Systems," 13 August 2017. [Online]. Available: https://medium.com/@m_n_malaeb/recall-and-precision-at-k-for-recommender-systems-618483226c54.

[8] W. contributors, "Precision and recall," [Online]. Available: https://en.wikipedia.org/w/index.php?title=Precision_and_recall&oldid=945184172.