Collaborative Filtering and Content-based Movie Recommendation System

Andreas Adolfsson
Dpt. of Computer Engineering
Santa Clara University,
Santa Clara, Ca, 95053
Aadolfsson@scu.edu

Sai Grandhi
Dpt. of Computer Engineering
Santa Clara University,
Santa Clara, Ca, 95053
Sgrandhi@scu.edu

Yifei Pei
Dpt. of Computer Engineering
Santa Clara University,
Santa Clara, Ca, 95053
Ypei@scu.edu

Keng-Lun Yeh
Dpt. of Computer Engineering
Santa Clara University,
Santa Clara, Ca, 95053
Kyeh@scu.edu

Abstract

The Internet provides access to seemingly endless data but it does not teach us how to use this massive resource. In our research, we want to utilize MovieLens dataset of movies and user ratings to compare different recommendation engine models. We will investigate both content-based and collaborative filtering models for selecting movies. We will evaluate and compare accuracies across algorithms. We also used all recommender systems to recommend movies for one user in our project.

***Key Words**: Collaborative Filtering, Recommender System, Matrix Factorization, Big Data, Unsupervised Learning, Content-based.*

1 INTRODUCTION

The Internet allows us to access just about any information within a few clicks. The vast size of this resource, however, frequently obscures information such that users struggle to obtain what they need. This problem also hinders businesses from providing the best products or services for their users. To combat this - Amazon, Netflix, YouTube, and many other companies depend heavily on complex recommendation systems to provide a state-of-the-art solution to this continuously growing problem.

In our project, we will use movie data to build a movie recommender system. Our system will use the user-specific ratings for each user provided GroupLens, a research lab in the Department of Computer Science and Engineering at University of Minnesota. We will leverage

the users existing ratings to make rating predictions for new movies, allowing for the provision of movie suggestions that may interest users.

As machine learning has become a hot topic for prediction problems, we want to utilize these methods to recommend movies for users. During our background exploration, we discovered three types of collaborative filtering are widely used in recommender systems: item-item collaborative filtering, user-user collaborative filtering and matrix factorization. For item-item collaborative filtering, recommender systems are based on the similarity between items calculated using people's ratings of those items. For user-user collaborative filtering, recommender systems predict the ratings by finding similar users. Lastly, matrix factorization algorithms work by decomposing the user-item interaction matrix into the product of two lower dimensionality rectangular matrices. Besides collaborative filtering methods, we also used content based recommendation with movie genre and movie release year. There are two different MovieLens datasets provided by GroupLens. One has 20 million ratings from more than 100,000 users while the other one has 100,000 ratings from 610 users. In order to work with the 20M ratings dataset, faster computation platforms and big data manipulations skills are required. As we lack these, we decided to implement the models on the 100K dataset.

2 DATA

We found a dataset on GroupLens [1]. The dataset describes ratings and free-text tagging activities from MovieLens, a movie recommendation service. It contains 100836 ratings and 3683 tag applications across 9742 movies. These data were created by 610 users between March 29, 1996 and September 24, 2018. The dataset was generated on September 26, 2018. Users were selected at random for inclusion. All selected users had rated at least 20 movies. The dataset contains 4 files: tags.csv, movies.csv, ratings.csv, and links.csv. For our research, we will be considering only ratings.csv and movies.csv data files. The user ratings for each user are stored in ratings.csv. The metadata for each movie is present in movies.csv file. There was no missing data in this dataset.

3 PREPROCESSING

1. We dropped "timestamp" column from ratings.csv because we didn't need the day each user rated a movie for recommendations.
2. User IDs go from 1 to 100836. Because these IDs will index a Numpy array, so we want them to start from 0. We also want to make sure we utilize all space in the array so we don't want the max User ID to be 100K, but only have 500 users. However, after checking the dataset, we found there is no missing user ID between minimum and maximum of user IDs. So we need to subtract 1 from each IDs.
3. Movies.csv file contains the data in the following format:

	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 1: Initial format of movies.csv

4. Each movie has the year embedded with the title and the genres column is a pipe-separated list.
5. The year is brought out and all the genres are added as columns to the dataframe. If the movie contains the genre, the boolean value for that column is updated to be true.
6. Now genres can be used as a sparse boolean matrix for analysis and recommendations.

	movieId	title	year	(no genres listed)	Action	Adventure	Animation	Children	Comedy	Crime	...	Film-Noir	Horror	IMAX	Musical	Mystery	Romance	Sci-Fi	T
0	1	Toy Story (1995)	1995.0	False	False	True	True	True	True	False	...	False	False	False	False	False	False	False	
1	2	Jumanji (1995)	1995.0	False	False	True	False	True	False	False	...	False	False	False	False	False	False	False	
2	3	Grumpier Old Men (1995)	1995.0	False	False	False	False	False	True	False	...	False	False	False	False	False	True	False	
3	4	Waiting to Exhale (1995)	1995.0	False	False	False	False	False	True	False	...	False	False	False	False	False	True	False	
4	5	Father of the Bride Part II (1995)	1995.0	False	False	False	False	False	True	False	...	False	False	False	False	False	False	False	

Figure 2: movies dataframe modified for analysis

7. We shuffle the dataset and set the cutoff 0.8. And then 80% of the dataset go to training dataset and 20% of the dataset go to testing dataset.

4 Exploratory Data Analysis

In order to understand the data that we used matplotlib and pandas to analyze the movielens dataset. Figure 3 contains the frequency of each genre in the genres column. The genre that was most frequent was Drama genre, followed by Comedy, Thriller, and Action.

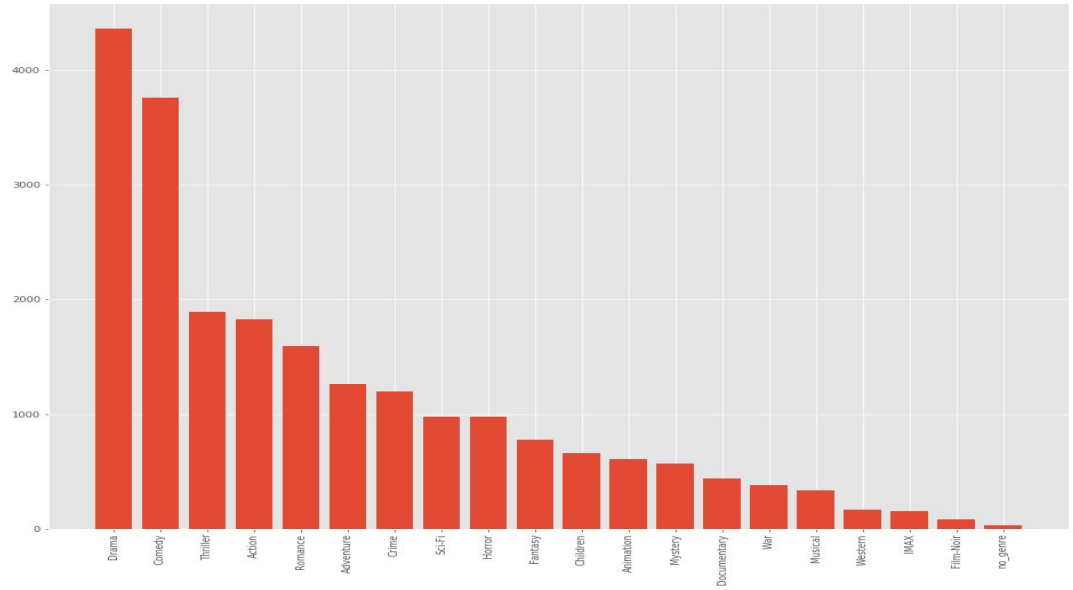

Figure 3: Frequency of each genre

To understand how users rated movies based on the year, the scatter plot in Figure 4 was plotted. From the figure, it can be seen that movies released recently, especially from 1980s and after, have received more ratings than those released previously. Moreover, movies released before 1960s usually received a rating of 3 or higher. This means that users tend to watch the old movies that they heard were good instead of selecting any random movie. It can also be seen with the high-number of low-rated movies from 1990s till now; indicates that users tend to watch new movies without preconception and hence are able to give an honest rating. Users signed up for MovieLens service seem to consist of a younger audience as they tend to watch most recently released movies a lot more than the old ones.

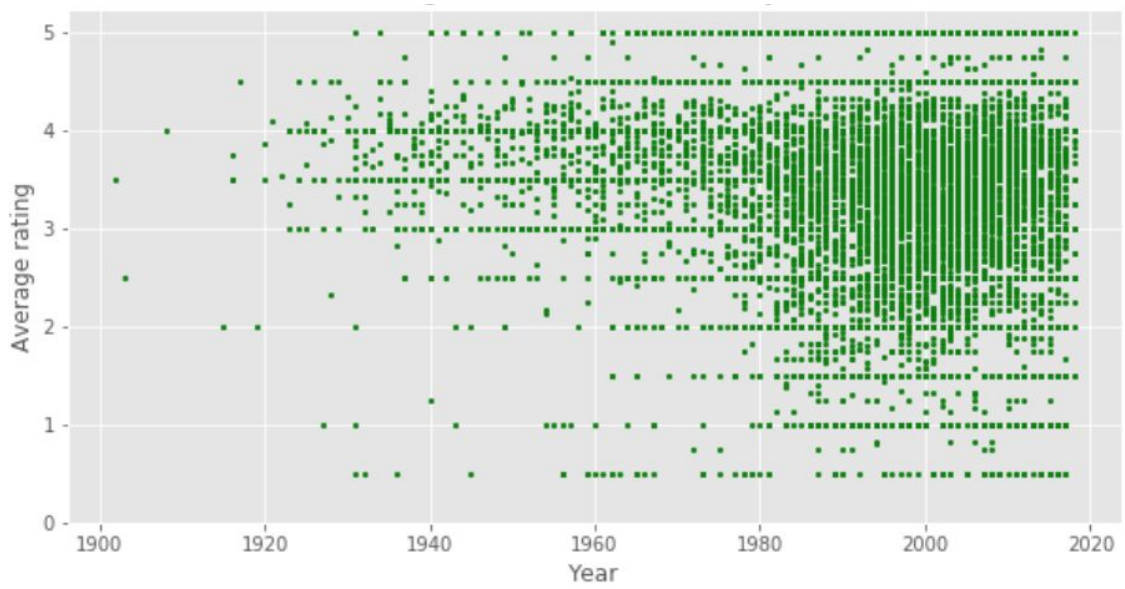

Figure 4: Average movie ratings according to the year

5 METHODS

As mentioned previously, we had four methods for movie recommending. Content-based recommendation, User-User Collaborative Filtering, Item-Item Collaborative Filtering, and Matrix Factorization.

5.1 Content-based recommendation

For content-based recommendation, we used movie genres and movie release year as features of similarity. To predict how a given user would rate a movie, we select top k movies that are most similar to the target movie, and use their similarity score as weights to predict how the user will rate the movie, based on how he/she rated those k similar movies. Genre similarity is computed by using cosine function (Figure 5) and release year similarity computed by an exponential decay function (Equation 1).

$$y = e^{-diff/10}$$

Equation 1: Exponential decay function applied to recommend content based on years

$$a = [a_1, a_2, \dots, a_m]$$
$$b = [b_1, b_2, \dots, b_m]$$

$$cosine(a, b) = \frac{\sum_{i=1}^{m} a_i b_i}{\sqrt{\sum_{i=1}^{m} a_i}\sqrt{\sum_{i=1}^{m} b_i}}$$

Figure 5: Cosine similarity between genres for movie 'a' and movie 'b'

5.2 User-user Collaborative Filtering

With user-user collaborative filtering, we aimed to predict any given users, how he/she would rate movies.

We first used cosine similarity to find out top 50 similar user to the user we want to predict.

Then we predicted the rating of the target user by summing all 50 similar user's rating times similarity then divided by total similarity.

In the equation, rate (i, j) stands for the prediction of user i's rating on movie j. k stands for the top 50 similar users. Set J are the union of the 50 users, and the users that has rated movie j. sim(i,k) stands for the cosine similarity value between user i and user k. The rate (k,j) on the right hand side stands for the actual rating user k gives to movie j.

$$rate(i, j) = \frac{\sum_{\overline{k \in J}} sim(i, k) \cdot rate(k, j)}{\sum_{\overline{k \in J}} sim(i, k)}$$

5.3 Item-Item Collaborative Filtering

With item-item collaborative filtering, we aimed to predict any given item, how it would be rated by all user who has not rate it. To save space, it was suffice to say that it was the same as our user-user collaborative filtering, just the subject and the object are swapped. Two reasons why in theory item-item CF could be better than user-user CF. First, items were more stable than users, the taste of user may change in time. Second, there were less items than users, took less time to go through all items finding similar items.

5.4 Matrix Factorization

We decomposed the rating matrix into 2 matrices that the dot product of these 2 matrices approximately equalled to the rating matrix. In the decomposition process, we also considered biases and regularization terms. The training algorithm was called "alternating least squares".

Alternating Least Square Algorithm:
W = randn(N, K) ; U = randn(M, K); B = [0] * N; C = [0] *M
For t in range (T):

$$w_i = \left(\sum_{j \in \Psi_i} u_j u_j^T + \lambda I \right)^{-1} \sum_{j \in \Psi_i} (r_{ij} - b_i - c_j - \mu) u_j$$

$$u_j = \left(\sum_{i \in \Omega_j} w_i w_i^T + \lambda I \right)^{-1} \sum_{i \in \Omega_j} (r_{ij} - b_i - c_j - \mu) w_i$$

$$b_i = \frac{1}{|\Psi_i| + \lambda} \sum_{j \in \Psi_i} (r_{ij} - w_i^T u_j - c_j - \mu)$$

$$c_j = \frac{1}{|\Omega_j| + \lambda} \sum_{i \in \Omega_j} (r_{ij} - w_i^T u_j - b_i - \mu)$$

6 EVALUATION

1. Loss would be calculated based on Root Mean Square Error (RMSE) and Mean Absolute Error (MAE) between real ratings and predicted ratings among the datasets.
2. We compared MSE and MAE among four algorithms.
3. We recommended top 10 movies for User ID 75 based on the rating scores for the movies that the user hadn't rated yet using four different algorithms.

7 Final Results

Based on Table 1, we saw all the proposed recommendation algorithms were better than the random recommendation. The RMSE and MAE scores of user-user collaborative filtering were approximately equal to item-item collaborative filtering, which indicated these two algorithms were similar in term of accuracy. Content based recommendation algorithm was more accurate than user-user or item-item collaborative filtering algorithms. The reason for this might be content based recommendation algorithm took years of movies into calculations, which might affect the results. Matrix factorization algorithm was the best recommendation algorithm in our project because the accuracy was highest. The reason might be matrix factorization was the most state-of-the-art algorithm in the recent years. Thus, it should perform better than other algorithms in our project.

Table 1: Algorithm Evaluations

Algorithm Names	RMSE	MAE
Random	1.4385	1.1478
Content Based	0.9380	0.7263
User-User	0.9962	0.7719

Item-Item	0.9942	0.7749
Matrix Factorization	0.9039	0.6984

Table 2: Top 10 Movies of User 75 by Random Recommendation

Movie Names	Years	Ratings
Lord of the Rings: The Fellowship of the Ring	2001	5.0
Jungle Book, The	1967	5.0
Star Wars: Episode IV - A New Hope	1977	5.0
For a Few Dollars More	1965	5.0
Alien	1979	5.0
Rebecca	1940	5.0
Raiders of the Lost Ark	1981	5.0
Sting, The	1973	5.0
North by Northwest	1959	5.0
Terminator 2: Judgement Day	1991	5.0

Table 3: Top 10 Movies for User 75 by Content-Based Recommendation

Movie Names	Years	Ratings
Winnie the Pooh and the Blustery Day	1968	4.94
Sleeping Beauty	1959	4.93
Charlotte's Web	1973	4.93

Aristocats	1970	4.87
Tommy	1975	4.85
Cats Don't Dance	1997	4.67
Prince of Egypt	1998	4.5
Batman: Mask of the Phantasm	1993	4.4
Road to El Dorado	2000	4.3
Brave Little Toaster	1987	4.2

Table 4: Top 10 Movies for User 75 by Item-Item Recommendation

Movie Names	Years	Ratings
Lenny	1974	4.1
Salaam Bombay!	1998	4.02
Passage to India, A	1984	4.0
Oliver Twist	1948	3.9
Venus	2006	3.8
Shop Around the Corner	1940	3.8

Song of the Little Road (Pather Panchali)	1955	3.7
4 Little Girls	1997	3.7
Bank Dick	1940	3.5
Black Widow	1987	3.41

Table 5: Top 10 Movies for User 75 by User-User Recommendation

Movie Names	Years	Ratings
Dances with Wolves	1990	3.2
Apollo 13	1995	2.8
Fargo	1996	2.8
Pulp Fiction	1994	2.6
Dumb & Dumber	1994	2.4
Stargate	1994	2.2
Shine	1996	2.1
Willy Wonka & the Chocolate Factory	1971	2.0
Star Wars: Episode VI -	1983	2.0

Return of the Jedi		
Schindler's List	1993	1.9

Table 6: Top 10 Movies for User 75 by Matrix Factorization

Movies	Years	Ratings
Young Frankenstein	1974	3.95
Shaun of the Dead	2004	3.93
High Noon	1952	3.88
Glory	1989	3.88
Boondock Saints	2000	3.86
Philadelphia Story	1940	3.85
Dr. Strangelove or: How I Learned to Stop Worrying	1964	3.83
Seven Samurai (Shichinin no samurai)	1954	3.83
Thank You for Smoking	2006	3.81
Godfather	1972	3.81

8 Lessons Learned

The hardest part for the project was how to process the large dataset and analyze them. Another difficulty was to understand matrix factorization algorithm for recommendation system because this algorithm was quite abstract. It was also hard to implement all the algorithms to analyze such big datasets. However, Surprise had all the functions for these recommender system algorithms that we could use directly.

9 Conclusions

We used four different approach to build recommender systems for movies and compared the result of these algorithms with random recommendation. We found that content-based recommender system performed better than classic collaborative filtering (user-user collaborative and item-item collaborative). However, matrix factorization, a relative new collaborative filtering algorithm, performed the best. Our future work is to ensemble all those algorithms to build a better recommendation system.

Responsibilities:

Yifei: Background Research, Matrix SVD
Sai: Data preprocessing, Exploratory data analysis, Content recommendation
Keng-Lun Yeh: User-User/Item-Item Collaborative Filtering, Results
Andreas: Code Management, Debugging and Presentation

References

[1] F. Maxwell Harper and Joseph A. Konstan. 2015. The MovieLens Datasets: History and Context. ACM Transactions on Interactive Intelligent Systems (TiiS) 5, 4: 19:1â€“19:19. https://doi.org/10.1145/2827872

[2] "Matrix Factorization: A Simple Tutorial and Implementation in Python." *Quuxlabs*, 16 Sept. 2010, www.quuxlabs.com/blog/2010/09/matrix-factorization-a-simple-tutorial-and-implementation-in-python/.

[3] Tomar, Ankur. "USER-USER Collaborative Filtering Recommender System in Python." *Medium.com*, Medium, 25 Aug. 2017, www.medium.com/@tomar.ankur287/user-user-collaborative-filtering-recommender-system-51f568489727.

[4] Sarwar, Badrul et al. "Item-Based Collaborative Filtering Recommendation Algorithms." *Contents: Using the Digital Library*, ACM, 3 May 2001, www.dl.acm.org/citation.cfm?id=372071.

[5] Code Link: https://github.com/raghavgr/MovieRecommenderModels