Movie Recommendation System

Author: Anna Abrahamyan
Supervisor: Eduard Mihranyan

Abstract

The rapid growth of Internet and information increases the necessity of effective recommendation systems for filtering user preferred products. These systems track customer behavior, such as purchase history, browsing activity, share habits, ratings in order to model user preference. In this paper, we introduce recommendation system model based on explicit feedback of user preferences of movies and visualize the recommendations. Then evaluate the model, and try to give explanations to the recommendation system results. In particular, show the impact of recommendation systems on our modern world.

Keywords: Recommendation system model, explicit feedback, customer loyalty, movies ratings, ALS

1 Introduction

Nowadays, recommendation systems are the key aspects of having loyal and satisfied customers. Recommendation systems are used in e-commerce, retail and media, to suggest the users items that are the most relevant to their preference. They have been under development since the early 1990s. The most important function of recommendation systems is to help customers sort through large variety of offered products and easily find the ones they prefer. The length of subscription to a website is related to the number of recommended movies they enjoyed. If subscribers fail to find movies that interest and engage them, they tend to cancel their subscription. Connecting subscribers to movies that they will enjoy is therefore critical to both the subscribers and the company. The most famous e-commerce sites that are famous with their recommendation systems are Amazon, Spotify, TikTok, Meta, and Netflix. In 2006, Netflix held a competition, with prize money of 1 million USD dollars to the group or individual who would improve their recommendation system algorithm. Netflix provided 100M ratings of 17K movies by 500K users, in the form of a triplet of numbers: (User,Movie,Rating). (Funk, 2006) More than 40000 teams from 150 countries participated in the competition. The company reported the RMSE performance of the Cinematch model as 0.9514, a 9.6% improvement over simply predicting individual movie averages. The Prize would get the team with a system that can improve on that accuracy by

an additional 10%. (Bennett et al., 2007) The competition lasted more than 2 years, with Progress prizes every year for the best result. In 2007, Korbell team won the first progress prize with 8.43% improvement. The best results were given by matrix factorization(SVD) and Restricted Boltzmann Machines (RBM) algorithms, together with 0.88 RMSE. The field is on continuous research and improvements of the algorithms.(Amatriain and Basilic, 2012)

There are two main paradigms of recommendation systems: content based and collaborative filtering. Content based relies on the features and attributes of the item. For example, in movie recommendation system, our model would suggest movies based on what the user has already watched in the past and find similar movies(with same actors, tags, directors or genre). Collaborative filtering relies on the behavior and information of other users and their ratings. (Hu et al., 2008) Recommendation systems rely on two different types of inputs: explicit feedback and implicit feedback. The most used one is explicit feedback, which includes explicit input by users regarding their interest in products, such as ratings. But as this kind of feedback is not always available researches also use implicit feedback, observing user behavior, such as browsing history, purchase history, search patterns, click or mouse movements and etc.

2 Literature Review

There are two main types of collaborative filtering, which are neighborhood models and latent factor models. (Koren et al., 2009) Neighborhood methods compute the relationships between items or users. User-oriented methods mainly distinguish like-minded users with the similar browsing history or ratings, while item-oriented methods estimate unknown ratings on the basis of similar items that tend to be rated resemblance. A product's neighbors are other products that tend to get similar ratings when rated by the same user. (Herlocker et al., 1999)

An alternative way of collaborative filtering is latent factor models that try to explain ratings by characterizing both items and users. The results, as reported on the largest available data sets tend to be consistently superior to those achieved by neighborhood models. Some of the most successful realizations of latent factor models are based on matrix factorizations. Matrix factorization models map both users and items to a joint latent factor space of dimensionality f, such that user-item interactions are modeled as inner products in that space. The high correspondence between user factor and item factor leads to a recommendation. These methods have become popular by combining good scalability with predictive accuracy. (Bokde et al., 2015) We will focus on models that are induced by Singular Value Decomposition (SVD). Applying SVD in the collaborative filtering domain requires factoring the user-item rating matrix. A typical latent factor model associates each user u with a user-factors vector $x_u \in \mathbb{R}^f$ and each item with an item-factors vector $y_i \in \mathbb{R}^f$. (Hu et al.,

2008) To learn the factor vectors the model minimizes the regularized squared error:

$$\min_{q^*,p\cdot} \sum_{(u,i)\in\kappa} \left(r_{ui} - q_i^T p_u\right)^2 + \lambda \left(\|q_i\|^2 + \|p_u\|^2\right),$$

where the k is a set of the pairs movies and items (u,i) for which the ratings, $r_u i$ is known.

There are two approaches to minimize this equation: stochastic gradient descent and alternating least squares (ALS). Stochastic gradient descent algorithm loops through all ratings and predicts r_{ui} and computes the associated prediction error.

$$e_{ui}^{def} = r_{ui} - q_i^T p_u$$

Meanwhile Alternating Least Squares approach suggests to fix one of the unknowns and solve quadratic problems. ALS rotates between fixing the q_i's and fixing the p_u's. When all p_u's are fixed, the system recomputes the q_i's by solving a least-squares problem, and then vice versa, fixing q_i and computing p_i. While in general stochastic gradient descent is easier and faster than ALS, ALS is more favorable. AlS can use parallelization. In ALS, the system computes each q_i independently of the other item factors and computes each p_u independently of the other user factors, hence is time efficient. (Koren et al., 2009)

3 Dataset and Visualizations

3.1 Dataset

The Dataset was taken from Kaggle.(Banik, 2017) It contains metadata for all 45,000 movies listed in the Full MovieLens Dataset and 26 million ratings from over 270,000 users. The dataset consists of movies released on or before July 2017. Data points include cast, crew, plot keywords, budget, revenue, posters, release dates, languages, production companies, countries, TMDB vote counts and vote averages. Ratings are on a scale of 1-5 and have been obtained from the official GroupLens website.

Here is the hyperlink for the dataset The Movie Dataset — Kaggle

3.2 Visualizations

The number of unique movies in our dataset is 45436. We have 270896 unique users who rated the movies. After cleaning the data, we can see some insights about the movie ratings.

	rating float64
count	26024289
mean	3.5280903543608817
std	1.0654427636662291
min	0.5
25%	3
50%	3.5
75%	4
max	5

(a) Summary of Ratings

	userId float64
count	270896
mean	96.06745393065974
std	205.71960644391683
min	1
25%	15
50%	30
75%	93
max	18276

(b) Summary of Users

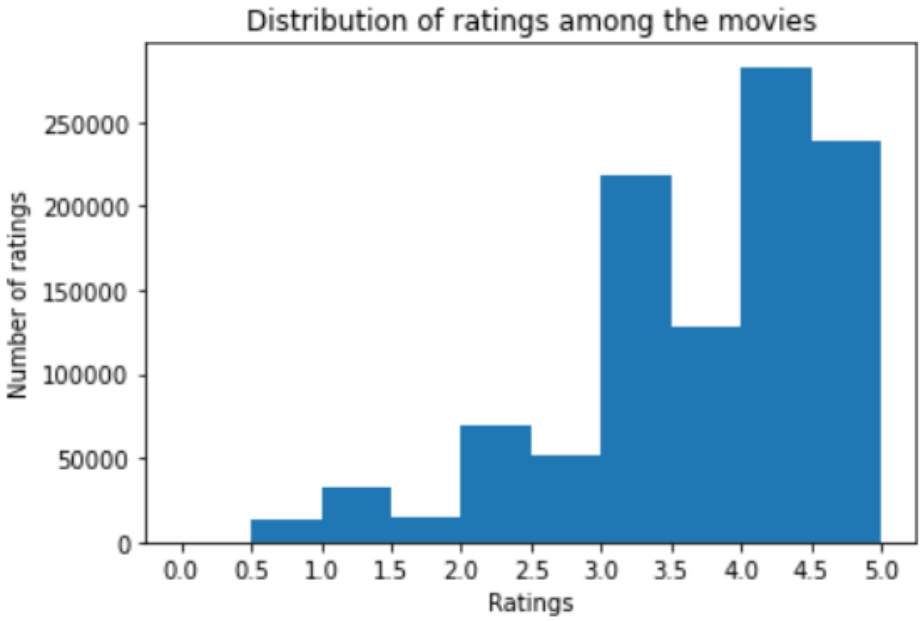
	id float64
count	45115
mean	576.8433780339134
std	3037.3805816429194
min	1
25%	2
50%	8
75%	69
max	91921

(c) Summary of Movie

Figure 1: Summary Statistics

In Figure 1 a, we can see summary statistics for the ratings. Overall we have 26024289 ratings. The mean of the overall ratings that users give is 3.5. In Figure 1 b, we can see that the max number of ratings a user gave is 18276., the minimum is 1. The average number of ratings users give to movies is 96. But as we have higher std, 205, we know that the points are further from the mean. Figure 1 c shows us analysis for movies. We overall have 45115 movie ids, and the highest number of the ratings got 'The Million Dollar Hotel' movie. The mean of the ratings that movies got is 576. Standard Deviation is 3037, which again means that the points are further from the mean.

Figure 2: Distribution of ratings



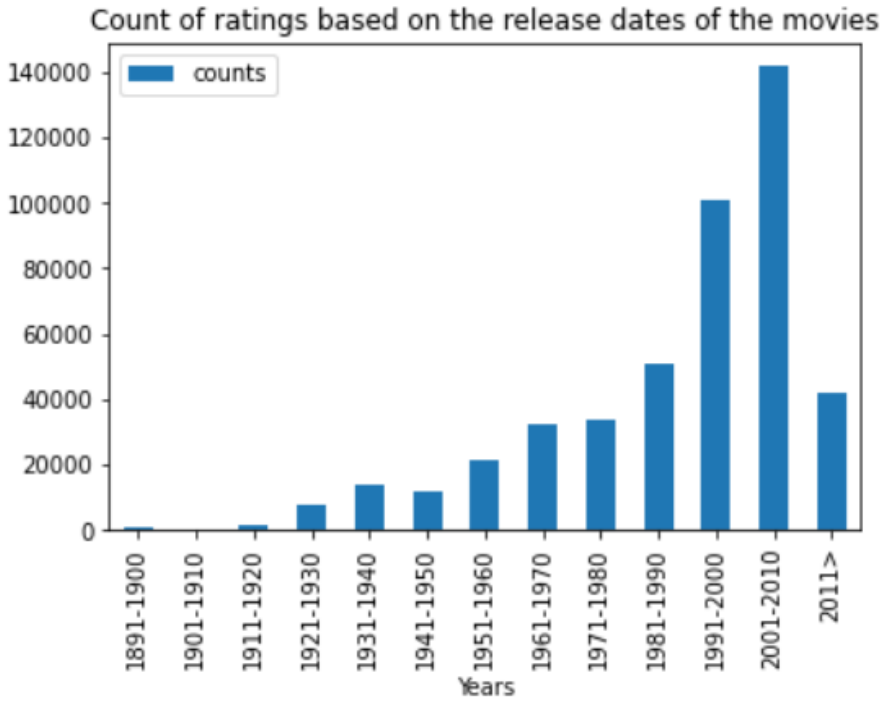
In Figure 2, we can see the number of given ratings distribution for the movies. We can see that the most frequent rating users gave to the movies is between 4-4.5.

We overall have 5084 user who only rated 1 movie, which is approximately 1.8% of the whole users.

Figure 3, shows us the distribution of movie release dates and the number of ratings they got. From our plot we can see that the most popular movies were the movies between the years 1991-2000 and 2001-2010.

In Figure 4, we can see the distribution of movie genres and the number of ratings they recieved. The mots popular genres among the users were Drama, Comedy, Action, Adventure, Crime and Horror.

Figure 3: Distribution of the top popular movies



4 Experiments

4.1 Set Up

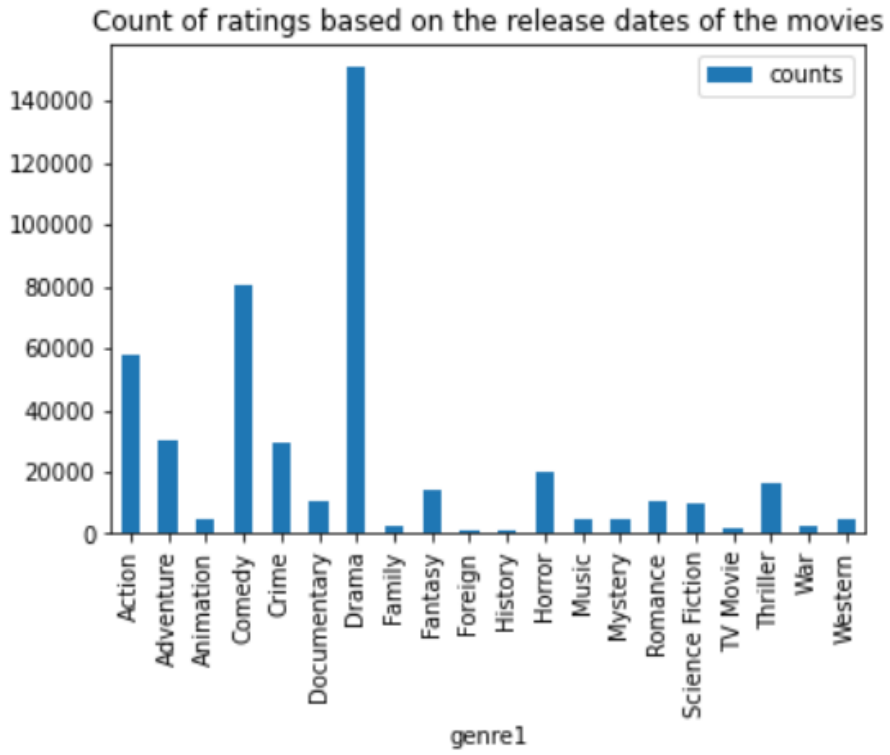
The model is implemented on PySpark using the ALS (Alternate Least Square) method. The model aims to fill in the missing entries of a user-item association matrix by a small set of latent factors. The implementation in spark.ml has the following parameters: regularization parameter, rank, iterations,etc.

Before implementing the model, we manipulated the dataset and converted it as a form of (user, item, rating) tuples. Then divided the dataset to training, validation and testing sets.

For choosing the optimal hyperparameters, we implemented grid search with a range of variables. The number of iterations was 10, as default, the range of ranks was [8,10,12,14,16,18,20] and regularization parameters [0.001, 0.01, 0.05, 0.1, 0.2]. After running the grid search with all the hyperparameters, the optimal result was with 14 latent factors and 0.05 regularization model, where the RMSE is 0.8145505115149433 and is the smallest.

Lastly, after having the best model with the optimal hyperparameters, we made a prediction and checked the testing error using out-of-sample data. Here again the RSME of rating predictions was 0.8145.

Figure 4: Distribution of the top popular movies



4.2 Results

After getting satisfactory results, the next step is defining a function for the inference. Here we have two functions. The first one takes a user's ID and gives their favourite movies(the ones they rated the highest). The second function again takes user id, top number of recommendations to show and gives the recommended movies.

After having recommended movies for 20 distinct users, we scraped the IMDB webpage to get the corresponding images to visualize the recommendations. Movie posters are very important as they contain the message and feeling of the movie and help users to decide whether they want to watch the movie or not.

Figures 5-12, demonstrate the results of the model, show 8 distint users, their favorite movies and the recommended movies by our model.

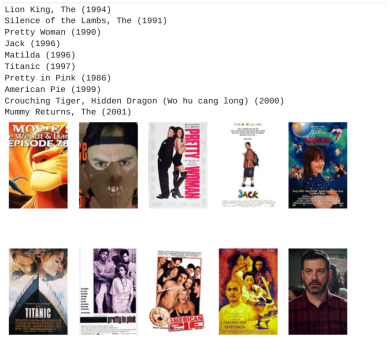
The recommended movies for the users vary in genres, years and even popularity, to give users excitement. And the users won't get bored by getting the same popular movies all the time as recommendation and will have more diverse choice.

(a) User 148369

(b) User 259478

(c) User 20022

(d) User 8754



(e) User 99975

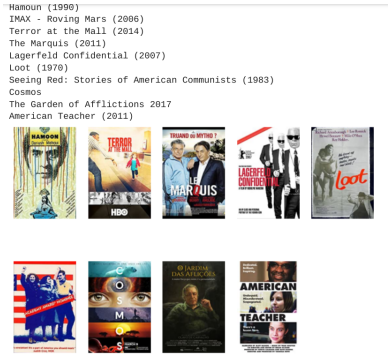
(f) User 111156

(g) User 7773

(h) User 5

Figure 5: Favorite movies of 8 users

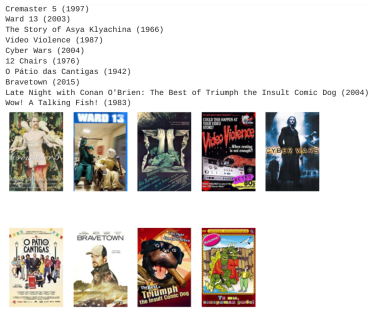
(a) User 148369



(b) User 259478

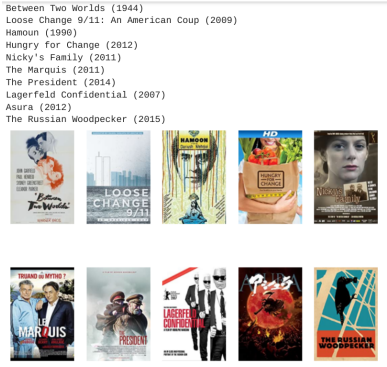
(c) User 20022

(d) User 8754

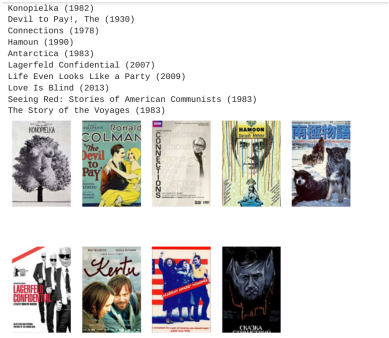


(e) User 99975

(f) User 111156



(g) User 7773



(h) User 5

Figure 6: Recommended movies of 8 users

5 Conclusion

Recommender systems have gained and continue to gain importance, as information in e-commerce increases rapidly. A lot of streaming services provide leading recommendation systems, but the industry still grows. In this paper, we developed a collaborative filtering approach for movie recommendations. The model provides users with personalized movie recommendations and also visualizes them. Specifically, we used latent factors method with ALS model. The model shows the top 10 recommended movies for the specific user and also visualizes their 10 most favorite movies. The results of the experiments on MovieLens datasets show that the model achieved great performance (measured by RMSE) comparable with other recommendation algorithms based on matrix factorization.

6 Future Work

For future work, one way to improve the recommendation system is to update the dataset with latest films, this will help to be able to analyze the results more efficiently. Another way to improve overall recommendation systems, would be to use more advanced matrix factorization models by Deep Learning. (Cheng et al., 2016) For instance being able to include time and date in the model. Having time and date in the recommender system would highly effect the customer staisfaction, as it would take into consideration seasonal preference effects. For example in December people would be more prone to watching Christmas movies, meanwhile in the summer or in the afternoon people will prefer to watch fun and easy movies, mainly containing adventures and romances.

References

Simon Funk. Netflix update: Try this at home, 2006.

James Bennett, Stan Lanning, et al. The netflix prize. In *Proceedings of KDD cup and workshop*, volume 2007, page 35. Citeseer, 2007.

Xavier Amatriain and Justin Basilic. Netflix recommendations: Beyond the 5 stars (part 1). *Netflixtechblog*, 2012.

Yifan Hu, Yehuda Koren, and Chris Volinsky. Collaborative filtering for implicit feedback datasets. In *2008 Eighth IEEE international conference on data mining*, pages 263–272. Ieee, 2008.

Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, 2009.

Jonathan L Herlocker, Joseph A Konstan, Al Borchers, and John Riedl. An algorithmic framework for performing collaborative filtering. In *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 230–237, 1999.

Dheeraj Bokde, Sheetal Girase, and Debajyoti Mukhopadhyay. Matrix factorization model in collaborative filtering algorithms: A survey. *Procedia Computer Science*, 49:136–146, 2015.

Rounak Banik. The movies dataset, 2017.

Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, et al. Wide & deep learning for recommender systems. In *Proceedings of the 1st workshop on deep learning for recommender systems*, pages 7–10, 2016.