# Deep Reinforcement Learning for Movie Recommendation Systems

**Vishnu Chhabra**
The Ohio State University
chhabra.67@osu.edu

**Dinh Song An Nguyen**
The Ohio State University
nguyen.2687@osu.edu

**Marcelo Morales**
The Ohio State University
morales.374@osu.edu

## Abstract

This report discusses a novel approach to recommender systems for movie recommendations. Our approach utilizes a deep reinforcement learning based model, built upon group recommendations and an actor-critic framework, utilizing Q-learning. We then compare our methods to some accessible base methods, i.e., standard value decay, K-nearest neighbors. Finally, we compare our methods with current state-of-the-art technology and propose improvements to our current work.

## 1 Introduction

With the increased consumption of media in today's world, recommender systems are becoming more common and important. Recommendation systems play an important role in the business world. A good recommendation system leads to an improved user experience and customer satisfaction (Afsar et al., 2022; Batmaz et al., 2019; Srinivasan, 2019 [Online]). In other words, the more efficient a company's recommendation system is, the more likely it is to stay ahead of the competition and generate more revenue.

The two main paradigms of the current popular recommender systems are collaborative filtering and content-based filtering. Traditional approaches to recommender systems are to treat them as classification or prediction problems, as is the case with collaborative filtering and content-based filtering. Specifically, reinforcement learning (RL) methods can address the sequential problem associated with traditional models. Other methods, such as collaborative neural filtering and different deep-reinforcement learning (DRL) algorithms, can also successfully address some of these limitations with additional improvements to current models. DRL has been applied to the recommender system in areas such as news recommendation, but not movie recommendation (Zheng et al., 2018).

## 2 Literature Review and Related Work

### 2.1 Movie Recommender System

Current state-of-the-art methods for film recommender systems, especially MovieLens 1M (Harper and Konstan, 2015), mainly include using kernel-based methods such as Glocal-K. Essentially, this attempts to compress the large size of the user's preferences into a small matrix of important features that describe the user's preferences, mainly using an autoencoder for this purpose. (Han et al., 2021). Another approach that gives great results on this issue is to use transformers, as observed in SSE-PT, which provides sequential recommendations using personal transformers.(Wu et al., 2020). The aim of our project was to create a model that competes with these state-of-the-art methods in terms of accuracy.

### 2.2 DRL in Recommender Systems

Deep reinforcement learning has been applied to news recommendation (Zheng et al., 2018). However, we do not see widespread application of it in movie recommendation systems. This is because of the complexity of creating an appropriate environment due to the sparseness of the data. Another issue we see across the board is the issue of scalability. Reinforcement Learning based systems do not scale, especially for problems in recommender systems (Afsar et al., 2021). A solution to both of these problems is to use an actor-critic framework, which is shown in a section below. Some work has been seen that utilizes deep reinforcement learning in movie recommendations. However, it does not use inter-group members interactions, and the user-item embedding and deep reinforcement layers cannot be trained simultaneously, which leads to unstable and low performance (Liu et al., 2021). In summary, many deep reinforcement learning-based models have been used for problems in recommen-

dation systems, but not many reliable and stable models have been found for movie recommendation systems in particular. (Afsar et al., 2021).

## 3 Data

For this project, we used the Movielens 1M data set (Harper and Konstan, 2015), a data set managed by the movie recommender system, Movielens. The dataset consists of User ID and relevant user information such as gender, profession, timestamps (time users are rating after logging in), Ratings on a scale of 1 to 5, and movie information such as Movie ID, genre and title.

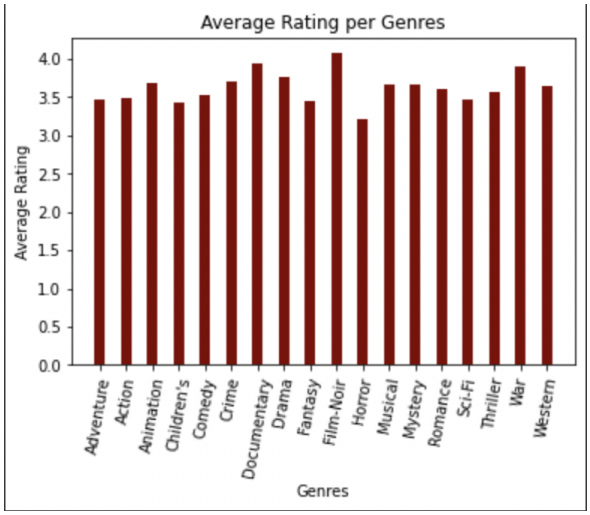


Figure 1: Average rating per genre. The 'Film-Noir' genre is the highest rated genre in this dataset.

Upon analysis, we found that *Film-Noir* is the genre with the highest rating as shown in Figure 1.Additionally, we also consider which genre has the highest user interaction, which can be seen in Figure 2. Using the previous figures, we see that higher ratings may not indicate increased interaction with users.

Before putting data in the RL environment, we generated an input data set based on the Movielens 1M data set. To do this, we randomly group users into bins. For each bin, if a movie gets rated by all users, that movie will get a score of 1. If a movie is rated by only some users in a bin or no user at all, the movie gets a score of 0. The idea of binning, grouping users together into smaller bins, is useful to get the trajectory of the user's actions. This trajectory can later be used to create the Markov Decision Process for the environment (Mate et al., 2021; Liu et al., 2021).

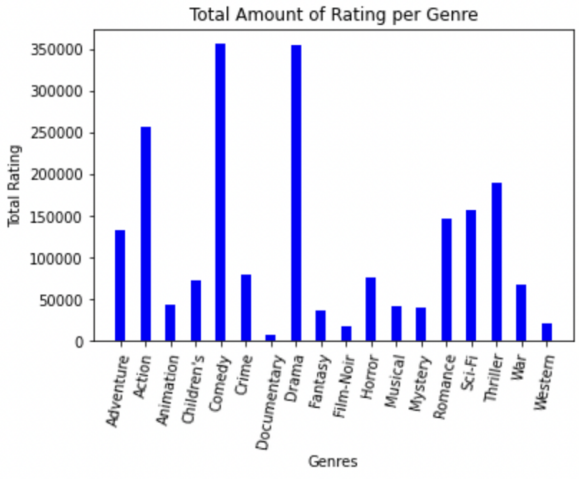


Figure 2: Bar chart shows which genre is rated the most by users in the dataset.

## 4 Experiment Setup

Since we divide users into bins, the goal of the recommender is to essentially perform a classification of a good recommendation. Two metrics used to measure the effectiveness of a recommendation were recall and normalized discounted cumulative gain. Training and testing were split with the standard $70/30$ split, respectively.

Our problem was modeled using the Partially Observable Markov Decision Process (POMDP). The POMDP is similar to the usual Markov Decision Process (MDP), but we do not know the entire state currently; we only know it partially. The MDP consists of the tuple $(S, A, T, R, \Omega, P)$, in which

- $S$ represents a set of states. A state $s_t$ is the state at time $t$. A state has a group $g$ and the historical trajectory $h_t$ at time $t$. The historical trajectory $h_t$ is derived from the history of reviews, browsing, and interactions between users in the group. The group $g$ consists of at most five users $\{u_i | i \leq 5\}$.

- $A$ represents actions set $a_t = \{m_{t,N} | N \in \mathbb{N}\}$ the list of $N$ movies that is recommended at time $t$.

- $R$ represents rewards set $r_t \in [0, 1]$ at time $t$. If $r_t = 1$, the movie is well received by the group, and 0 is not. The reward function is formally represented as

$$R : S \times A \rightarrow [0, 1]$$

meaning the reward $R$ is obtained by moving state $S$ using action $A$.

- $\Omega$ is observation probabilities based on the observation of features derived from users: average ratings given by a user, average ratings given to a movie, average ratings given to a genre, average ratings given by an age range, and average ratings given by gender.
- $P$ is transition probabilities. $P(s', a, s)$ is the probability of transitioning from state $s'$ to state $s$ using action $a$.

### 4.1 Environment

The environment consists of multiple groups of users in which its members are the users attained from the data set. We set up our environment to stimulate a POMDP. However, since the dataset is sparse, some users have not rated some films (because they haven't watched them), so to give values to this unknown rating, we use matrix factorization to predict this unknown rating. Thus, our environment can return a reward when the agent provides an action for an unknown rating.

### 4.2 Agent

We used an **Actor-Critic** framework to assign rewards. Our Actor-Critic framework is built similarly to another movie recommendation system that utilized a group-based approach and trained the agent using the Deep Deterministic Policy Gradient (DDPG) algorithm (Liu et al., 2021). Unlike this work, our agent is trained using the Proximal Policy Gradient (PPO).

The agent is compromised of the following models: the actor, the critic, and the state embedding. This agent is built in a way similar to the group-based work (Liu et al., 2021). The actor takes the state as an input and outputs the weight of an action. The action is then carried out by the environment and is judged by the critics. The critic model will output a $Q$ value of the state action tuple. Lastly, the state embedding embeds one state into its embedding.

We use PPO because it is different from the popular Q-learning algorithm in the RL settings in the sense that it is an on-policy algorithm.PPO collects data and uses this data set to update policy decisions. After the policy is updated, these data are discarded (Shi et al., 2021). We thought it would be appropriate for the movie recommendation system, as selecting movies is a continuum of action.

| **SVD** | *Fold 1* | *Fold 2* | *Fold 3* | *Fold 4* | *Fold 5* | *Mean* | *Std. Dev.* |
|---|---|---|---|---|---|---|---|
| RMSE | 0.9355 | 0.9408 | 0.9399 | 0.9317 | 0.9427 | 0.93812 | 0.003985173 |
| MAE | 0.7376 | 0.7454 | 0.7383 | 0.7344 | 0.7415 | 0.73944 | 0.0037377 |
| Fit time | 8.66 | 8.59 | 9.06 | 9.76 | 9.06 | 9.026 | 0.415961537 |
| Test Time | 0.27 | 0.27 | 0.46 | 0.27 | 0.38 | 0.33 | 0.077717437 |

Table 1: Baseline results for SVD model. The result showed is the cross validation from 1 to 5 folds

| **KNN** | *Fold 1* | *Fold 2* | *Fold 3* | *Fold 4* | *Fold 5* | *Mean* | *Std. Dev.* |
|---|---|---|---|---|---|---|---|
| RMSE | 0.9794 | 0.9794 | 0.972 | 0.9829 | 0.9806 | 0.97886 | 0.003660382 |
| MAE | 0.7715 | 0.7713 | 0.7691 | 0.7764 | 0.7759 | 0.77284 | 0.002835207 |
| Fit time | 0.36 | 0.33 | 0.33 | 0.33 | 0.34 | 0.338 | 0.011661904 |
| Test Time | 2.86 | 2.86 | 3.28 | 2.97 | 2.88 | 2.97 | 0.160249805 |

Table 2: Baseline results for KNN model. The result showed is the cross validation from 1 to 5 folds.

The user's interests in a movie can change drastically and randomly, and thus choosing an on-policy algorithm helps take this into account.

## 5 Result

We implemented two baseline recommender systems, one using K-Nearest-Neighbor (KNN), and one using Singular Value Decomposition (SVD). The test results from the use of the baseline models are shown in table 1 and table 2.

The current benchmark for the movie recommender system using the Movielens 1M data set is the Glocal-K model (Han et al., 2021) which has the RMSE of 0.7208. Another benchmark model that uses NDCG is the SSE-PT model with the NDCG@10 of 0.6292 (Wu et al., 2020). This model does not perform better than either of the benchmarks. Although we believe that our model performs better than the baseline model, more careful testing is required as baseline and model testing metrics differ. Due to the results of the model with our current recall and NDCG values, we believe our hypothesis to be true, as the base model is incredibly simple.

## 6 Further Improvements

One way we could further improve this model is to add another layer to the embedding, which allows different members of the group to have different influences when the group is diverse, allowing

| | *Recall@10* | *NDCG@10* |
|---|---|---|
| Group | 0.1187 | 0.1381 |
| User | 0.0726 | 0.0062 |

Table 3: Result of running the RL model with recall and normalized discounted cumulative gains for both group and user.

for a more complex and diverse environment. Another technique we could explore is to utilize Soft Actor-Critic reinforcement learning algorithm instead of our Proximal Policy Optimization, which is a much more stable approach to this problem. As a soft actor-critic algorithm maximizes rewards while also maximizing entropy (Haarnoja et al., 2018) and, in general, leads to better performance compared to Proximal Policy Optimization. We expect that these changes would positively reflect the accuracy of the model.

## 7 Conclusion

In conclusion, we presented an approach to solving problems in recommender systems, utilizing an Actor-Critic-based deep reinforcement learning model, with a proximal policy optimization reinforcement learning algorithm, using grouping of users for the environment. We compared our method to the state-of-the-art and some very commonly used baseline methods such as SVD(Singular Value Decomposition), KNN(K-Nearest Neighbors), implemented by us, and then presented our results. We note that our method provides a lower result than the current state of the art. We also present some improvements we can make to further increase the accuracy of the model such as utilizing a Soft Actor Critic algorithm and accounting for diversification in the groups.

## 8 Contribution

Each member contributed equally to literature review, coding, experimentation, and writing reports.

- **Vishnu Chhabra** *(34%)*: Responsible for the environment, the agent, and the baseline models.
- **Marcelo Morales** *(33%)*: Responsible for data preprocessing, and the agent of the RL.
- **Dinh Song An Nguyen** *(33%)*: Responsible for data preprocessing, and the environment of the RL.

## References


Mehdi Afsar, Trafford Crump, and Behrouz Far. 2022. Reinforcement learning based recommender systems: A survey. *ACM Computing Surveys*.

Mohammad Mehdi Afsar, Trafford Crump, and Behrouz H. Far. 2021. Reinforcement learning based recommender systems: A survey. *CoRR*, abs/2101.06286.

Zeynep Batmaz, Ali Yurekli, Alper Bilge, and Cihan Kaleli. 2019. A review on deep learning for recommender systems: Challenges and remedies. *Artif. Intell. Rev.*, 52(1):1–37.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. 2018. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. *CoRR*, abs/1801.01290.

Soyeon Caren Han, Taejun Lim, Siqu Long, Bernd Burgstaller, and Josiah Poon. 2021. Glocal-k: Global and local kernels for recommender systems. *CoRR*, abs/2108.12184.

F. Maxwell Harper and Joseph A. Konstan. 2015. The movielens datasets: History and context. *ACM Trans. Interact. Intell. Syst.*, 5(4).

Zefang Liu, Shuran Wen, and Yinzhu Quan. 2021. Deep reinforcement learning based group recommender system. *CoRR*, abs/2106.06900.

Aditya Mate, Lovish Madaan, Aparna Taneja, Neha Madhiwalla, Shresth Verma, Gargi Singh, Aparna Hegde, Pradeep Varakantham, and Milind Tambe. 2021. Field study in deploying restless multi-armed bandits: Assisting non-profits in improving maternal and child health. *CoRR*, abs/2109.08075.

Bichen Shi, Elias Z. Tragos, Makbule Gulcin Ozsoy, Ruihai Dong, Neil Hurley, Barry Smyth, and Aonghus Lawlor. 2021. Dares: An asynchronous distributed recommender system using deep reinforcement learning. *IEEE Access*, 9:83340–83354.

Aishwarya Srinivasan. 2019 [Online]. Recommendation systems using reinforcement learning. Medium.

Liwei Wu, Shuqing Li, Cho-Jui Hsieh, and James Sharpnack. 2020. Sse-pt: Sequential recommendation via personalized transformer. page 328–337.

Guanjie Zheng, Fuzheng Zhang, Zihan Zheng, Yang Xiang, Nicholas Jing Yuan, Xing Xie, and Zhenhui Li. 2018. Drn: A deep reinforcement learning framework for news recommendation. In *Proceedings of the 2018 World Wide Web Conference*, WWW '18, page 167–176, Republic and Canton of Geneva, CHE. International World Wide Web Conferences Steering Committee.