Implementation and Analysis of Recommender Systems

Group 9. Project 19

Mudit Verma
1217422453
mverma13@asu.edu

Swetha Kondisetti
1222643058
vkondise@asu.edu

Sumair Bashir
1222371254
sbashir3@asu.edu

Suresh Murali Sankar
1219289019
smuralis@asu.edu

Ritvik Gaur
1219379213
rgaur3@asu.edu

Sai Kiran Bommareddy
1222782522
sbommar4@asu.edu

***Abstract*—This document is a intended to record and report the findings of the Semantic Web Mining course group project. The objective of the project is implementing recommendation systems using multiple methods to analyse and compare the efficiency of each against a baseline model implementation using various metrics. More precisely, the baseline model would be implemented using matrix factorization technique. This is used to compare the other two methods explored - Deep Neural Network based approach and reinforcement learning method. The MovieLens data set is used as data source to train and test the implemented methods.**

***Index Terms*—Recommender systems, Matrix Factorization, Reinforcement Learning, Deep Neural Networks**

I. Introduction

The course acquainted our team the techniques and methods used in present day businesses that rely on recommendation systems such as - streaming platforms, informational media guides, e-commerce, blogs etc. The common challenge with most of the present day consumer services are predominantly customization and recommendation of products based on customer profile. An effective recommendation system need to factor in parameters that reflect the personality or profile of the user, as well as a good understanding of the options that could be recommended to the user, along with the context at the time of use. The primary issues discovered were - copious amount of data to analyse, biased user data, semi-structured format of the data. In the following section, we would try to delineate the approach taken to design and evaluate the each method of implementation.

II. Problem Statement

The goal of this project is to design and implement a Movie Recommendation system based on various algorithmic approaches to evaluate, compare and contrast each of them based on various metrics to determine their overall effectiveness. It will also demonstrate how well each of these algorithms mitigates or suffers from common problems that arise when designing a Recommendation System, such as the cold start problem, non-stationary customer preferences, data privacy and user profiling, robustness to negative sampling, model compatibility, reproducibility etc. At the end of the project we expect to have a detailed analysis based on the said performance measures over three paradigms of recommender systems, the traditional Collaborative Filtering method (utilizing the subclass of Matrix Factorization), a Deep Neural Network based method and finally a Reinforcement Learning based method.

III. Related Works

To understand and evaluate various implementations of recommendation systems, we need to perform review of the data source domain and the related literature review for awareness of implementation techniques and algorithms.

A. Literary review summary

We performed review of recommendation systems from three fronts, Collaborative filtering methods, deep learning based solutions and finally, deep reinforcement learning based approaches.

A popular sub-class of collaborative filtering algorithms that has been successful is Matrix Factorization. This relies on decomposing the user-item interaction matrix. A intuitive understanding of vector representation and the effects implied through factorization and decomposition proved critical in understanding incisively the approaches to handle explicit interactions - like ratings as in Frunk MF [9], or implicit interactions - such as ”likes, bookmarked” as in SVD++ [10]. The primary motive of this review was to evaluate the detail of user data embedding or to say distinctively identify data based on the differences in embedded knowledge. Here we are concerned about the implementation of a model for primarily producing user rating prediction and Top-K recommendations results.

For further understanding of constraints with implementing recommendation systems, common one’s like dimensions of

the test and train data in deep learning model training, optimizing the efficiency of parameter tuning for DL (Deep learning) methods, references to prior research paper provided an good understanding of the same (Natural Collaborative Filtering [7], Neural Network Matrix Factorization [3]). Common preprocessing included removing duplicates, removing biased data. The literary review helped in initial feature reduction by introducing compounded/ derived columns.

The other end on the spectrum of deep learning approaches have typically been end-to-end strategies which model the problem as that of supervised learning (Deep Factorization Machine, [5]). There has also been some surge in utilizing the Graph Neural Networks for modeling the high level interactions. This framework makes it easy to integrate user/item information as a social network

Reinforcement Learning is another branch of AI that has been leveraged for the problem of recommendation systems. The idea is to utilize feedback from an environment where the system (or agent in this case) is acting. These feedbacks could be simulated from existing data or could be given in real time by users in the loop. In many cases, the recommendation system decision making is formulated as a finite Markov Decision Process that recommends an item according to the current state (and keeps no memory of the past, thus Markovian). Several popular RL algorithms have been adapted to be used for the recommendation systems like Policy Search (for example Actor-Critic models, REINFORCE [1], [4], [14]), or Value based methods (like DQN [2]). A direct benefit of utilizing an RL approach over single-step decision making like classification is the richer expressive power in modeling the state and reward feedback, for example as in the KERL [12] framework the authors could incorporate a knowledge base into the reward function to address sparsity and cold-start.

A general problem in all the above methods has been one of negative sample. Discovering informative negative feedback from missing data is hard, especially in the case of implicit interactions. Several works exist, like KG policy network [13], that posit different ways of generating negative samples from data that are further fed to above-mentioned algorithms. As we will realize through our exploratory analysis that the reinforcement learning algorithm indeed helps with the negative sampling. Additionally, the RL approach also helps with the possible distribution shift in the human preference data with its ability to gradually adapt with changing preferences.

IV. System Architecture & Algorithms

The primary objective is to evaluate the performance of different techniques implemented on top of the baseline model and that gave an good understanding of the strengths and flaws of each implementation method. This helped to implement hybrid systems with a more dynamic recommendation - based on the context and query.

The initial phase of the project was about a concrete understanding of the domain and pre-processing or grooming of data. This helped us in making sense of the basic analysis on the data - for example observing a set of user rating data under demographic constraints and other attribute filters gave an intuitive idea of user mentality towards rating, genre they choose to watch, and sometime a part of context of environment (Why some movies got high ratings during some parts and time in the past relating to social events pertaining to the period in scrutiny). This analysis proved critical in grooming the data for each implementation. By introducing derived features to the dataset to involve bias for genuine reviews, removing duplicates and redundant dataset.

The overview diagram explains our system architecture in Figure 1. We first obtain the dataset, that is the movie lens dataset for several users' rating on several movies. We then filter the data to remove any outliers, missing data points, and add several fields like average rating. As part of the preprocessing step we also bin several continuous fields like age into discrete buckets. We then train three different methods, the Matrix factorization method that takes in the User Matrix and the Movie-Item Matrix as input, the Neural Collaborative Filtering approach that takes in the embeddings of the user-item matrices and finally the Reinforcement learning approach that requires environment semantics. Finally, all of these algorithms have been implemented with a similar interface to support the demo-webpage to recommend top-K movies to a given user. We also allow the system to take in a user-id and a movie-id information to predict the rating that the user might attribute to that movie.

A. Matrix Factorization

The matrix factorization algorithm is an extremely popular algorithm for recommender systems and became particularly popular during the Netflix Prize challenge. There exists many variants of the Matrix Factoriazation approach, for example the FunkMF, SVD++, Assymetric SVD, Group specific SVD to name a few. Since we are using the matrix factorization algorithm as a baseline, we implemented the vaniall MF (FunkMF) algorithm.

The idea of using matrix factorization as a baseline is that it intuitively represents the projection of user preferences to their ratings to extract values of un-rated movies without any convolutions as such in AI models. The other factors were the efficiency and time taken to perform these operations. As we are dealing with matrices with multiple columns, the idea is to decompose the User-Item Matrix (R) into User-Features (P), Features-Item (Q) matrices, where Features are the latent features. Matrix Factorization/ decomposition are methods that reduce a matrix into constituent parts as by the following equation :

$$R \sim P \times Q^T = \tilde{R} \tag{1}$$

(Eq. 1) that elucidate calculations of complex matrix operations. Here we obtain the constituent matrices P & Q using gradient descent.

B. Deep Neural Network based Collaborative Filtering

Collaborative filtering is a good technique to identify ranking of in a collection of items based on the user interest. We can use collaborative filtering to identify lists like most

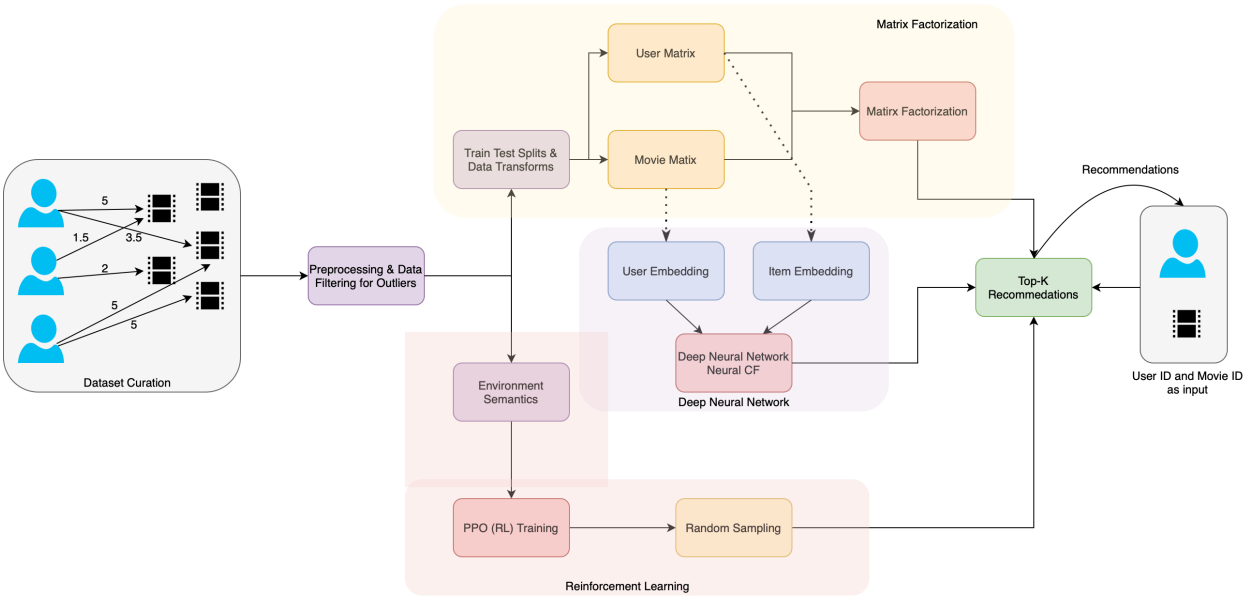

Fig. 1. System Architecture and Proposed Algorithms

watched, most anticipated, general top-k recommendation. The matrix factorization task discussed previously would embed the constituent matrices in a latent space. We would use these embedded matrices to perform collaborative filtering (Fig. 2, 3).

In this approach, similar to matrix factorization method, we allow non-linear embedding space. For this solution we take advantage of DNN (Deep Neural Network) ability to be efficient in predictive analysis. Based on the data the network is provided, it can predict about future action or missing ratings in our case.

We have implemented the rating prediction using DNN using Pytorch. PyTorch is one of the popular deep learning frameworks for AI and Machine Learning. Pytorch uses tensor data structure and uses a training algorithm called back propagation.

In the implemented model, we use six fully connected layer and a Relu activation layer and a dropout layer. The number of connected layers and the dropout constraints are instructed by the primary evaluation by comparing with the matrix factorization method results.

Additionally we also experimented with popular libraries such as tf_recommends to obtain Deep Neural Network performance benchmarks and found that our implementation at par with other DNN implementations in terms of performance.

C. Deep Reinforcement Learning Approach

We model the problem of recommender system as a Markov Decision Process as $\mathcal{M} =< S, T, A, R >$ where S is the state space is a tuple of ¡Mean Movie Rating, Movie Embedding, User Embedding, Mean User Rating, User Occupation, User Gender Bucket, User Age Bucket¿. The actions are to predict the rating the user would give for a given movie. Note that, the fact that we train the data for all the users in conjunction

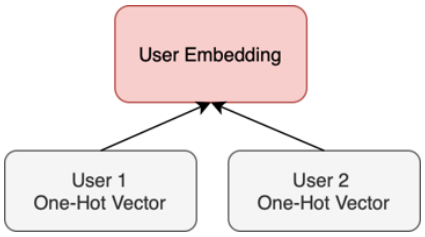

Fig. 2. Vector embedding of user matrix for the Deep Neural Network based Neural Collaborative Filtering

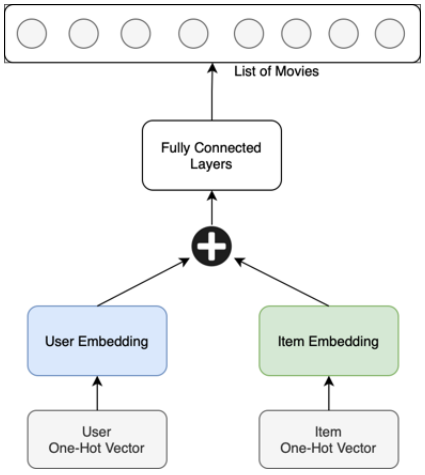

Fig. 3. Using embedded matrices in collaborative filtering for DNN approach

makes this process a collaborative filtering approach, where the system takes into account similarities between users to determine the rating they would give to a given movie. Finally

our reward function is as follows :

$$R = \begin{cases} 1 & y_p = y \\ \frac{\log(y_p - y)}{2} & \text{otherwise} \end{cases} \quad (2)$$

The objective here is to be able to predict the rating a user u_i will give for a movie m_j. Once we predict the movie rating r_{ij} for the problem tuple $< u_i, m_j >$ we are ready to recommend k movies to the user. In the held out set that we use to recommend movies, we first randomly sample 100 movies. Within this set of 100 movies we rank them by the rating prediction function via our RL approach. We then finally pick the $top-k$ movies. A major advantage of the random sampling before we finalize our top-k recommendations, we realized is in the coverage metric as shown in the results section. We find that this random sampling significantly improved the catalog-coverage with marginal effect on other metrics like top-10 accuracy, RMSE, MSE etc.

For training the rating prediction function, we utilized the Proximal Policy Optimization algorithm. This is an on policy algorithm that tries to take small gradient updates for the underlying actor-critic neural network. This algorithm is amongst the best performing algorithms in the field of reinforcement learning, especially in the cases of rating distribution drift. A common problem with recommender systems is the fact that the human preferences are likely to drift with time. The Reinforcement Learning approach helps with this problem since algorithms like PPO are more robust to distribution shifts in the reward functions (which in effect correspond to the human preferences).

V. Datasets & Preprocessing

The MovieLens dataset, widely used in research and education, is the preference. Description of movies by users expressed in the form of (user, item, rating, timestamp) tuples. We used the "ml-latest-small" dataset. It describes the 5 star ratings, genres and descriptive short phrase's metadata. The dataset, available as comma-separated-values files(links.csv, movies.csv, ratings.csv and tags.csv), contains 100836 total ratings for 9742 movies created by 610 distinct users. The ratings are described as ten unique values in the increments of 0.5 in the range of 0.5 - 5.0. The tag expressions allow a user to associate short phrases with movies rated by them such as "Boxing story", "way too long" or "holocaust" etc. Each user has rated at least 20 movies and are associated with a user-ID. Similarly, each movie is associated with an ID and the movie-ID's map across the dataset unlike user-ID's for whom a global mapping is not present. Additionally, each movie is attributed to at least 1 genre and at most 18 genres [6], [11]. An environment-semantics wrappers around the dataset was curated, depending upon the paradigm we considered for each implementations . We converted the dataset to showcase implicit interactions for Deep Neural Network based approaches and formed a wrapper around the dataset to conform it to environment / agent semantics required by reinforcement learning algorithms. For these methods libraries like PyTorch were used to create the dataset by extracting the raw data from the CSV files and transforming it into a dataset that is compatible with the training framework used for that method. For example the DNN utilized the PyTorch Lightening module to aid with training of its model from the transformed dataset.

We note that during our exploratory analysis, most of the values in the rating matrix are unknown as users have not rated the majority of the movies. We compute a metric known as sparsity of the dataset defined as :

$$S_D = 1 - \frac{non\_zero\_entries}{\#users * \#movies} \quad (3)$$

We find that the sparsity of the MovieLens dataset is extremely high 93%. This shows the extent of the difficulty of the problem of recommending movies, especially with sparse data.

For the RL algorithm, the preprocessing steps requires that we convert the dataset into environment semantics popularly accepted by Deep RL frameworks. To do this, we parsed the data to create an internal state of the environment. The environment starts from a random user-id. Then we scan the dataset for all the movies rated by the given user-id. These ratings are sequentially processed and the RL agent uses this to compute the reward function. After a scan for a particular user, we then randomly sample a user and repeat this process. Readers can find more details on this in section IV-C.

VI. Evaluations

A. Models used for performance evaluation

For our project, we'll use three models and compare their performance using metrics. The first model is the matrix factorization, which is the baseline model. Matrix factorization is useful for determining the relationship between users and movie matrices, as well as making predictions using both item and user entities.

The second is the Deep Neural Networks (DNN) model. DNN models can be used to overcome the limitations of matrix factorization. Due to the flexibility of the network's input layer, DNNs may readily add query and item features, which can assist in capturing a user's individual interests and increase the relevancy of suggestions.

Third, the Deep Reinforcement Learning (DRL) model. DRL considers a recommendation system as a sequential decision-making procedure, unlike other static recommendation methods. This system is a dynamic adaptation that operates for a long-term objective with changing user preferences. It models the interactions between users and the recommender system using a user-rating reinforcement learning scheme.

B. Evaluation Metrics

Every recommendation system needs an customized evaluation metric to assess the effectiveness of the proposed approaches.

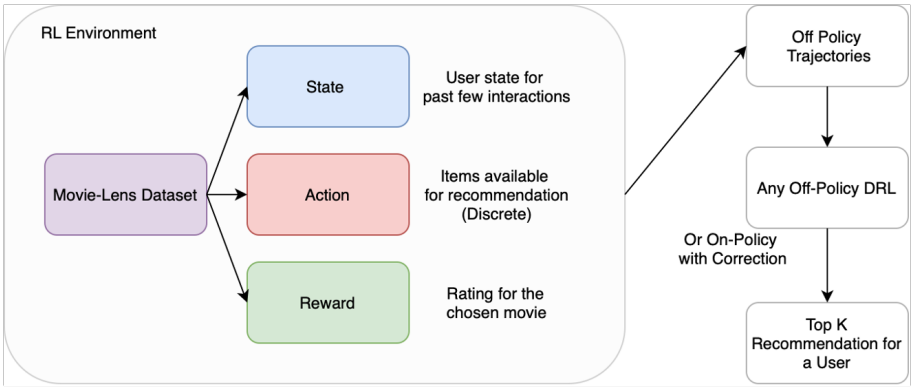

Fig. 4. Deep Reinforcement Learning architecture for Recommender system. We propose the environment semantics required by the RL agent followed by utilizing an off-policy RL algorithm, or an on-policy algorithm with correction, for example, Proximal Policy Optimization. Finally, we generate top-k Recommendations for users.

RMSE: We will use RMSE as one of our measures for evaluating the three models in this project, and it is one of the most often used evaluation metrics for recommender systems. RMSE metric is the square root of the average squared difference between the expected and actual values in a dataset. It is the deviation from the actual recommendation of the model. It's calculated as follows:

$$RMSE = \sqrt{\sum_{i=0}^{n}(P_1(u,i) - P_2(u,i)^2/n}$$

where
$P_1(u,i)$ is the predicted rating of user u of movie i,
$P_2(u,i)$ is the expected rating of the user u of movie i and n are the number of samples.

Top-k recommendation accuracy: This metric assesses how often the recommender system matches the user's actual selection. It determines whether the predicted k-recommendations match the user's selection; if so, the recommendation is considered successful. Top-K recommendation accuracy is the ratio of successes on a validation dataset Xval and it is given by the formula:

$$top - Kaccuracy = 1/(X_{val}) \sum_{x_i \epsilon X_{val}} 1_{x_i \epsilon rec_k(x_i)} \quad (4)$$

where,
$X_{val}$is the cardinality of the validation set,
$1_{x_i \epsilon rec_k(x_i)}$ is one if the k-recommendations include the true item xi , otherwise it is zero.

Catalog Coverage: It counts the recommendations suggested by the algorithm on the validation set, to see if the recommendation has been predicted by any input in the catalog Xcat. The catalog coverage can be calculated as:

$$catalogcoverage = (1/|X_{cat}|)|\{\cup_{x_i \epsilon X_{val}} rec_k(x_i)\}_{\neq}| \quad (5)$$

where
$\neq$ is used to denote that the cardinality is taken over the set of all unique items in the union of recommendations

C. Evaluation Results

The plots from Figure 5 through 12 show various plots for RMSE of Matrix Factorization followed by top-1, top-10, top-100 accuracies for the DNN method, and finally the KL Divergence, Value function loss and Rating Prediction accuracy of RL method. Finally, the table VI-C shows the various metrics like MSE, RMSE, top-1, top-10 accuracy values, catalog coverage values for all the three methods (MF, DNN and RL). The table also reports the average reward received by the RL agent and the Rating Prediction accuracy.

Matrix Factorization Evaluations:
- Final MSE: 1.13
(Fig. 5)

Neural Network Evaluations: -
MSE : 1.07809
-
RMSE : 1.03831
(Fig.: 6 7 8)

RL specific metrics evaluation:
Since RL Primary Task is Rating Prediction : Exact Prediction Accuracy is 35%
(Fig.:12, 9, 10, 11)

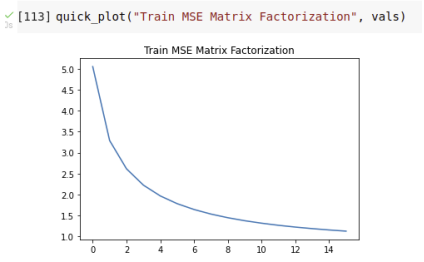

Fig. 5. Plot of RMSE values for matrix factorization

TABLE I
TABLE: COMPARISON FOR DIFFERENT METRICS

Metric	Matrix Factoriza-tion	DNN	RL
MSE	1.13	1.07809	0.8613
RMSE	1.0630	1.03831	0.9281
Top-1 Accuracy	50%	55%	-68%
Top-10 Accuracy	70%	75%	72%
Catalog Coverage	73%	81.4%	82.6%
Avg Reward	NA	NA	0.20 of 1
Rating Prediction Accuracy	NA	NA	35%

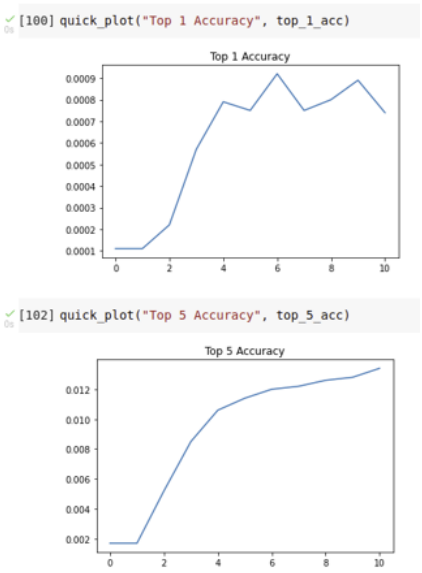

Fig. 6. Accuracy in predicting the top movie, DNN Method

VII. UI / VISUALIZATION INTERFACE DESIGNS

The User Interface to input and extract ratings and recommendations is implemented as a single page application as in Fig. 13. The input is the id of the user which who's ratings are to be predicted by the recommender system. We used Bootstrap Design front end for HTML based on Jinja templates to work with a Flask back end. Once the user puts in the required information, the flask backend engine uses

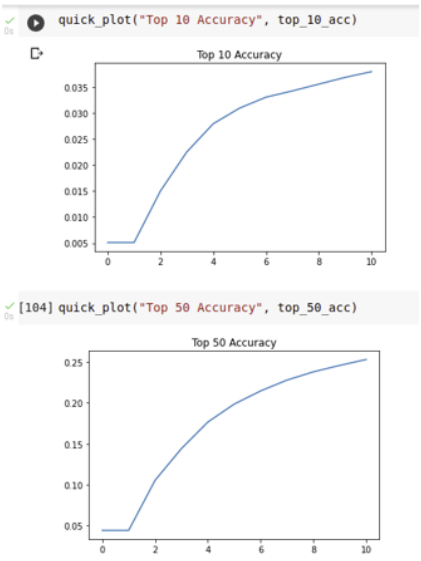

Fig. 7. Accuracy in finding predictions in top 10 movies, DNN Method

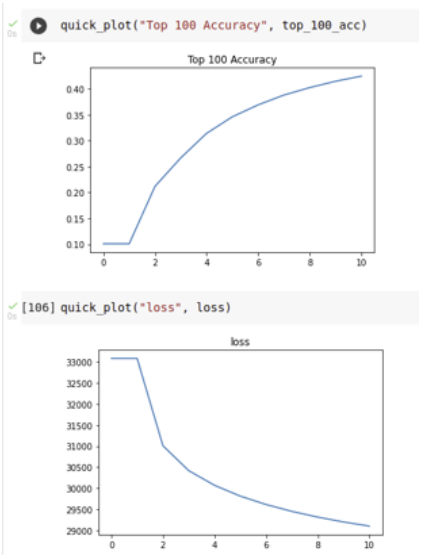

Fig. 8. Accuracy in finding predictions in top 100 movies, DNN Method

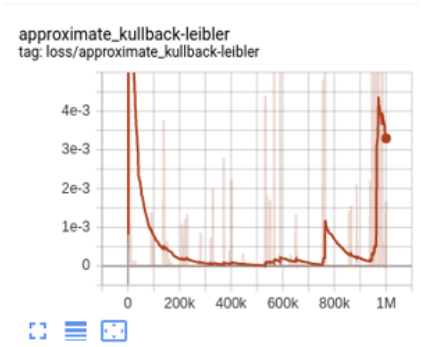

Fig. 9. KL Divergence plot for PPO training (Reinforcement Learning Method)

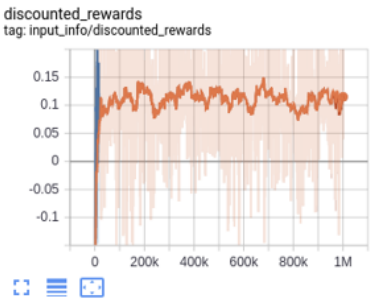

Fig. 10. Expected reward function throughout the training of RL algorithm.

Fig. 11. Value function of the PPO training algorithm, RL Method

Fig. 12. Console message returning the accuracy - 0.341 $\approx$ 35%, RL Method

the pre-loaded models for Matrix Factorization, DNN and RL and queries the respective model. The response is top-k recommendations for the user and predicted ratings (if the algorithm predicted any rating). Then the results are displayed in the form of a list to the user in the web interface.

VIII. Division of Work and Team Members' Contributions

The division of work between the team members of the project are as indicated in the Table II.

TABLE II
TABLE: TASKS SUMMARY AND MEMBERS

Task	Members
Literature Review	All of us discussed the literature with specific focus on contemporary research and evaluation methods
Design Discussions	MF - Kiran, Swetha, DNN - Ritvik, Sumair DRL - Mudit
Data Preprocessing	For MF / DNN : Kiran, Sumair, Ritvik For DRL : Mudit
Algorithm Implementation	MF : Kiran DNN : Mudit, Ritvik, Sumair DRL : Mudit
Testing Evaluation	Evaluation Measures : Swetha, Suresh Visualization : Swetha, Suresh Testing Algorithms : Suresh, Ritvik
Presentation	Proposal Presentation : Mudit, Suresh, Kiran, Ritvik, Sumair, Swetha Demo and User Interface: Mudit, Ritvik
Final Report	All

IX. Conclusion

In this project we have looked at three implementations of recommendation system, namely the Matrix Factorization (FunkMF), a Deep Neural Network approach (Neural Collaborative Filtering) and finally a Reinforcement Learning based approach. We find that we were able to achieve near state of the art MSE values with the RL & the DNN approach. We also find that the Matrix Factorization approach works very well, and is orders faster than the training time required for the other two approaches. Additionally, we find that the RL approach is also able to handle distribution shift of human preference, however, we found this advantage during our exploratory analysis, and a more systematic study would be required to substantiate any claims.

Additionally, we also reported several measures like RMSE, MSE, Top-1, Top-100 accuracy values, and Catalog coverage. We find that all the algorithms were able to recommend predictions which covered a large part of the possible movies that could be recommended. We find that the insights from this work has led us to several new questions be answered. Namely, the RL approach is performing at par, at times slightly better than DNN approach, however in effect the RL algorithm is performing a regression task of predicting user ratings. It is unclear about the advantages of choosing one over another. We also found that the DNN and RL approaches are very sensitive to hyperparameter tuning, especially the PPO algorithm. Matrix Factorization on the other hand is extremely robust and fast to converge. As part of future work, we propose the following additions to our current work. First, we would like to run our analysis on larger movie-lens datasets, that would give better insights into the results. Second, we would want to extend the Matrix Factorization approach and implement other

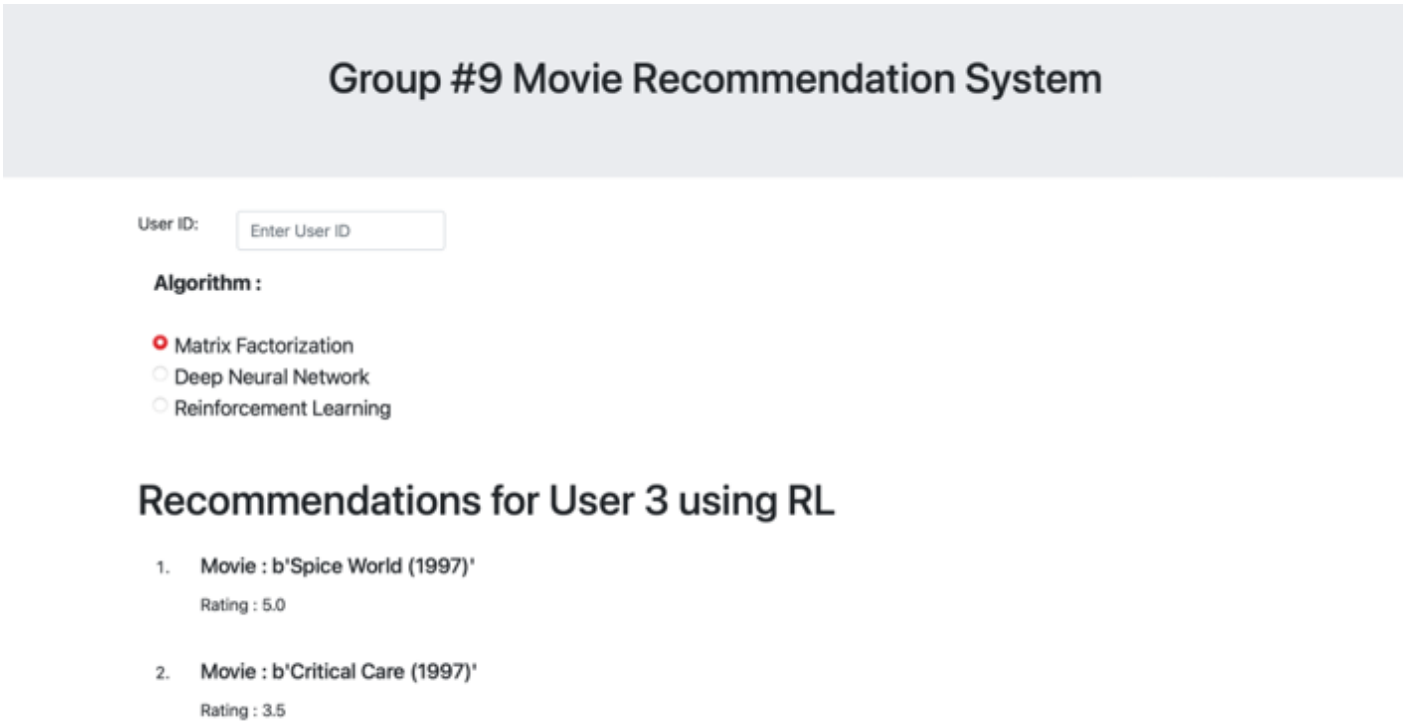

Fig. 13. User Interface for observing different model recommendations. We can enter a user ID followed by the choice of algorithm to obtain movie recommendations. We showcase that for User 3, using the RL approach the human user would receive top 10 movie recommendations.

advanced variants such as Group-MF and SVD++ which are very popular baselines themselves. Finally, we want to extend the RL based approach to take into account the context of user while predicting rewards. For example, based on the order of the movies reviewed by the human, or the order in which they watched the movie, such context information can be extremely helpful in real-time movie recommendations. A challenge to this, however, is the limited availability of such private data. Finally, we would also like to explore various visualization techniques to showcase our implementations such that these systems could be used by public in real-time over the internet.

REFERENCES

[1] M Mehdi Afsar, Trafford Crump, and Behrouz Far. Reinforcement learning based recommender systems:
A survey. arXiv preprint arXiv:2101.06286, 2021.

[2] Haicheng Chen. A dqn-based recommender system for item-list recommendation. In 2021 IEEE International Conference on Big Data (Big Data), pages 5699–5702. IEEE, 2021.

[3] Gintare Karolina Dziugaite and DanielMRoy. Neural network matrix factorization. arXiv preprint arXiv:1511.06443, 2015.

[4] Wei Fang and Fanyuan Zeng. Policy gradient for items recommendation on virtual taobao. 2020.

[5] Huifeng Guo, Ruiming Tang, Yunming Ye, Zhenguo Li, and Xiuqiang He. Deepfm: a factorization-machine based neural network for ctr prediction. arXiv preprint arXiv:1703.04247, 2017.

[6] F Maxwell Harper and Joseph A Konstan. The movielens datasets: History and context. Acm transactions on interactive intelligent systems (tiis), 5(4):1–19, 2015.

[7] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. Neural collaborative filtering. In Proceedings of the 26th international conference on world wide web, pages 173–182, 2017.

[8] J. Loy, "Deep Learning based Recommender Systems," Medium, 19-Oct-2020. [Online]. Available: https://towardsdatascience.com/deep-learning-based-recommender-systems-3d120201db7e. [Accessed: 13-Apr-2022].

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

[10] Rajeev Kumar, BK Verma, and Shyam Sunder Rastogi. Social popularity based svd++ recommender system. International Journal of Computer Applications, 87(14), 2014.

[11] Anne-Marie Tousch. How robust is movielens? a dataset analysis for recommender systems. arXiv preprint arXiv:1909.12799, 2019.

[12] Pengfei Wang, Yu Fan, Long Xia, Wayne Xin Zhao, ShaoZhang Niu, and Jimmy Huang. Kerl: A knowledge-guided reinforcement learning model for sequential recommendation. In Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval, pages 209–218, 2020.

[13] Xiang Wang, Yaokun Xu, Xiangnan He, Yixin Cao, Meng Wang, and Tat-Seng Chua. Reinforced negative sampling over knowledge graph for recommendation. In Proceedings of The Web Conference 2020, pages 99–109, 2020.

[14] Junzi Zhang, Jongho Kim, Brendan O'Donoghue, and Stephen Boyd. Sample efficient reinforcement learning with reinforce. arXiv preprint arXiv:2010.11364, page 97, 2020.