Movie Recommendation Systems

Harika Chadalavada, Harpreet Kour, Saurav Jayakumar *13th December, 2022*

1 INTRODUCTION

Recommendation systems are one of the most successful areas of machine learning applications. A recommendation system is a type of information filtering system which attempts to predict the preferences of a user and make suggestions based on those preferences. With the high volume of data being generated, movie recommendations are becoming more and more sophisticated especially post the pandemic. Commercial movie libraries effectively exceed 15 million films, which boundlessly exceeds the visual ability of any single individual.

With a large number of motion pictures to browse, individuals now and then get overwhelmed [1], and therefore businesses are not able to retain their existing customer base. An efficient recommendation system is necessary for both the business and the customer. Traditional recommendation methods are difficult to express the essential information of the data and require manual extraction of features. The effect of feature extraction often determines the performance of those algorithms, therefore in recent years, deep learning has been favored by most researchers [2].

While deep neural networks generate recommendations based on historical data of users' movie history, it does not provide a solution to long-term user engagement. Reinforcement learning can help by introducing exploration which lets the model learn about new interests over time in movie recommendations. As part of this project, we are doing a comparative study between three algorithms used to make movie recommendations namely, **Matrix Factorization**, **Deep Learning**, and **Reinforcement Learning**. But first, let's talk about the two main approaches that are widely used for recommendation systems.

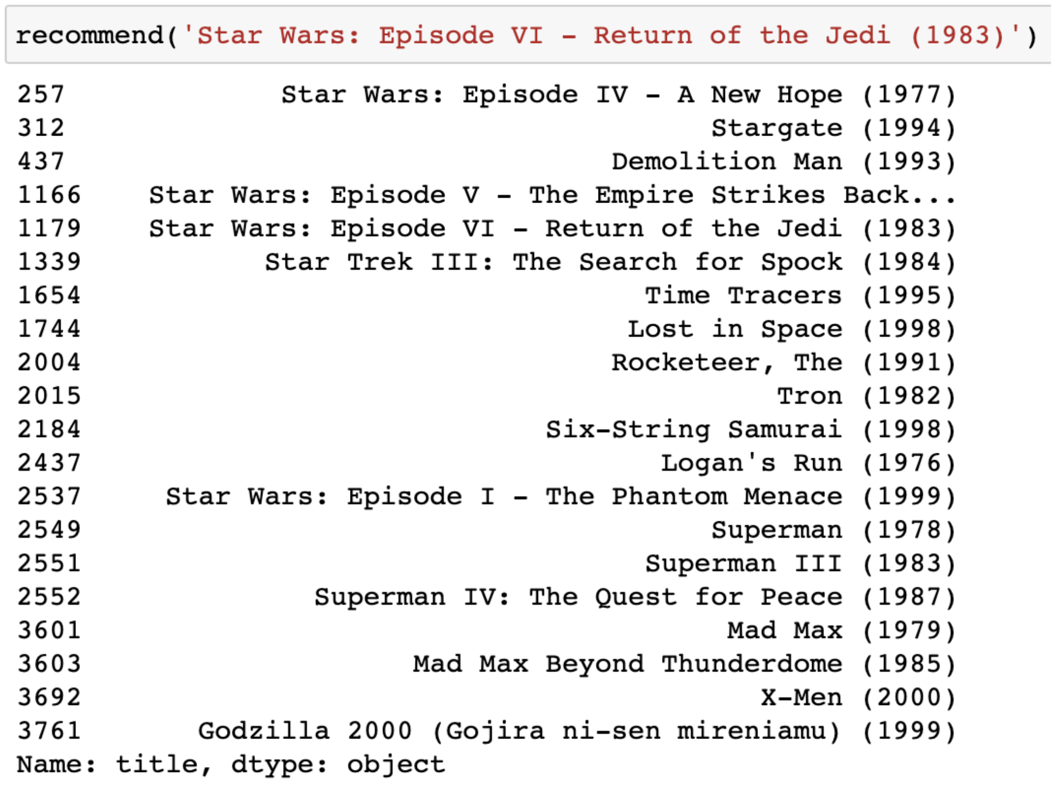
1.1 Collaborative Filtering

Collaborative filtering is a technique for predicting unknown preferences of people by using already known preferences from many users. It uses similarities between users and items simultaneously to provide recommendations. It uses the Cosine and Pearson correlation similarity approach. The main challenges with it are data sparsity, scalability, and cold start problem.

1.2 Content Based Filtering

Another common approach when designing recommender systems is content-based filtering. Content-based filtering methods are based on a description of the item and a profile of the user's preferences. This model has limited ability to expand on the users' existing interests and lacks novelty and diversity.

Demonstrating a simple movie recommender based on content-based filtering - tfidVectorizer which converts a collection of raw documents to a matrix of TF-IDF features [3], basically transforms text to feature vectors that can be used as input to the estimator. Given the movies as an input to our function **recommend**, it uses cosine similarly (a measure of similarity between two non-zero vectors of an inner product space that measures the cosine of the angle between them) to generate a list of movies as output. Suppose a user wants to watch a movie similar to 'Fight Club (1999)' then we can recommend other movies to the user by calculating the cosine similarity between Fight Club and those movies.



```
recommend('Star Wars: Episode VI - Return of the Jedi (1983)')
```

```
257             Star Wars: Episode IV - A New Hope (1977)
312                                       Stargate (1994)
437                                 Demolition Man (1993)
1166    Star Wars: Episode V - The Empire Strikes Back...
1179    Star Wars: Episode VI - Return of the Jedi (1983)
1339           Star Trek III: The Search for Spock (1984)
1654                                  Time Tracers (1995)
1744                                 Lost in Space (1998)
2004                                Rocketeer, The (1991)
2015                                          Tron (1982)
2184                               Six-String Samurai (1998)
2437                                    Logan's Run (1976)
2537     Star Wars: Episode I - The Phantom Menace (1999)
2549                                       Superman (1978)
2551                                   Superman III (1983)
2552            Superman IV: The Quest for Peace (1987)
3601                                        Mad Max (1979)
3603                   Mad Max Beyond Thunderdome (1985)
3692                                          X-Men (2000)
3761      Godzilla 2000 (Gojira ni-sen mireniamu) (1999)
Name: title, dtype: object
```

Figure 1: Working of recommend function

2 RELATED WORK

While we are covering a wide range of algorithms, we have not explored in depth many of the different architectures available under those algorithms. The paper: 'Reinforcement Learning based Recommender Systems: A Survey' [7] uses a Markov decision process (MDP) to formulate recommender systems and solves them using a Reinforcement Learning model and the paper: 'Knowledge-aware attentional neural network for review-based movie recommendation with explanations' [8] uses a knowledge-aware attentional neural network (KANN) for dealing with movie recommendation tasks by extracting knowledge entities from movie reviews and capturing understandable interactions between users and movies at the knowledge level. But none of these papers have compared the quality of recommendations provided by the algorithms on the same dataset for the same user which is what we are aiming to address through this project.

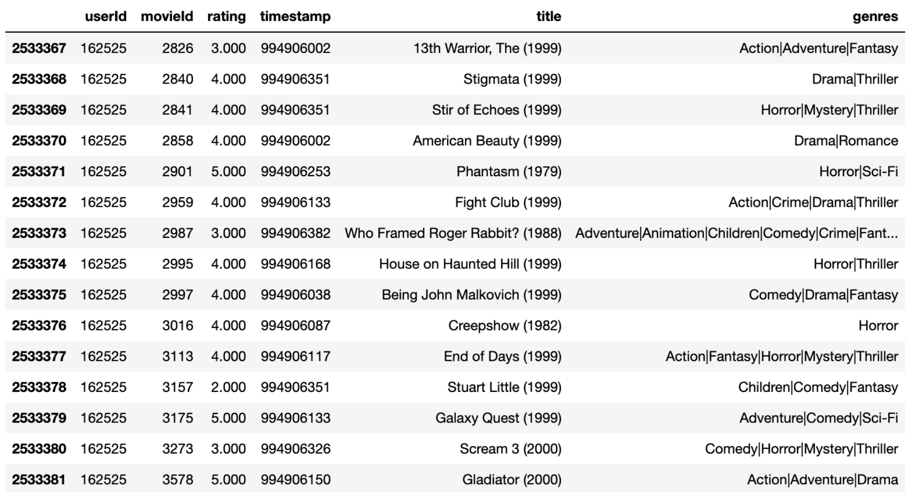
3 DATA COLLECTION AND PREPARATION

We are using a subset of the MovieLens 25M dataset which describes 5-star rating and free-text tagging activity from MovieLens, a movie recommendation service. The entire dataset contains 25000095 ratings and 1093360 tag applications across 62423 movies. This data was created by 162541 users between January 09, 1995, and November 21, 2019, and was generated on November 21, 2019. Users were selected at random for inclusion and all selected users had rated at least 20 movies. No demographic information is included; each user is represented by an id, and no other information is provided.

Exploratory Data Analysis -

- Data Files - The three CSV files that we are using are ratings.csv, tags.csv, and movies.csv. All ratings are contained in the file 'ratings.csv'. Each line of this file represents one rating of one movie by one user and has the following format: userId, movieId, rating, and timestamp. Ratings are made on a 5-star scale, with half-star increments (0.5 stars - 5.0 stars). Timestamps represent seconds since midnight Coordinated Universal Time (UTC) of January 1, 1970. All tags are contained in the file 'tags.csv'. Each line of this file represents one tag applied to one movie by one user and has the following format: userId, movieId, tag, and timestamp. Tags are user-generated metadata about movies. Each tag is typically a single word or short phrase. The meaning, value, and purpose of a particular tag are determined by each user. Movie information is contained in the file ‘movies.csv‘. Each line of this file represents one movie and has the following format: movieId, title, and genres. Movie titles are entered manually or imported from TMDB Movie website and include the year of release in parentheses. The year ranges from 1874 to 2019. Genres are a pipe-separated list and can contain more than one value.

- Sample of raw data -



	userId	movieId	rating	timestamp	title	genres
2533367	162525	2826	3.000	994906002	13th Warrior, The (1999)	Action|Adventure|Fantasy
2533368	162525	2840	4.000	994906351	Stigmata (1999)	Drama|Thriller
2533369	162525	2841	4.000	994906351	Stir of Echoes (1999)	Horror|Mystery|Thriller
2533370	162525	2858	4.000	994906002	American Beauty (1999)	Drama|Romance
2533371	162525	2901	5.000	994906253	Phantasm (1979)	Horror|Sci-Fi
2533372	162525	2959	4.000	994906133	Fight Club (1999)	Action|Crime|Drama|Thriller
2533373	162525	2987	3.000	994906382	Who Framed Roger Rabbit? (1988)	Adventure|Animation|Children|Comedy|Crime|Fant...
2533374	162525	2995	4.000	994906168	House on Haunted Hill (1999)	Horror|Thriller
2533375	162525	2997	4.000	994906038	Being John Malkovich (1999)	Comedy|Drama|Fantasy
2533376	162525	3016	4.000	994906087	Creepshow (1982)	Horror
2533377	162525	3113	4.000	994906117	End of Days (1999)	Action|Fantasy|Horror|Mystery|Thriller
2533378	162525	3157	2.000	994906351	Stuart Little (1999)	Children|Comedy|Fantasy
2533379	162525	3175	5.000	994906133	Galaxy Quest (1999)	Adventure|Comedy|Sci-Fi
2533380	162525	3273	3.000	994906326	Scream 3 (2000)	Comedy|Horror|Mystery|Thriller
2533381	162525	3578	5.000	994906150	Gladiator (2000)	Action|Adventure|Drama

Figure 2: Sample of combined data with ratings and movie titles

- Below is some basic statistics about our data:
  - No of Users who rated movies: 162541
  - No of Movies: 59047
  - No of ratings: 25000095
  - No of user comments: 73050
  - No of Movies commented by user: 45251
  - Percentage of user comments: 44.94%
  - No of movies commented by user: 76.63%
- From the basic statistics of the data we can say that the average movie rating is 3.53. The lowest rating a movie has received is 0.5 and the highest is 5.0. The total number of movies produced is over 200K while the overall number of users rating these movies is more than 160K.
- Upon inspecting all three CSV files for null values and unique values we can see that tags.csv had some null values which were removed using the ’dropna()’ function of pandas.
- Distribution graph of movie ratings - The count is in 1000000 values.

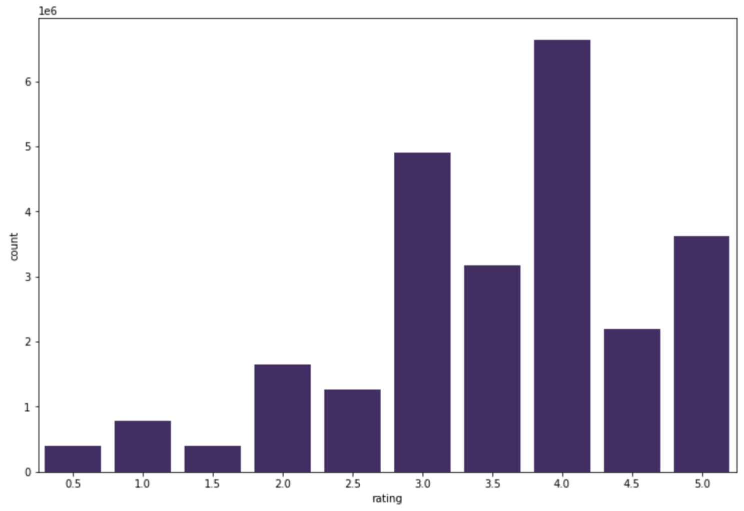
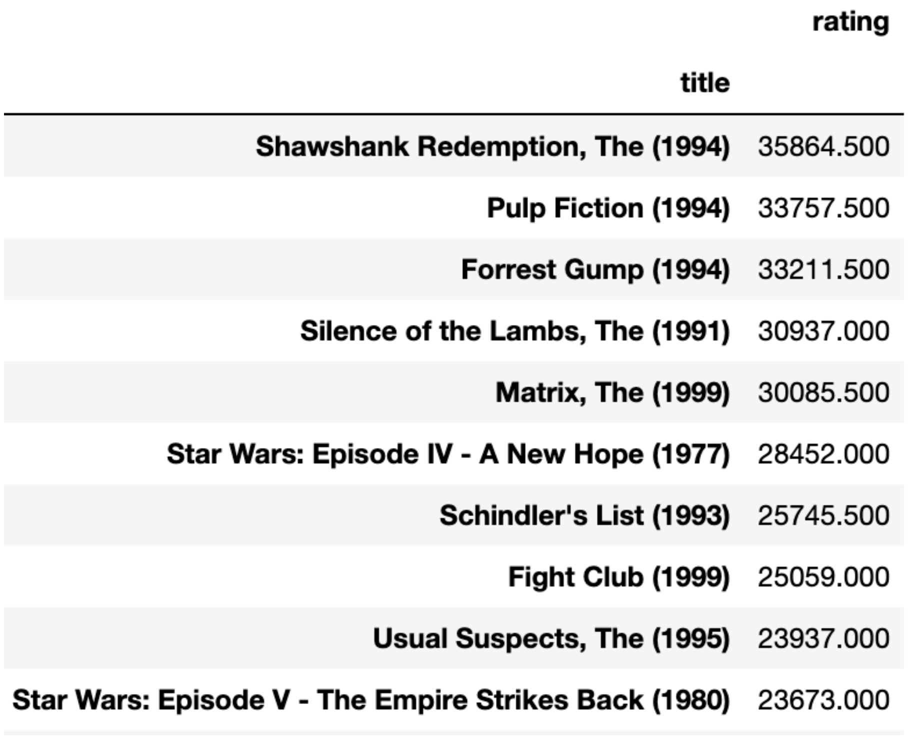


Figure 3: Distribution for movie ratings

- Top 10 movies according to highest ratings



title	rating
Shawshank Redemption, The (1994)	35864.500
Pulp Fiction (1994)	33757.500
Forrest Gump (1994)	33211.500
Silence of the Lambs, The (1991)	30937.000
Matrix, The (1999)	30085.500
Star Wars: Episode IV - A New Hope (1977)	28452.000
Schindler's List (1993)	25745.500
Fight Club (1999)	25059.000
Usual Suspects, The (1995)	23937.000
Star Wars: Episode V - The Empire Strikes Back (1980)	23673.000

Figure 4: Top 10 movies of all time

4 MODEL BUILDING

The models we are comparing are:

4.1 Matrix Factorization

4.1.1 Algorithm

The matrix factorization algorithm generates latent features when two different kinds of entities are multiplied. It takes a rating matrix R(U(Σ)VT) each representing the two factors - Users (People) and Items (Movies). The model will learn two matrices from the input - the user Embedding Matrix and Item Embedding Matrix. The algorithm decomposes such that one summarises Users and one summarises Items and the decomposed matrices will be the approximations of the Ratings Matrix. U represents how much users like each feature and V(T) represents how relevant each feature is to each movie. (Σ) is the diagonal matrix of singular weights. All these are put back together and we get good interferences for the missing values. The result can be generated when the math cost function RMSE is minimized through matrix factorization. In the real world, sometimes this algorithm gets a little complex. This is solved using SVD - Singular Value Decomposition of the matrix, one of the oldest and simplest form for Matrix Factorization. SVD represents data by removing the least important part. It takes simpler data by eliminating the least used. The model is built using the Ratings.csv and Movies.csv files and creates a User-Item Matrix. The model procures predictions using the trained SVD model and get predictions. Accuracy for SVD models is best determined by RMSE. We apply cross validation method in order to validate the prediction results. The output of the model is the prediction of movie ratings for given user and movie id combination.

4.1.2 Evaluation

Evaluating of model performance is done with RMSE and MAE with K=5 on 5 split(s). The RMSE before tuning the model was 0.89. After tuning, the RMSE was 0.90 .

4.1.3 Limitations

The performance of this model does not meet industry standards. With recent advances in deep learning, online users currently encounter recommendations trained with various types of (hybrid) neural networks (e.g. MLP, CN, RNN, etc.). Also, the lowest RMSE is achieved when the latent factors are less. But it is not probable that the user's taste (rating) is determined by such a low number of factors.

4.2 Deep Learning

4.2.1 Algorithm

The model uses an implicit feedback technique for training data which is a combination of movies.csv and ratings.csv. Train and test data is split based on the leave-one-out methodology. The expected output of the model is the next set of movies that the user will interact with. This number can be varied as an input. The model is built with the TensorFlow recommenders library which is based on Keras. The predict_movie function is taking inputs of user_id and the number of recommendations we want and gives as the output set of movies the inputted user_id should watch. This function is based on two retrieval models namely movie_model and user_model which are taking in raw features like user ratings, user ids, and movie titles as input and giving us user embeddings and movie embeddings as output. Embeddings make it easier to handle large data sets like in our case. Both the retrieval models use **keras.layers.embeddings** function for the conversion. Next, a network is set up by utilizing these embeddings and feeding it to the deep neural network. The network is learning these embeddings in such a way that it is trying to minimize the distance between them. The less the distance between the two, the higher the chances of the user liking the movie. The output of the network is predicted movie ratings for the user. For the network Keras with dense layer and **Rectified Linear Unit (ReLU)** as activation, a function is being used. Top predicted ratings are then summarized and displayed based on the number of movies requested in the input. This is done using the following function **tfrs.layers.factorized_top_k.BruteForce**. I have used ADAGrad optimizer in the model so that the learning rate and other hyperparameters of the neural nets do not have to be modified manually.

4.2.2 Evaluation

For this algorithm, the model is evaluated based on Root Mean Squared Error (RMSE). The model has been run twice, the first run includes 5% of the data set and the second run includes 10% of the dataset. The results are presented below.

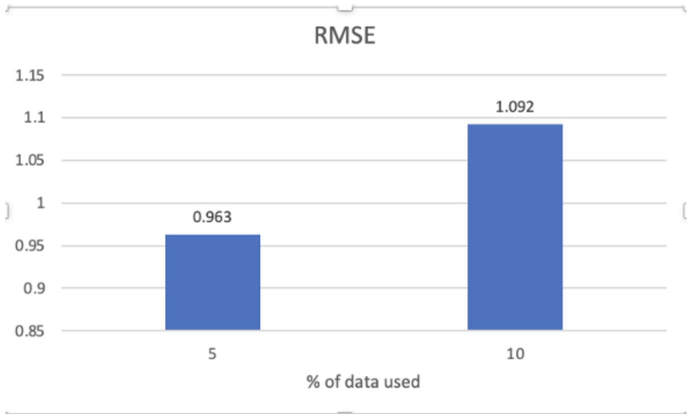


Figure 5: RMSE for Deep Learning

4.2.3 Limitations

The model does suffer from the cold start problem. If we have a new movie or a new user who has not rated many movies training becomes difficult and the prediction suffers. Also as the number of users grows, the algorithms suffer scalability issues. During the course of the project, we faced problems in handling the data set as it consists of 25 million records and our computational capabilities were not that massive.

4.3 Reinforcement Learning

4.3.1 Algorithm

The program consists of an actor-critic model used to generate an experience replay memory buffer for training the model [6]. The actor and critic models are built similarly and use a GRU so that the temporal relations between the ratings are taken into consideration. Initially, the movie ids are embedded using a context-target approach for mitigating the sparsity of one-hot encoded movie ids. This is then used to encode user rating history which is sorted based on timestep. The actor part takes in this encoded user's rating history as input and generates a movie (action) for the user to watch next. The critic part takes in the same user's rating history and the recommended action as inputs and determines the rating (reward) that the user would give to the movie. This data is encoded in <S(t), A(t), R(t+1), S(t+1)> format (S - states, A - action, and R - reward) and stored in the memory buffer. The memory buffer is then used for training the model and calculating the best parameters.

4.3.2 Evaluation

Since reinforcement learning considers the sequence of movies watched by a user, it would give the best possible recommendations. With our experimentation, we received squared difference loss values as low as 1.83 which indicates that the model is able to fit the reduced dataset almost perfectly.

Comparing the ratings of the recommended movies to the input provided to the model 6, we can see that they follow a similar trend in distribution. The model mostly recommends movies with high ratings which means that a user will good suggestions in most cases as long as they have movies in their history with sufficient ratings to indicate their preferences.

4.3.3 Limitations

While we have received near-perfect loss metrics with our current design, the model still lacks the robustness and precision required from a system fit for production deployment.

- Since the model has been trained with a subset of the original data, it would not have been able to derive the best interpretation from the ratings.
- We have not explored different hyperparameter combinations and that could have a positive and significant impact on our recommendations.
- We have not undertaken any measures to minimize the cold-start problem. This impacts our model as a new user or movie will find it difficult to be recommended a movie or to a user, respectively.

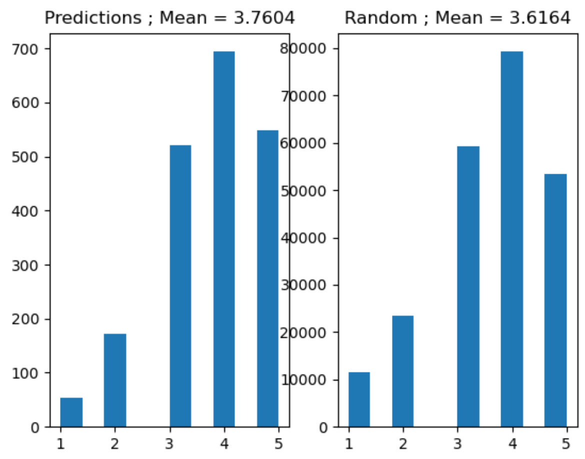


Figure 6: Prediction distribution of the RL model

- If we want to selectively look at the recommendations for a particular user, our model will not be useful in its current form as it is not as interactive as would be required of a live system.

5 CONCLUSION

All three algorithms are recommending movies to users with minimal error in our testing. The RMSE value for Matrix Factorization and Deep Neural Networks is almost similar around 0.9. Even for reinforcement learning, the predicted and actual values of recommendations are very close. All the models perform decently in our testing, the matrix factorization model would not perform well in the real world as it only considers basic similarities between users and movies. The neural network model would perform better as it would be able to extract meaningful relationships between the users and movies which could help the model predict even for users or movies with minimal ratings. The reinforcement learning model is expected to perform the best as it considers the sequence of ratings assigned by a user which is extremely useful in the real world. Since our data set was varied and these systems work on individual users' ratings it can be improved and other features can also be added.

6 FUTURE WORK

For future work, we can explore more hyperparameter combinations to improve the model performance. We can use the entire 25M rating data set for the best possible rating representation. This would require increasing our computational capabilities. Another idea that we could not explore due to time constraints is incorporating a matrix factorization model for embedding and utilizing deep neural networks for the actor-critic model in reinforcement learning. Essentially a combination of all three algorithms to better the recommendation system.

7 REFERENCES

Below are the websites and the papers that we have referred till the current stage of the project

[1] - Furtado, F. and Singh, A., 2020. Movie recommendation system using machine learning. International journal of research in industrial engineering, 9(1), pp.84-98.

[2] - Wu, L., Chen, C.H., Guo, K. and Wang, D., 2020. Deep learning techniques for community detection in social networks. IEEE Access, 8, pp.96016-96026.

[3] - `https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html`

[4] - `https://www.tensorflow.org/agents/api_docs/python/tf_agents/bandits/agents/lin_ucb_agent/LinearUCBAge`

[5] - `https://www.tensorflow.org/agents/api_docs/python/tf_agents/bandits/metrics/tf_metrics/RegretMetric`

[6] - `https://github.com/egipcy/LIRD`

[7] - M. Mehdi Afsar, Trafford Crump, Behrouz Far. 2018. Reinforcement Learning based Recommender Systems: A Survey. 1, 1 (June 2018), 37 pages. https://doi.org/10.1145/1122445.1122456

[8] - Liu, Y., Miyazaki, J. Knowledge-aware attentional neural network for review-based movie recommendation with explanations. Neural Comput & Applic (2022). https://doi.org/10.1007/s00521-022-07689-1

[9] - `https://www.kaggle.com/code/mfaaris/hybrid-and-tensorflow-recommender-system`

[10] - `https://www.tensorflow.org/recommenders/examples/basic_retrieval`

[11] - `https://www.tensorflow.org/recommenders/examples/basic_ranking`

[12] - `https://www.jiristodulka.com/post/recsys_cf/`

[13] - `https://towardsdatascience.com/recommendation-system-matrix-factorization-d61978660b4b`