Movie Recommendation System

Xinyu Jiang	Qinglu Sun	Qiuyang Wang
xiji6874@colorado.edu	qisu4484@colorado.edu	qiwa8995@colorado.edu
109036441	108641542	108741913
CSCI6502-001B	CSCI6502-001B	CSCI6502-001B

May 1, 2021

1. **Abstract**

The team aims to create a movie recommendation system uses Autoencoder based on collaborative filtering, and compare the precision, recall rate F1 score and average time cost of Autoencoder with the traditional models which use item based collaborative filtering and user based collaborative filtering.

2. **Introduction**

In modern days, watching movies play an essential role in the people's daily entertainment. However, with the development of the Internet, the cost for people finding a movie they want to watch is getting higher and higher. As an information demander, searching useful data from a sea of data is usually a time-consuming work. The recommendation system is an important tool to solve this problem. The recommendation system mines data from existing API, then enables information to be displayed in front of users who are interested in.

Due to that reason, we aim to design an algorithm that predict the evaluation of movies that users potentially interested in or movies that would highly appreciate based on users' browsing history.

3. **Related work**

Movie Recommendation engine

The scope of the project is very much similar to Movie Recommendation Engine which aims to recommend movies based on user input. The trending Movie Recommendation Engines on the market are using different algorithms to provide suggestions through a filtering process based on user preferences and other factors.

For instance, Netflix utilises a two-tiered two-based ranking system where ranking happens in each tow (strongest recommendations on the left) and across rows(strongest recommendations on top). Each row highlights a particular theme and is typically generated using one algorithm. The recommendation system developed by Netflix uses variety algorithms, Personalised Video Ranking, Top-N Video Ranker, Trending Now Ranker, etc.

There are 4 types of trending recommendation engines

(a) Content-Based

The Content-Based Recommender relies on the similarity of the items being recommended. The basic idea is that if you like an item, then you will also like a "similar" item. It generally works well when it's easy to determine the context/properties of each item.

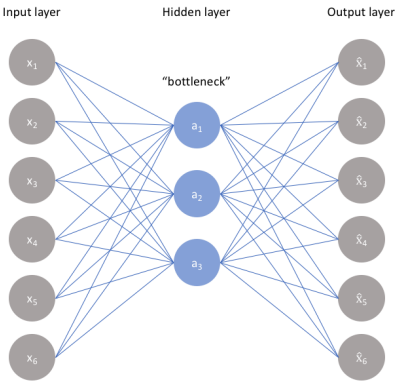
(b) Collaborative Filtering

The Collaborative Filtering Recommender is entirely based on the past behavior and not on the context. More specifically, it is based on the similarity in preferences, tastes and choices of two users. It analyses how similar the tastes of one user is to another and makes recommendations on the basis of that.

(c) Matrix Factorization
The goal of Matrix Factorization is to learn the latent preferences of users and the latent attributes of items from known ratings (learn features that describe the characteristics of ratings) to then predict the unknown ratings through the dot product of the latent features of users and items.

(d) Deep Learning
Lately, deep learning has demonstrated its effectiveness in coping with recommendation tasks. Due to its state-of-theart performances and high-quality recommendations, deep learning techniques have been gaining momentum in recommender system. Compared with traditional recommendation models, deep learning provides a better understanding of user's demands, item's characteristics and historical interactions between them.
The user latent features and movie latent features are looked up from the embedding matrices for specific movie-user combination.

(e) AutoEncoder
Autoencoders are an unsupervised learning technique in which we leverage neural networks for the task of representation learning. Autoencoder takes an unlabeled dataset and frames it as a supervised learning problem tasked with outputting x‘, a reconstruction of the original input x. This network can be trained by minimizing the reconstruction error, L(x,x‘), which measures the differences between the original input and the consequent reconstruction. The bottleneck is a key attribute of our network design; without the presence of an information bottleneck, the network could easily learn to simply memorize the input values by passing these values along through the network.



4. Proposed work

(a) Collaborative filtering
Collaborative filtering recommendation discovers user preferences by mining user historical preference data, groups users according to different preferences, and recommends products with similar preference. We shall build both user-based and item based Collaborative filtering recommendation. For user-based, calculate the relationship between users according to the degree of preference of different users for the same movie. Recommend movies among users with the same preferences. For item-based, it is quite identical to the user-based collaborative filtering algorithm, which exchange movies and users. The relationship between movies is gathered by calculating the ratings of different users for different movies. Recommend similar movies to users based on the relationship between movies.

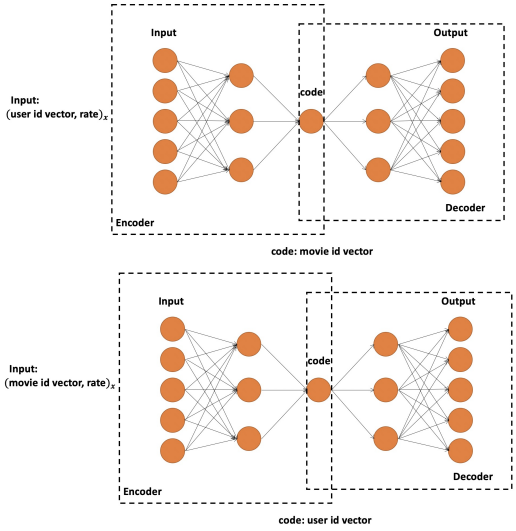
i. User-based CF
Calculate user similarity matrix and count the co-rated items between users using rating data and split it to training set and test set and build inverse table for item-users where the key is the movieID, and the value is the list of userIDs who have seen this movie, we saved the total movie number, which will be used in evaluation

ii. Item-based CF
Calculate movie similarity matrix and count co-rated users between items using rating data and split it to training set and test set.

(b) Auto-encoder based recommendation sys-

tem

i. Auto-encoders
Since the encoder only learns its typical features, and the features can be roughly restored to the original data by the decoder, we uses the co-occurrence matrix in collaborative filtering to complete the auto-encoding of the item vector and user vector. In this way, inputting the learned low-dimensional features into the corresponding neural network can greatly reduce the training time.
In our model, we uses information about user such as the user's gender and age, the year of the film, the specific year the user rated the movie such as which day of the year did the user rated the movie and what day of the week that user rated the movie. When we have IDs, they are ordered in the N-dimensional space of the users or the movies. Related movies/users which is based on user ratings are at a short distance of Euclidean metric. We also uses the vector user ID, vector movie ID, movie data, ratings data and user data for neural network for the prediction function. We also uses the region and time zone information, go through linear transformation and output to .NPY model file as the input of autoencoder. The first autoencoder is based on the assessment of user data using the values of user data of evaluation for this movie, ratings that user rated this movie, rates vector and vector of the user ID, for our first autoencoder, in one training cycle, all values in the autoencoder are given for a single product. For the second autoencoder which is trained on the data of movie rates, we use the values: the movie identifier vector, Information about the movie, which assessed the user, the rates data of the movie by the user, rates vector. In addition to that, the algorithm is defined as one user for different products, and the results are evaluated according to this user.



ii. Methods for receiving user ID and products ID vectors
After about 100 cycles of autoencoder training, we put products ratings by a single user as the second autoencoder. The second autoencoder uses its product ratings for the first autoencoder to balance the evaluation.
For example, a single user wrote M random products rates which is M different input vectors for autoencoder. At the same time, the algorithm receives M different values autoencoders, so that we receive M vectors of partial user IDs. Therefore, to correctly output the evaluation of the product, it is mandatory to design an autoencoder that would have a middle layer contains N elements which is equal to the size of user/products IDs. Then, the algorithm will calculate the average of those M values which will be the new values of user IDs vectors. After this process, we mark the result as L1 and obtain the average. Similarly, the algorithm will calculate L2 values based on users' feedback. By averaging and compressing L1 and l2 vectors, the algorithm is gradually changing the vector of user and products IDs to the target values vectors.

iii. Building Model
During training process, we are keep

records of training loss and verification loss, and we are saving the model every 50 samples trained, we also save movie id, user to movie score, weight and bias. We reiterate our training of neural network and autoencoders to approximate the user evaluation of the movie, we train the neural network for the prediction function on the data on the movie, ratings data, vector movie ID, vector user ID and user data to predict the unseen movie rating which can give the user movies under given conditions, for example if a user have 10 movies watched last week. The most basic collaborative recommendation system might recommend the same movies to him a day, which lacks novelty. By scoring those movies by the day of the week, we can recommend different categories of movies to the user on different day of the week. After the training, we have a model that consists of users, product IDs and neural network which correspond to two autoencoders and a neural network for rating prediction.

After learning process finished, we can predict the ratings that are not rated by users. In addition to that, to make the prediction more accurate we add functions that could add new ratings and correct users and products IDs vectors and add new products and new users. To calculate the vectors for new products and user IDs, we use autoencoder of the resulting model. In this case, we use the same procedure with generating vectors during the last training cycle expect that the algorithm does not teach autoencoders on the new data, but only calculates the identifiers. Therefore, the process calculating identifiers can be used when new elements come in so that we developers could adjust identifiers in the framework of pre-trained model. During the process of calculating/correcting identifiers, to adjust the processing speed, the algorithm does not correct the part of the model given by neural network, instead the algorithm would extend the models, so that the new changes would be illustrated in the vectors of users and products IDs. Thus, an accurate model would need additional training models through a certain period such as once a day, once a week, once a month and etc. This additional training can be done simultaneously with the prediction of system rates while adding new products, users, and user rates.

.

```
Numpy version : 1.14.3
Pandas version : 0.23.0
Theano version : 1.0.5
finding nearest ...
[0] 0.000000 -- 2319 : Reach the Rock (1997) -- Comedy
[1] 1.834755 -- 3147 : Green Mile, The (1999) -- Drama|Thriller
[2] 1.843280 -- 3753 : Patriot, The (2000) -- Action|Drama|War
[3] 1.848975 -- 3916 : Remember the Titans (2000) -- Drama
[4] 1.859276 -- 2332 : Belly (1998) -- Crime|Drama
[5] 1.862094 -- 875 : Nothing to Lose (1994) -- Drama
[6] 1.865473 -- 2571 : Matrix, The (1999) -- Action|Sci-Fi|Thriller
[7] 1.869167 -- 2502 : Office Space (1999) -- Comedy|Romance
[8] 1.878296 -- 3510 : Frequency (2000) -- Drama|Thriller
[9] 1.878901 -- 2762 : Sixth Sense, The (1999) -- Thriller
[10] 1.884768 -- 3617 : Road Trip (2000) -- Comedy
[11] 1.888285 -- 2706 : American Pie (1999) -- Comedy
[12] 1.888580 -- 628 : Primal Fear (1996) -- Drama|Thriller
[13] 1.893522 -- 3578 : Gladiator (2000) -- Action|Drama
[14] 1.895016 -- 3793 : X-Men (2000) -- Action|Sci-Fi
[15] 1.895984 -- 2329 : American History X (1998) -- Drama
[16] 1.909368 -- 2959 : Fight Club (1999) -- Drama
[17] 1.912501 -- 3052 : Dogma (1999) -- Comedy
[18] 1.913305 -- 2913 : Mating Habits of the Earthbound Human, The (1998) -- Comedy
[19] 1.920988 -- 3624 : Shanghai Noon (2000) -- Action
[20] 1.924326 -- 2841 : Stir of Echoes (1999) -- Thriller
[21] 1.929609 -- 2860 : Blue Streak (1999) -- Comedy
[22] 1.930199 -- 1485 : Liar Liar (1997) -- Comedy
[23] 1.931081 -- 3301 : Whole Nine Yards, The (2000) -- Comedy|Crime
[24] 1.932817 -- 428 : Bronx Tale, A (1993) -- Drama
[25] 1.933213 -- 223 : Clerks (1994) -- Comedy
```

Calculate the shortest distance of the movie

```
def main(load_id):
    consts = Consts()
    consts.load_from_ids = load_id
    rng = numpy.random.RandomState()
    theano_rng = RandomStreams(rng.randint(2 ** 30))
    user_lines = UserLines(rng = rng,theano_rng = theano_rng,consts = consts)
    rating_info = numpy.zeros(1,dtype=theano.config.floatX)
    wday = 4 # friday
    rating_info[0] = get_aranged(value = wday, min_value = 0, max_value = 6)
    #user_id = user_lines.rng.randint(low=0,high=user_lines.matrix_ids.users_count)
    #user_ids = user_lines._find_nearest(user_id,5)
    user_indices = [user_lines.rng.randint(low=0,high=len(user_lines.users_cvs)-1) for it in numpy.arange(5)]
    user_ids = [user_lines.users_cvs.at[indice,"id"] for indice in user_indices]
    #user_lines.build_rate_for_rand_user(rating_info = rating_info, user_ids = user_ids, consts = consts)
    user_lines.build_rate_for_rand_user(rating_info = rating_info, user_ids = user_ids, consts = consts)
    sys.stdout.write("all done\n")
    return
```

Recommend movies for users on Friday for user id 3970

5. Design and Implementation

- **Experiment Data Set**

  We used MovieLens datasets which provides 25 million movie ratings and one million tag applications applied to 62,000 movies by 162,000 users. Includes tag genome data with 15 million relevance scores across 1,129 tags. We also used IMDB datasets which contain various movie ratings and user profiles data.

  To verify the effectiveness of the model, we use the Movielens-1m real data set which is often used in recommending systems for experiments. The MovieLens data set is downloaded from the Group Lens website [1] and created by Nisota university collects from MovieLens website. The following introduces this data set in terms of its size

and data characteristics. As shown in Table 4-1, MovieLens-1m includes a total of 992,482 ratings for 3,544 movies from 6040 users. The MovieLens data set is an integer between 1 and 5. To show the user's favorite movie, a score of 1 indicates that the user does not like the movie, and a score of 5 indicates that the user enjoys the movie a lot. At the same time, it can be seen from Table 4-1 that the data set is very sparse (about 95 percent).

Data set	MovieLens-1m
Number of Users	6040
Number of Movies	3883
Number of Evaluation	1000209
Sparsity%	95.735

- **Data Pre-possessing**

  - To get accuracy testing result, we need to do data pre-processing on the data set. The MovieLens-1m data set mainly contains three categories: users data, movies data, and ratings data. Among them, users data is a user information file, which contains information such as "id", "sex", "age", "occupation", and "zipcode". For user data, we expand the zipcode information and added latitude, longitude, and time zone attribute. For movie data, we take the movie release year as a single attribute as an attribute Input for the recommendation system. Ratings data is a rating file, including user ID, movie ID, rating, and timestamp fields. We expand the timestamp attribute to weekday, day of year, and year of rating.
  - The experiment uses the ten-fold cross-validation method, that is, the data set is equally divided into ten disjoint subsets, and three of them are taken as the test set each time. The average of the five experimental results is used as the final experimental result.

- **Experiment Environment**
  The development language of this movie recommendation system is Python 3.8.0, the integrated development environment is Spyder of Anaconda, and the main deep learning framework is Theano version: 1.0.4. To ensure the fairness of the experimental results, the three models are all tested in the same experimental environment.

- **Autoencoder on user and movie data**

  We used 4 values from pre-processed user data to train our first autoencoder which is trained in the assessment of user data, they are:

  (a) vector of user IDs;
  (b) user data set evaluation on the movie(the vector of a data of the user);
  (c) rating the the user have on the movie(vector of the rating data);
  (d) rating itself(rating vector);

  We used 4 values from pre-processed movie data to train our second autoencoder model which is trained on the data of movie ratings, they are:

  (a) the movie identifier vector(vector of N elements);
  (b) information about the movie, which accessed by the user(movie data vector);
  (c) the ratings of the movie, rated by the user(movie data vector);
  (d) rating itself(rating vector);

- **Preparation products ratings for users**
  We ordered IDs in the N-dimensional space of the users and movies, and we related movies/users(based on user ratings) are at a short distance of Euclidean metric. Next, we used a neural network for the prediction function on:

  (a) the vector movie ID;
  (b) data on the movie;
  (c) other information(like day of week/time when the user rated/watched the movie);
  (d) vector user ID;
  (e) user data;

- **Building model**
  We reiterate our training of autoencoders and neural network to approximate the function by the user evaluation of the movie. There are some nuances of cold start training of autoencoders of random initial values.
  After training, we have a model consisting of users and movie IDs of neural network and two autoencoders for prediction of ratings.

- **Experiment and result analysis**

  We test User-Based Collaborative filtering algorithm, Item-Based Collaborative filtering algorithm, and collaborative filtering recommendation algorithm based on Autoencoder and DNN. The test results include the following:

  – Using the three collaborative algorithms above, we design a collaborative filtering recommendation system based on Autoencoder and DNN. Then we test three models on the MovieLens data set separately and compare the test results of the three algorithms through the evaluation criteria. The recommendation system calculates that the Presicion, Recall and F1 values based on Autoencoder and DNN which we measure all the accuracy results and found that the third algorithms yield higher accuracy than the first two.

  – We compare the time cost of the three algorithms by changing the number of recommended users and analyze it. It is concluded that the collaborative filtering recommendation algorithm based on Autoencoder and DNN has a good result compare to other algorithms in time-consuming perspective.

6. **Evaluation**
   The indicators for evaluating recommendation performance mainly include the following: user satisfaction, prediction accuracy. For the evaluation of accuracy, some of these indicators can only be obtained through user surveys, such as user satisfaction. In addition to that, some indicators are calculated through some quantitative calculations, such as prediction accuracy. These indicators can be used to evaluate whether a recommendation system can meet the needs of users. Among these indicators, the prediction accuracy is an evaluation indicator that is often used in recommendation algorithms, because this indicator can be calculated through offline experiments and can provide a quantitative evaluation for the recommendation system. In our experiment, we tested the User-Based Collaborative filtering algorithm, Item-Based Collaborative filtering algorithm and our model. We will take the average of Precision and Recall obtained by randomly recommending 10 user movies and calculate the F1 value from the average of the two. The specific experimental results are as follows:

	Precision	Recall	F1
User-CF	0.3716	0.0759	0.1260
Item-CF	0.2430	0.2448	0.2439
Our model	0.4080	0.2963	0.3433

It can be seen from the results that our model greatly improves the recall rate compared with the traditional collaborative filtering method under the same data set. And on this basis, the accuracy of the prediction is improved. In addition to that, this shows that the method of using Autoencoder and DNN is better than the traditional User-Based Collaborative filtering algorithm and Item-Based Collaborative filtering algorithm in the task of movie recommendation. Furthermore, we also tested the recommended time for users by the system. We set the number of movies recommended for each user to 50, and respectively calculate the average time required for users to recommend movies. The experimental results are as follows:

	Time
User-CF	5.3766ms
Item-CF	6.2430ms
Our model	0.7350ms

This shows that due to the sparse data of the data set, the effects of the traditional User-Based Collaborative filtering algorithm and Item-Based Collaborative filtering algorithm are greatly reduced, the time spend on training the model is too long. The Autoencoder method avoids the large-scale matrix calculation in the tradi-

tional coordinated filtering method. In addition to that, it greatly reduces the query time complexity, so that it is a good choice for the recommendation system.

7. **Discussion**
   There has been several changes on our project scope, in the start of the semester, we wanted to build a web application for recommending movies for registered users. And the initial algorithms we wanted to use for recommendation were content-based or collaborative filtering, since both of these two algorithms are very popular in current industry, we thought implementing the algorithms would be much easier and we can focus on web application building. But we changed our project scope after meeting with the professor, when she suggested us to implement more complex and challenging recommending algorithms. After few discussions, we had decided that to implement Autoencoder, a type of artificial neural network used to learn efficient data codings in an unsupervised manner, to our recommendation system. After few weeks of implementation and development, we found out that the algorithm was much more difficult than we thought. Therefore we changed our project scope again, from building up a complete web application, to only focus on the recommendation system itself, which is the most important part of the project. Implementing Autoencoder based on collaborative filtering for movie recommendation was extremely hard for us, after weeks of hard working, in the end of the semester, thanks to the help of the papers/lectures about Autoencoder online, we have finally developed our recommendation system which based on Autoencoder.
   The following are other discussions about the algorithms and datasets.

   - Filtering Algorithm
     In our initial project write up, we plan to use content-based or collaborative filtering for the recommender system. For now, we are using neural network-¿ autoencoder item based and user based collaborative filtering.
   - Data Selection
     In addition to that, in the initial project planning, we are going to collect data from OMDB API, then generate recommend movie list using the movie recommendation algorithm. For now, we are using data from pre-exist data, and generate recommend movie based on pre-exist data(We are now using those pre-exist data testing the recommendation code and we may switch back using data from OMDB API)

8. **Other related work**
   While we completed the movie recommendation system, we also implemented the following functions:

   - Completed the calculation of users with the same hobby, because people traditionally believe that users with similar characteristics will also like similar movies. In addition, it can also be implemented to recommend users with the same interest to users in the recommendation system.
   - The calculation of the movies that are closer in the Euclidean space. This allows us to intuitively analyze the relationship between each movie.
   - We cluster all the movies in the data set. The clustering is not based on the category given by the data set itself, but combines user similarity, user ratings of movies, and similarity between movies.

9. **Conclusions**
   In recent years, the rapid development of the Internet has allowed people to browse more and more information on the Internet, but there is very little information that attracts people's interest, so personalized recommendations are getting more and more attention. Therefore, the development of recommending systems has also been greatly promoted. The main tasks completed in this paper are as follows:

   - This project uses the classification of traditional recommendation algorithms, as well as several classic algorithm principles in recommendation algorithms. In this project, we mainly use the Auto-encoder pair to extract the auxiliary information of the movie to get great prediction result.
   - In this project, we design a personalized movie recommendation system based on the incorporation of auto-encoder and DNN. In

addition to recommending movies for users, it also include functions of recommending users, calculating similar movies, and clustering.

10. **Limitations**
    Throughout the project, we get good result in prediction accuracy and time spending, but there are still some limitations, those following questions can be further studied in the follow-up.

    - In this model, only the MovieLens-1m data set is used for testing, and no more data sets are tested, which makes the model effect unconvincing.
    - The training and prediction of the model are all performed offline, and the real-time performance of movie recommendation is not considered. Thus, it is impossible to make effective recommendations for user changes in time. Therefore, in the follow-up research, real-time performance will be taken into consideration

11. **Future Integration**
    Just like what we planned in the start semester when we wanted to build a web application for movie recommendation. For future integration, we are going to build a web application based on the recommendation system we have developed. Currently, we have finished some of the back-end works, and minor frontend works. In the future, a user can login/register to add his/her watched movies, and our recommendation system will generate recommend movies based on these input. Also, users can view each others' profiles which contains the movies the user have watched and the ratingsreviews.

12. **Milestones**
    We are looking for achieving the following of the milestones we set out at the beginning of the project:

    - **Milestone 1:** Finalizing the topic of the project. In addition to that, we set up group meeting time, venison control software(Github).
    - **Milestone 2:** Instead of building up a movie website with complete features to visualize our recommender, which was what we planned in the beginning of the project, we are considering of only finishing the recommendation system itself, since we are mainly focusing on recommender algorithms, the remaining efforts on building up other parts of the project will be minimized. Therefore, we decided to only focus on recommendation system and its algorithms.
    - **Milestone 3:** Pre-processing the data. Obtain the datasets and convert the csv files into npm files which can be read by the encoder.
    - **Milestone 4:** Train Autoencoder with the pre-processed data.
    - **Final Milestone:** Reiterate training of Autoencoder and neural network to simulate user evaluation of movies.

13. **Appendix**

    - **Honor code**

      Honor Code Pledge
      On my honor, as a University of Colorado Boulder student,
      I have neither given nor received unauthorized assistance.

    - **Work distribution**
      - **Qiuyang Wang**:
        * Essay Writing
        * Algorithm Comparison
        * Presentation Writing
      - **Qinglu Sun**:
        * Auto-encoder Implementation
        * Data set Pre-processing
        * Experiment Result Analysis
      - **Xinyu Jiang**:
        * Essay Writing
        * Model Evaluation
        * Presentation Writing

References

1. Chong, D. (2020, May 2). Deep Dive into Netflix's Recommender System - Towards Data Science. Medium. https://towardsdatascience.com/deep-dive-into-netflixs-recommender-system-341806ae3b48
2. Chong, D. (2020, May 2). Deep Dive into Netflix's Recommender System - Towards Data Science. Medium. https://towardsdatascience.com/the-4-recommendation-engines-that-can-predict-your-movie-tastes-109dc4e10c52
3. Jordan, J. (2018, March 19). Introduction to autoencoders. Jeremy Jordan. https://www.jeremyjordan.me/autoencoders/
4. Hrishekesh Shinde 2018 Deep Autoencoders for movie recommendations — A practical approach. https://medium.com/@shrishekesh/deep-autoencoders-for-movie-recommendations-a-practical-approach-ae539f3473e4
5. Jeffrey Lund, Yiu-Kai Ng, Movie Recommendations Using the Deep Learning Approach https://students.cs.byu.edu/~ng/papers/IRI2018-75.pdf
6. Dr.Pardeep Kumar, MOVIE RECOMMENDATION SYSTEM USING AUTOENCODERS http://www.ir.juit.ac.in:8080/jspui/bitstream/123456789/22814/1/MOVIE%20RECOMMENDATION%20SYSTEM%20USING%20AUTOENCODERS.pdf
7. Julio C´esar Barbieri Gonzalez de Almeida, WHEN AUTOENCODERS MEET RECOMMENDER SYSTEMS: COFILS APPROACH https://www.cos.ufrj.br/uploadfile/publicacao/2754.pdf
8. Vaddy, S. (2020, May 10). User based Collaborative Filtering Movie Recommendation. Medium. https://medium.com/@srinidhi14vaddy/collaborative-filtering-movie-recommendation-60461c7ef897
9. Abhinav Ajitsaria. (2021, April 24). Build a Recommendation Engine With Collaborative Filtering. Real Python. https://realpython.com/build-recommendation-engine-collaborative-filtering
10. Prem Kumar,Netflix Movie Recommendation — Using Collaborative Filtering https://towardsdatascience.com/tensorflow-for-recommendation-model-part-1-19f6b6dc207d
11. Google developers, Collaborative Filtering https://developers.google.com/machine-learning/recommendation/collaborative/basics
12. Siddhartha Banerjee, Collaborative Filtering for Movie Recommendations https://keras.io/examples/structured_data/collaborative_filtering_movielens
13. Ramya Vidiyala, How to Build a Movie Recommendation System https://towardsdatascience.com/how-to-build-a-movie-recommendation-system-67e321339109
14. So Ham,Create Your Own Movie Movie Recommendation System https://www.analyticsvidhya.com/blog/2020/11/create-your-own-movie-movie-recommendation-system/
15. Rounak Banik,Movie Recommender Systems https://www.kaggle.com/rounakbanik/movie-recommender-systems
16. Saniya Parveez, Roberto Iriondo,2020 Oct, Recommendation System Tutorial with Python using Collaborative Filtering https://pub.towardsai.net/recommendation-system-in-depth-tutorial-with-python-for-netflix-using-collaborative-filtersk=802d36f89bb9a2827eb370eb3b3675a1&source=friends_link&gi=80ea27b4ba4d
17. Yogesh Jhamb, Travis Ebesu, Yi Fang, 2018, Attentive Contextual Denoising Autoencoder for Recommendation https://dl.acm.org/doi/abs/10.1145/3234944.3234956