Int. j. inf. tecnol. (June 2022) 14(4):2067–2077
https://doi.org/10.1007/s41870-022-00858-4

ORIGINAL RESEARCH

Neural model based collaborative filtering for movie recommendation system

Kalyan Kumar Jena[1] · Sourav Kumar Bhoi[1] · Chittaranjan Mallick[2] · Soumya Ranjan Jena[1] · Raghvendra Kumar[3] · Hoang Viet Long[4] · Nguyen Thi Kim Son[5]

Received: 20 July 2021 / Accepted: 31 December 2021 / Published online: 4 February 2022

Abstract Due to the availability of enormous number of products of same domain is increasing day by day the possibility of getting your desire product is getting less. Therefore, a recommendation not only helps you find the best probable product according to your preference but also increases the efficacy to find the product for you in less time interval. Artificial neural network has been producing a tremendous result in the practical solutions like image classification, speech recognitions and various AI problems. The use of neural network to build recommendations system can also be used as an auto encoder in various sector. The neural network contains many layers and each layer contains many perceptron which holds the weight. While the network gets trained, the weights of each perception are optimized and get adjusted. Building a simple neural network model for predicting recommendations with high accuracy is the objective of this work. The dataset used in this recommendations model is contributed by the Movie-lens archive. Manipulating the data into a right form and format is the most important part of the model. The whole work is performed, experimented and evaluated in python as it consists of many predefined useful libraries. The result of the recommender model is evaluated by finding the Hit-Ratio. The Hit-ratio obtained by this model is 87.

Keywords Neural-network · Matrix factorization · Latent space · Implicit data, Movie predictions

1 Introduction

Recommender systems are the systems that recommend a piece of information to the user as per the demand. A movie recommender system provides clues to the consumers by way of filtering procedure i.e., based on consumer desire and surfing history [1–15]. E.g., Netflix recommendation system supplies the user with recommendations of the movies that are related to the ones that

✉ Nguyen Thi Kim Son
ntkson@daihocthudo.edu.vn

Kalyan Kumar Jena
kalyan.cse@pmec.ac.in

Sourav Kumar Bhoi
sourav.cse@pmec.ac.in

Chittaranjan Mallick
cmallick75@gmail.com

Soumya Ranjan Jena
s_rn_j@yahoo.com

Raghvendra Kumar
raghvendraagrawal7@gmail.com

Hoang Viet Long
longhv08@gmail.com

[1] Department of Computer Science and Engineering, Parala Maharaja Engineering College (Govt.), Berhampur, India

[2] Department of Basic Science (Mathematics), Parala Maharaja Engineering College (Govt.), Berhampur, India

[3] Department of Computer Science and Engineering, GIET University, Gunupur, India

[4] Faculty of Information Technology, University of Technology-Logistic of Public Security, Hanoi, Vietnam

[5] Faculty of Natural Sciences, Hanoi Metropolitan University, Hanoi, Vietnam

have been viewed in the past. So, the recommendation system [16–31] can be broadened into two aspects of filtering the first one is the classical content-based recommendation which is a recommendation to the user from the context of the data type, origin, creator, genre, etc. and the second aspect is the collaborative filtering that uses the user-item database to find the similarity between two or more user and recommends them with the data of others. ANN is a smaller replica of the human brain which contains neurons corresponding to one another. So precisely we can say the human brain does not follow an algorithm to do something, for example, recognizing a number 5 but the human brain is not a procedure rather it is training. So, the NN also gets trained with the help of hyperparameters to adjust the weight of the node (called a neuron). Therefore, taking the help of NN to perform filtering for recommendations not only will give the degree of artificial intelligence but also will give a higher efficacy to the recommendations. Besides all this, data is of 2 types according to the recommendation system such as explicit and implicit. Explicit data means the data that is retrieved through the user consent or in layman words directly for example a user purchase a product, rates a movie, or gives a satisfaction value such as thumbs up or down to a stake. Whereas implicit data is the data that is retrieved without the consent of the user or indirectly data for example a user watched a movie trailer or read an article, from this we can get an affirmation of intention but not clearly. We might know explicit data is always valuable but as compared to implicit data the explicit data is much difficult to collect. As we have collected a set of explicit data, therefore, we perform data preprocessing to convert it into implicit data. So, the core objectives of this model are:

1. In this work, a simple NN model is built for predicting movie recommendations with high accuracy.
2. The dataset used in this recommendation model is contributed by the Movie-lens archive [22]. The explicit data is converted into implicit data using various methods and data is split into training and testing sets.
3. The whole work is performed, experimented and evaluated in Python as it consists of many predefined useful libraries.
4. The result of the proposed recommender model is evaluated by finding the Hit-Ratio. The Hit-ratio obtained by this model is 0.87 which is more than other models such as GMF, MLP, NeuMF, and SDAE [26, 27].

The rest of the work is presented as follows. Section 2 presents the literature survey on the related works done by many researchers in this area. Section 3 discusses the dataset. Section 4 presents the proposed model. Section 5 presents the evaluation of the model using implementations. Section 6, at last, presents the conclusion and future scope of the work.

Table 1 Loss obtained while training the neural model

Epochs-Eph	Loss
Eph-1	0.219
Eph-2	0.202
Eph-3	0.187
Eph-4	0.181
Eph-5	0.180

2 Related works

Many research works have been done in the area of collaborative filtering for movie recommendations. Some recent methods are discussed as follows.

Neural-Collaborative-Filtering (NCF) model gives a basic architecture to represent matrix factorization in form of deep learning methods. It uses the latent space to find and extract the features to predict the user-item utility matrix. This method uses the embedding layer to convert the user vector and item vector to a latent space containing lower dimension data. Thus, training the layer to achieve an efficient network [1]. In an NCF Model with an Interaction-based Neighborhood [2], by using user-item interaction database they proposed deep learning models which successfully suggest a novel neighbourhood. In this model first time neighbourhood information which is implicit data is used in a neural collaborative filter to give information about the characteristic behaviour of a good neighbour. A new deep hybrid recommender system based on auto-encoder with NCF [3] has been proposed by the researchers who used implicit feedback rather than using explicit feedback of user, which is not easily available in a user interface. They proposed a hybrid recommender system framework that uses auto-encoders by combining user and item side databases. This dataset is based on a real-world dataset which gives better performance than the state-of-the-art method.

Joint NCF for Recommender Systems [4] is proposed by the researchers in which the model uses a joint NN that combines deep feature learning and deep interaction modelling based on a user-item matrix. J-NCF enables training that improve recommendation volatile performance. J-NCF gives better results than state-of-the-art methods, which is about 8.24%. When the data set of Amazon movies is used as input in this model the model shows improvement 12.42%, 14.42% and 15.06%, respectively. Learning binary codes with NCF for efficient recommendation systems [5] is proposed by the researchers

in reducing the storage requirement in the e-commerce market by using binary codes and collaborative filtering. In this model, by using NNs and binary codes the quantization loss is minimized. This model is highly effective in dealing with new users, new ratings and new items.

Context-Aware Quality of Service (QoS) Prediction with NCF for Internet of Things (IoT) Services [6] model is based on fuzzy clustering and neural collaborative filtering. By using a fuzzy clustering algorithm, this model is capable of providing clustering contextual information. This model is implemented on two real-world datasets and gives effective results. An Adversarial Approach to Improve Long-Tail Performance in NCF [7] is proposed to find a new adversarial training strategy to use in long-tail recommendations for the user. This model is highly useful to avoid the adversarial penalty and shows 25% effective than previous any other model. Predicting yield performance of parents in plant breeding [8] model uses neural collaborative filtering such as deep factorization machines (Deep FM), GMF to predict the best hybrid corn for a different place which give more growth rate in agriculture. This model was widely accepted by researchers. Commercial Site Recommendation Based on NCF [9] model uses neural collaborative filtering and NeuMF-RS method to propose product to the customer based on the commercial data, geographic data and also other heterogeneous data. This model works both on implicit and explicit data of costumer to predict the new product for the customer. This model shows more effective results than any other method in commercial site recommendation. An NCFM method for identifying miRNA disease [10] is proposed to predict miRNA disease. By using a sparse vector of disease and dense vector of miRNA the latent feature of interactions between miRNA and diseases formed. By this method, 90% prediction of miRNA was done successfully.

Grade Prediction with Neural Collaborative Filtering [11] model is used to predict which course should a student choose in his\her next term. In this rather than NCF, Matrix factorization (MF) is used to predict the best course for a student based upon latent knowledge space data. DC-RNFCF [12] is also called Debiasing Career Recommendations with Neural Fair Collaborative Filtering uses the social media data by using Neural Fair *Collaborative* filtering (NFCF) to recommend the user about their career and courses related items by using pre-training and fine-tuning approach. Neural Collaborative for Music Recommendation System [13], by using CF and NCF approach this model is taking 20,000 users, 6000 songs 470,000 transaction ratings. In music, there are many types of genres and every user has different types of music preferences. The CF approach for this model is still struggling so by the help of hybrid technique to get better suggestions as per the user previous record.

Deep Learning Architecture for Collaborative Filtering Recommender Systems [14] model uses deep learning architecture to get the non-linear connection between accurate recommendation, prediction and rehabilitees. Software Service Recommendation Based on Collaborative Filtering [15] model is used to fulfil the requirement of the user to get better web service suggestions through this model. This model uses the non-linear repository to get the abstract data of vectors. A Hybrid Approach for Neural Collaborative Filtering [16] model personalizes to get better performance for e-commerce and online entertainment through matrix factorization technique. The model takes the dataset from Movie lens, Pinterest and Yelp. By integration of deep learning with collaborative filtering to get an accurate result, as per the user input or user action the recommendation engine gets results accurately. HNCR [17] Hyperbolic Neural Collaborative Recommender is divided into two types: neighbour construction and recommendation framework. In the first part according to the user and item interaction history, it constructs a semantic neighbour relationship. The second part by using hyperbolic geometry to combine neighbour set into a recommendation. CF-NADE [18] model computes the equality between the use of data and the item data from the rating data. The CF method is based on a model-based technique that gets a high prediction output matrix of low rank. Dual-embedding based Neural Collaborative Filtering for Recommender Systems [19] utilize historical interaction between the user and item, to get which movie is similar to each other about the user interest based on the other user's similar interest. By embedding the items into low-dimensional space generate to store similar movies nearby.

Product recommender system using neural collaborative filtering for marketplace [20] model takes data from user explicit feedback and by rating on the product. Based on the previous orders and interaction of the product with the user it predicts a similar product as per user search. This model is using CF recommendations based on the past action of the user. Trust-Based Neural Collaborative Filtering [21] is using a deep NN model to know the interaction between the user and item. The interaction between user and item is merging through the Generalized Matrix Factorization (GMF) model to get the trust friend interest.

3 Dataset

The dataset to this model is retrieved from Movie-lens archive which contains 20 million data [22]. Mainly the dataset holds many sub-datasets like the data of movie genome, genres, movie ratings given by different users, movie links and tags. The subset of data we are using for this model is the movie rating dataset. It is considered to be

an explicit dataset as the user has given the ratings directly through any feedback channel.

4 Proposed model

The dataset of rating data holds the head containing movie ID, user ID, rating, timestamp as shown in Fig. 1. As elaborated above the data imported is explicit data so converting the data to implicit is the first step of the model. Converting the explicit data to implicit data is performed by converting the ratings to '1', this means if a user had rated a movie that means the user must have seen that movie. Thus, the data in the dataset is assumed to be a user-item interaction database.

After converting the data to the desired form, the second step of the model is to split the data into a training set and testing set. If a user has interacted with N i.e., (1,2, 3,…,N) number of movies then the testing data is defined to be the Nth movie whereas the movie behind the Nth movie i.e. (1,2,3,…,N − 1) is determined to be the training dataset, the following things can be visualized as shown in Fig. 2.

Converting the dataset to an implicit dataset enforce the dataset to be a positive class which means the dataset only contains the data of the user that has interacted with the movie. Thus, training a NN with the positive class will prevent good efficacy. So, the third phase of the model is finding four negative class data for each user. The four negative class data can be defined as the movie that is unseen by the user-chosen randomly from the dataset. The preprocessing of data is now said to be defined for feeding the NN. In the backend, the data preprocessing has obtained a user-item matrix that contains user vectors and item vectors. So, in the classical matrix factorization method the user-item utility matrix is factorized into two embedded matrix and while the matrix is again multiplied the error function is evaluated and thus with the help of gradient descent the error function is minimized. The Matrix factorization method can be defined by the formula given below [23]:

$$\hat{y} = U.I^T \tag{1}$$

where, U is the User feature Matrix, I^T is the Transpose of Item feature Matrix and $\hat{y}$ is the Predicted matrix. The matrix factorization method can also be implemented in the NN the most used NN is known as neural collaborative filtering. The model is defined as a series of fully connected layers, and embedding layer and an output layer. Precisely there are 8 layers out of which one is the embedding layer for item vector and user vector. The embedding layers are concatenated by dot product thus it produces a user-item utility matrix. Now the NN model uses hyperparameters to

Fig. 1 Visualization of Dataset

	userId	movieId	rating	timestamp
236	3	1	4.0	1999-12-11 13:36:47
237	3	24	3.0	1999-12-14 12:54:08
238	3	32	4.0	1999-12-11 13:14:07
239	3	50	5.0	1999-12-11 13:13:38
240	3	160	3.0	1999-12-14 12:54:08
...	...	...	...	...
20000258	138493	68954	4.5	2009-11-13 15:42:00
20000259	138493	69526	4.5	2009-12-03 18:31:48
20000260	138493	69644	3.0	2009-12-07 18:10:57
20000261	138493	70286	5.0	2009-11-13 15:42:24
20000262	138493	71619	2.5	2009-10-17 20:25:36

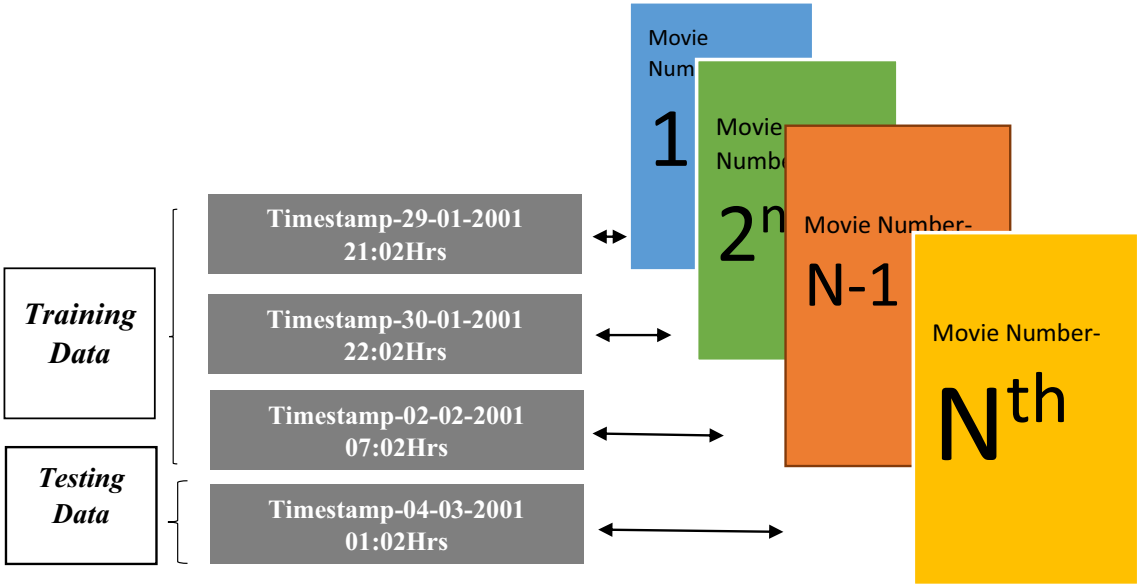


Fig. 2 Data splitting for any user

predict a user-item matrix by adjusting the weights of the node in the fully connected layer. The NN model is trained until the error function is minimized and thus, we get a predicted user-item utility matrix. Embedding can be defined as a lower-dimensional space representation of higher dimensional space data. The visualization of the model is shown in Fig. 3. Embedding in this model is set to 8 which mean the user/item vector is converted to 8 dimensions or the embedded vector contains 8 entities. The activation function used in the dense layer is RELU and it can be defined as [24]:

$$F(\alpha) = \max(0, \alpha) \tag{2}$$

Returns α for all values where $\alpha > 0$, else returns 0.

At last, the output is determined by the SoftMax function as gives a probabilistic approach to predict the output vector. The SoftMax function can be determined as [25]:

$$f_j(z) = \frac{e^{z_j}}{\sum_{k=0} e^{z_k}} \tag{3}$$

Considering, $f_j(z)/\sigma$ is the output of SoftMax, z is the input vectors, e^{z}_{j} standard exponential function for input vectors, K = number of classes in the multi-class classifier, e^{z}_{K} standard exponential function for output vector. The model defined replicates the matrix factorization method because the result in matrix factorization is equal to the result in the neural method. As described above the embedding helps to deal with lower-dimensional space but it also produces a latent vector which gives an ease to train the NN to predict the output. Testing the model by taking 99 negative class data with the testing data and then ranking the movies to see whether the model has predicted the movie or not. If the testing data is below the rank 10 thus it is a Hit. The recommender system can be evaluated by the use of Hit-ratio the hit ratio can be defined as the ratio between the total numbers of hits with respect to the number of events [26]:

$$\text{Hit} - \text{ratio} = \frac{\text{Total number of hits}}{\text{Total number of hit and miss}} \tag{4}$$

Algorithm 1 below represents the whole process as discussed above. The watched movies are converted to the ratings as 1. That means if a user has rated a movie he has already seen that movie. In line 1 it is fetching the database for each database entry. If ratings = = 1 that means a user has watched this movie. Lines 3–8 sorts the ratings according to their time stamp to know which movie the individual users are seeing first. In line 9 we get the test data for each user. In line 10, random data for each dataset of individual users are calculated. In line 11, by putting the total dataset we sorted the timestamp to get training data and evaluated the NN by the 4 random functions which is our test data. At last, lines 14–17 finds the hit ratio for evaluation of the models.

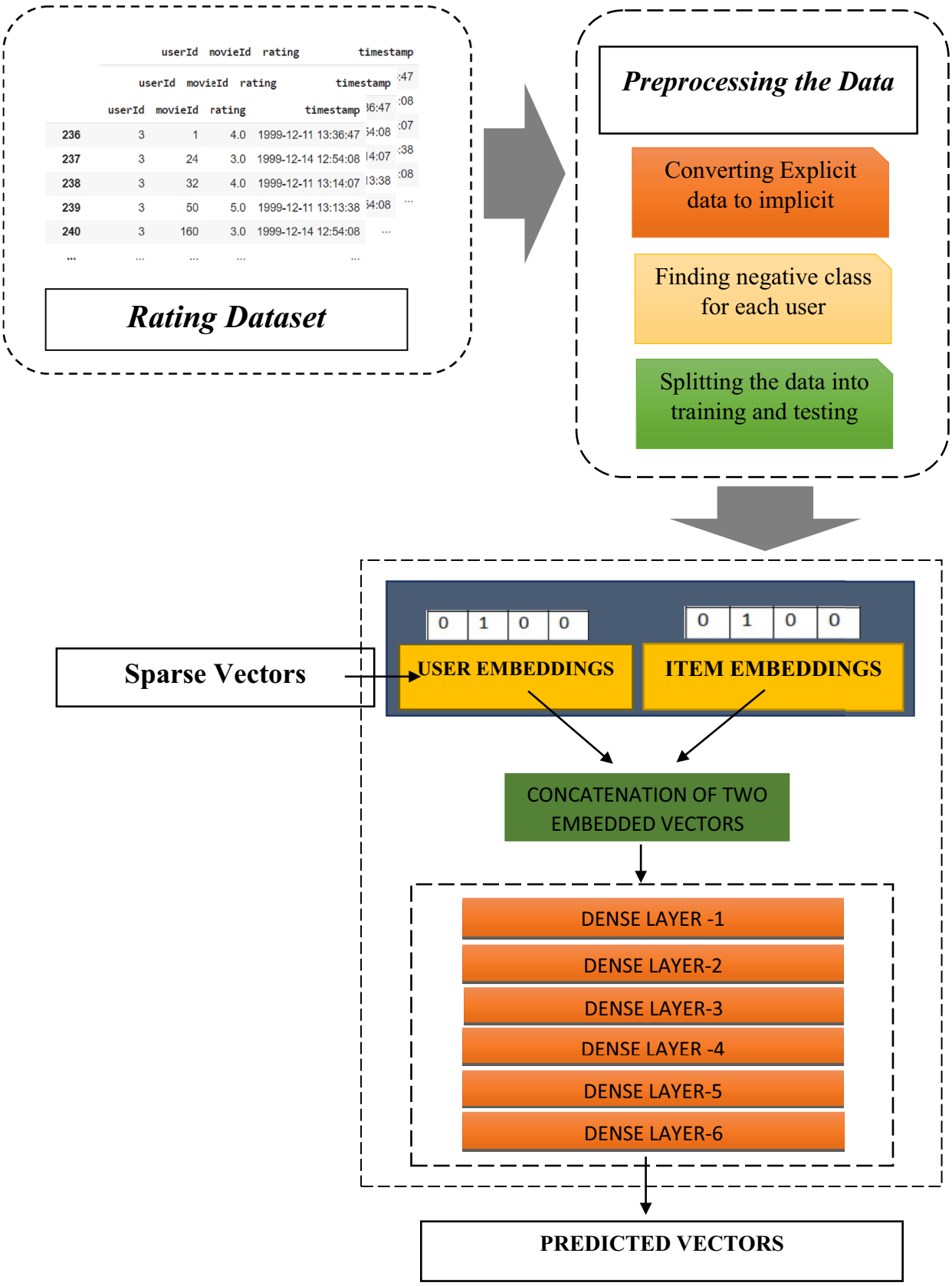


Fig. 3 Work flow of the model

Algorithm 1: Prediction of Movie using Recommendation Model

Input: Movie Ratings Dataset Containing (Movie Id, User Id, Ratings of movie given by Individual Users, Timestamp)
Output: Predicting Movie for Recommendation

```
for each entry in dataset
    if (Ratings ==1)
        for each user entries
            If (user_entry_timestamp[1]<user_entry_timestamp[each_entry])
                user_entry[1]==user_entry[n];
                {
                SORT user_entry_timestamp BY timestamp ASCENDING.
                }
                test_data = user_entry[1];
for each user
    find random 4 negative class data;
Training the neural model with the training data;
Predicted_vectors();
for each user
    Test=find random 99 negative class + test_data;
Predicted_vectors(Test);
prediction=rank(test)
```

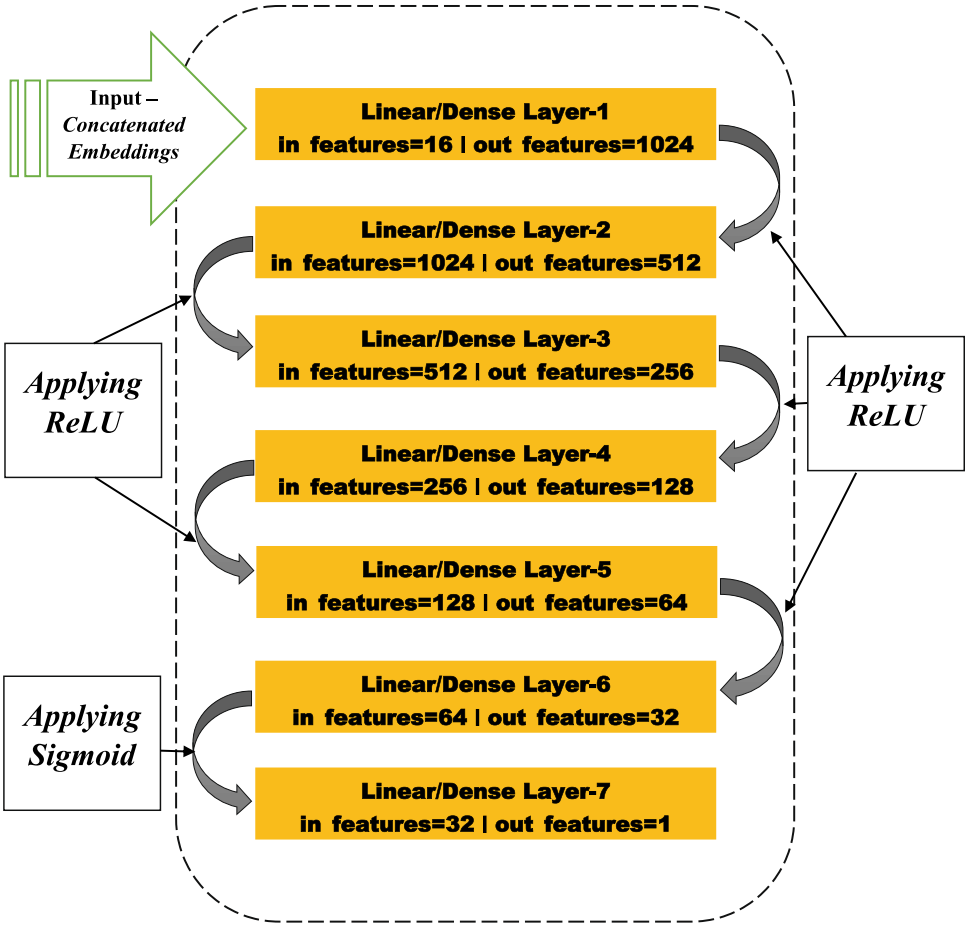
5 Experimentation and evaluation

The NN is implemented on python with the help of PyTorch library as it gives ease to implement the NN models. The pytorch_lightning library contains many predefined layers and activation functions to assimilate the NN. System requirements like GPU of 2 GB (NVIDIA) enhance the speed of training on a 64-Bit Windows Platform. The system specification of the model uses a RAM of 16 GB and processor Intel i5 10th Gen (4 physical core). Hyperparameters like batch size and the number of epochs are set to 512 and 5 respectively. The NN for the model is visualized in Fig. 4. The user vector and item vector are the outcomes of the user-item interaction and they are delivered to the embedding layer to get an output called latent vectors. The latent vector contains a probabilistic approach to determine the degree of an entity in an individual vector also considering the vector preferences. Concatenating the user and item vector and delivering the embeddings to a NN helps us to predict whether the user has interacted with other movies or not. Defining the NN starts with the first layer, as the embeddings are set to 8 thus the output of user embedding is 8 and the item embedding output is also 8 features vector. Concatenating both gives a vector of 16 features and thus input to the first layer is 16. The output of the first layer is 1024 features as we want to train the model efficiently and extract all the necessary features for predicting the problem statement. So, a linear layer or dense layer is a fully connected neural layer that contains many perceptrons. ReLU is determined as the activation function for the first layer. ReLU which is an activation function introduces non-linearity in training thus the data can distinguish easily while training.

Now the input of 1024 features to the second layer follows and the output of the ongoing layer is 512. Again, ReLU as the activation function. Decreasing the output features to half of the input vector help the NN to train linearly. After the second layer, the 3rd layer takes 512 features and gives an output of 256 features with an activation function known as ReLU. As the chronology maintains so the input to the 4th, 5th, 6th layers are 256, 128 and 64 features respectively whereas the ReLU known as the activation function remains the same for all. Alike the inputs the output also maintains the chronology of linear decreasing, therefore the output to the 4th, 5th, 6th layers are 128, 64 and 32 features respectively.

The last layer plays a significant role in predicting the movie as the input to the last layer is 32 features out of which only one feature is extracted. The one feature is considered as a vector of user-item interaction. The entity of the vectors is assigned by some probabilistic value that

Fig. 4 NN for the model



determines the prediction of the movies that a user have interacted. Thus, the recommendations will be the movies that have higher probable value. The use of activation function to this dense layer is sigmoid. As defined above sigmoid function squeezes the output into a range of [0,1]. Thus, every user-item interaction index will get a value within the range of [0,1] according to the predictions. Setting the Hyperparameter into the below settings the model gets trained-Total data-20000263, Testing data – 138493a, Batch Size-512, Epoch-5, Optimizer-Adam, Learning Rate of Adam optimizer-0.01 and GPU-1. The table obtained from the loss of every epoch is visualized below in Table 1.

As it can be seen while training the network the neural model the loss during 1st epoch eventually was very high that is 21% but gradually it decreases to 20% during 2nd epoch. In epoch number 3rd the gradual decrease in loss can be considered as a sharp decrease as the model at this stage shows only 18.7 percent losses. The loss remains at 18% for the next two epochs. The visualization is defined through the graph in Fig. 5.

The output of the model is the prediction of movies that would be recommended to the user according to their previous interaction with the movies. So, recommendations can easily visualize by defining a function that takes a user as its input and recommend the movies which are predicted by the NN in its predicted vector. But evaluating the model is our prime motive and thus we take 99 random data that are belonging to the negative class and add the test data we have achieved during the data split then rank them with the help of the predicted vector. If the rank of the movie we are testing is in the rank in between ten then it is a Hit whereas if the test data or movie is not ranked between 0 and 11 then it is a miss. As mentioned above if Hit ratio is determined as:

$$hitRatio@10 = \frac{\text{Total number of hits movie in rank of } 10}{\text{Total number of hit and miss}} \tag{5}$$

and the testing data is 138493, and the hit ratio for this model is determined to be 87% with a missing percentage of 13%. Introducing a series of dense layer increased the

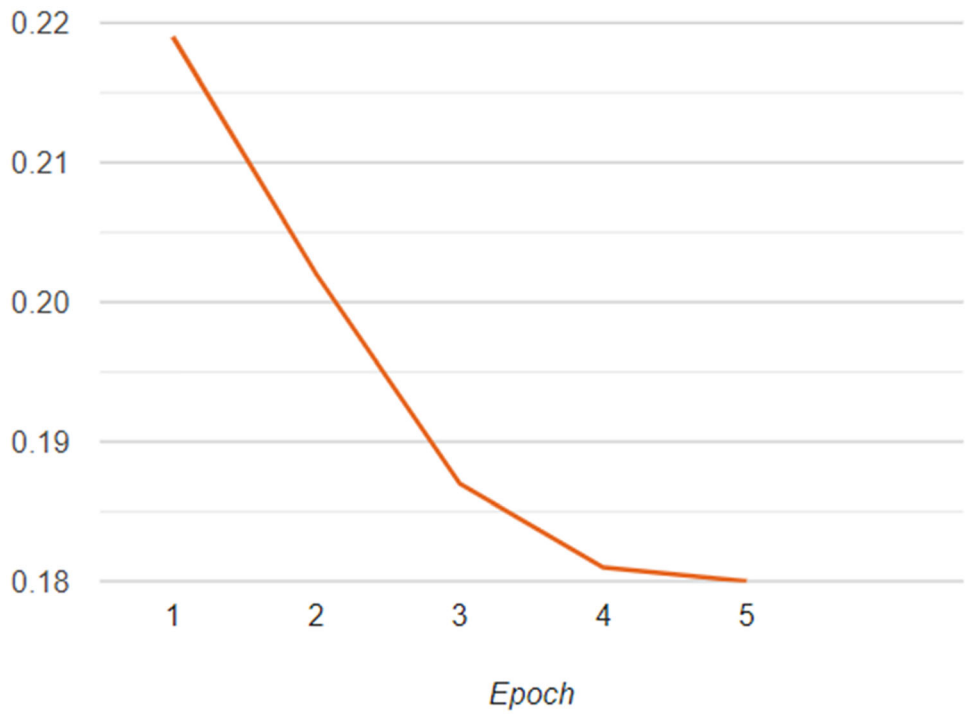


Fig. 5 Loss visualization graph

Table 2 Average Hit Ratio@10

Different Models Used	Hit_Ratio@10 taking average of epoch 5 and 8 factors
GMF [26]	0.69
MLP [26]	0.6
NeuMF [26]	0.68
SDAE [27]	0.89
NN with 7 layers (proposed)	0.87

efficacy as the movie lens data predicted in Neural-collaborative-filtering was 70–75% hit ratio [26], as this model have predicted more efficiently. The models compared are the GMF, MLP, NeuMF with the hit ratio @10. All the models are based on NNs and deep learning. The hit ratio@10 for the MLPs with 8 embeddings/features is MLP-0 = 0.452, MLP-1 = 0.628, MLP-2 = 0.65 and MLP-3 = 0.67. Similarly, the NeuMF and GMF give a HitRatio@10 are 0.0.69 and 0.68 respectively. Whereas the SDAE gives the highest ratio of 0.89. All the model uses the MovieLens 20 M data and thus we can see the average HitRatio@10 in Table 2 given below:

The proposed model gives a Hitratio @10 of 0.87 as mentioned above but the SDAE is considered to be more efficient as it uses three models which are hybridized. So, the comparison with other models is given below in Fig. 6.

From Fig. 6, the NN@7 or the proposed model shown in the graph is an efficient model. The model may not consider being best but the hit ratio of the model determines its efficacy.

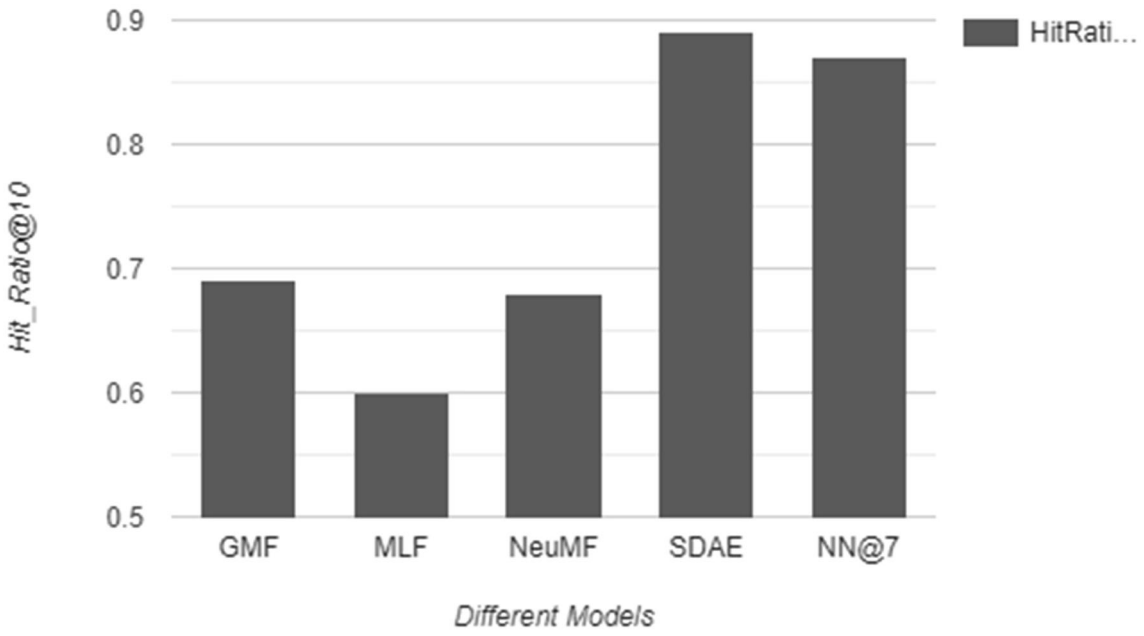


Fig. 6 Different models Hit Ratio@10

6 Conclusion

Prediction of the user-item vectors is the most important part to give the recommendations and to get the prediction the training must be efficient enough. Using a serial of dense layer not only performs good training but also helps in learning the parameters linearly. The data is converted to implicit data by manipulating it into the same class and finding four random data of negative class for each user and thus the first phase of the model is fulfilled. Whereas the second part is building a neural model, taking the help of neural-collaborative-filtering it was very easy to focus us into the NN architecture and we managed to make a NN that learns the parameters linearly. And the third phase of the model was to recommend the movie with the training prediction and evaluate the model. The recommendation was precisely done with the help of the ranking technique and the evaluation was carried out by a classical metric known as Hit-Ratio. The model not only gave positive feedback but also gave a hit ratio of 87%. Thus, the third objective of getting a higher accuracy is also fulfilled. The disadvantages of the model are the cold start problem but the collaboration does not deal with this type of problem. In future, new filtering models can be proposed or used for the development of a new recommendation model. The accuracy is also an important area for improvisations.

Funding None.

Declarations

Compliance with ethical standards None.

Ethical approval This article does not contain any studies with human participants or animals performed by any of the authors.

References

1. Xiangnan H, Lizi L, Hanwang Z, Liqiang N, Xia H, Tat SC (2017) Neural collaborative filtering. arXiv:1708.05031v2
2. Ting B, Ji-Rong W, Jun Z, Wayne XZ (2017) A neural collaborative filtering model with interaction-based neighbourhood. In Proceedings of the 2017 ACM on Conference on Information and Knowledge Management, pp 1979–1982. https://doi.org/10.1145/3132847.3133083
3. Liu YS (2018) A novel deep hybrid recommender system based on auto-encoder with neural collaborative filtering. Big Data Min Anal 1(3):211–221
4. Chen WF (2019) Joint neural collaborative filtering for recommender systems. ACM Trans Inf Syst (TOIS) 37:1–30

5. Li EC (2019) Learning binary codes with neural collaborative filtering for efficient recommendation systems. Knowl-Based Syst 172:64–75
6. Gao XW (2019) Context-aware QoS prediction with neural collaborative filtering for Internet-of-Things services. IEEE Internet Things J 7(5):4532–4542
7. Krishnan A, Ashish S, Hari S(2018) An adversarial approach to improve long-tail performance in neural collaborative filtering. In: Proceedings of the 27th ACM International Conference on information and knowledge management, pp 1491–1494. https://doi.org/10.1145/3269206.3269264
8. Khaki LW (2020) Predicting yield performance of parents in plant breeding: A neural collaborative filtering approach. PLoS ONE 15(5):e0233382
9. Li ZY, Bin G, Yan L, Yao J, Yi O, and Zhiwen Y (2018) Commercial site recommendation based on neural collaborative filtering. In: UbiComp '18: Proceedings of the 2018 ACM International Joint Conference and 2018 International Symposium on pervasive and ubiquitous computing and wearable computers, pp 138–141. https://doi.org/10.1145/3267305.3267592
10. Liu YS (2021) A neural collaborative filtering method for identifying miRNA-disease associations. Neurocomputing 422:176–185
11. Ren Z, Xia N, Andrew SL, Huzefa R (2019) Grade prediction with neural collaborative filtering. In: 2019 IEEE International Conference on data science and advanced analytics (DSAA), pp 1–10. https://doi.org/10.1109/DSAA.2019.00014
12. Islam RK (2021) Debiasing career recommendations with neural fair collaborative filtering. UMBC Faculty Collection
13. Girsang AW (2021) Neural collaborative for music recommendation system. IOP Conf Ser Mater Sci Eng 1071(1):012021
14. Bobadilla JS (2020) Deep learning architecture for collaborative filtering recommender systems. Appl Sci 10(7):2441
15. Chen LA, Angyu Z, Yinglan F, Fenfang X, Zibin Z (2018) Software service recommendation base on collaborative filtering neural network model. In: International Conference on service-oriented computing, Springer, Cham, pp 388–403. https://doi.org/10.1007/978-3-030-03596-9_28
16. Tran, Phong H, Nguyen HT, Ngoc-Thao N (2020) A hybrid approach for neural collaborative filtering. In: 2020 7th NAFOSTED Conference on information and computer science (NICS), IEEE, p 368–373. https://doi.org/10.1109/NICS51282.2020.9335910
17. Li AB (2021) Hyperbolic neural collaborative recommender. arXiv preprint arXiv:2104.07414
18. Zheng Y, Bangsheng T, Wenkui D, Hanning Z (2016) A neural autoregressive approach to collaborative filtering. In: International Conference on machine learning, PMLR, pp 764–773. http://proceedings.mlr.press/v48/zheng16.pdf
19. He GD (2021) Dual-embedding based neural collaborative filtering for recommender systems. arXiv preprint arXiv: arXiv:2102-02549
20. Faizin AIS (2020) Product recommender system using neural collaborative filtering for marketplace in Indonesia. IOP Conf Ser Mater Sci Eng 909(1):012072
21. Zeng YZQ (2019) Trust-based neural collaborative filtering. J Phys Conf Ser 122(1):012051
22. Grouplens M (2021) Movielens. https://grouplens.org/datasets/movielens/25m/. Accessed 01 Sep 2020
23. Yehuda K, Bell R (2009) Matrix factorization techniques for recommender systems. Computer 42(8):30–37
24. Agarap AF (2018) Deep learning using rectified linear units (relu). arXiv preprint arXiv:1803.08375
25. Kouretas I, Vassilis P (2019) Simplified hardware implementation of the softmax activation function. In: 2019 8th International Conference on Modern Circuits and Systems Technologies (MOCAST), IEEE, p 1–4
26. Areej ADQHAD (2020) Hit ratio: an evaluation metric for hashtag recommendation. arXiv:2010.01258
27. Huang Z, Yu C, Ni J, Liu H, Zeng C, Tang Y (2019) An efficient hybrid recommendation model with deep neural networks. IEEE Access 7:137900–137912
28. Afoudi Y, Lazaar M, Al Achhab M (2021) Hybrid recommendation system combined content-based filtering and collaborative prediction using artificial neural network. Simul Model Pract Theory 113:102375
29. Riaz Y, Deep U, Darad A (2021) Collaborative filtering based movie recommendation system (No. 5697). EasyChair
30. Thakker U, Patel R, Shah M (2021) A comprehensive analysis on movie recommendation system employing collaborative filtering. Multimed Tools Appl 80:1–26
31. Panda SK, Bhoi SK, Singh M (2020) A collaborative filtering recommendation algorithm based on normalization approach. J Ambient Intell Humaniz Comput 11:1–23