Received: January 2, 2023. Revised: February 12, 2023. 

INASS

International Journal of
Intelligent Engineering & Systems

http://www.inass.org/

# Deep Learning-Based Rate Prediction Model for Recommender System Using Clustering Techniques

**Ammar Abdulsalam Al-Asadi[1]*** **Mahdi Nsaif Jasim[2]**

[1]*Informatics Institute for Postgraduate Studies, Iraqi Commission for Computers and Informatics, Iraq*
[2]*College of Business Informatics, University of information technology and communications, Iraq*
* Corresponding author's Email: phd202010553@iips.icci.edu.iq

**Abstract:** Many deep learning-based recommender systems have been proposed recently. Where they involve all the users in datasets to build the latent representation of input data to be used later for predicting the missing rates. Despite the fact that, users have different interests, and these differences reduce the model prediction accuracy. This paper proposed a novel cluster-based denoising autoencoder model (cluster-based DAE) for rate prediction recommender systems. Instead of building a single model, it builds K models by using k-means algorithm to divide the users into groups based on their preferences. Each group trains a DAE model to generate recommendations for its members. The proposed method was trained and tested with MovieLens (100K, 1M, and 10M) datasets where 80% of the data are used for training and 20% for testing. The performance of the proposed method compared against other state-of-the-art methods that use deep learning to build rate prediction models. It outperformed the other compared methods in term of mean absolute error (with 12.9%, 14.7%, and 22.3%) and root mean squared error (with 24.2%, 18%, and 21.1%) using MovieLens 100K, 1M, and 10M datasets respectively.

**Keywords:** Recommender system, Deep learning, Denoising autoencoder, Clustering, K-means.

## 1. Introduction

The fast growth of internet applications and services produces a huge amount of information daily. It becomes a hard task for users to find contents that satisfy their desire when they use online applications with that rapid development of information [1]. For that reason, recommender systems (RSs) have become very necessary for users. RSs exclude unnecessary items to generate a list of recommended items for users based on their historical preferences [2]. Users' preferences can be represented in form of user-item interactions, where the interaction could be rate, buy, like, or click. Users and items features can be also utilized by RSs to generate personalized recommendations [3]. In the last decade, several recommender systems have been proposed. They can be nearly categorized into three types: collaborative filtering models (CF), content-based models (CB), and hybrid-based models [4].

CF approach has been quite effective approach. It utilizes the similarity among the users' preferences to generate recommendations [5]. It is based on the assumption that suggests users with similar tastes most likely have a similar opinion on an item [6]. On the other hand, CB approach try to recommend items that are most similar to the ones that are preferred by a user previously. Hybrid filtering combines CF and CB techniques to overcome the limitations of RSs [7].

Clustering techniques are the most widely used techniques in RSs for grouping users based on their similar interests [8]. They are unsupervised techniques that try to divide the data samples into clusters, where each cluster has the most similar samples (in terms of features) and they are different from the other cluster's samples [9].

There are many non-deep learning-based approaches of CF [6]. Matrix factorization (MF) techniques, such as probabilistic matrix factorization (PMF) [10] and biased matrix factorization

(BiasedMF) are particularly popular [11].

Deep learning-based recommender systems have been ardently studied lately, where the features of items and users are joint to generate predictions by following many perceptron layers [12]. Furthermore, deep learning (DL) techniques are being used in CF as well. For example, R. Salakhutdinov used restricted Boltzmann machines (RBM) in collaborative filtering with unsupervised nonlinear learning method [13]. Sedhain proposed AutoRec which is a novel framework for collaborative filtering. It uses autoencoder network to extract the latent space [14]. Zheng proposed a neural autoregressive architecture for collaborative filtering tasks, which is inspired by RBM model and neural autoregressive distribution estimator (NADE), the model called CF-NADE [15]. Zhuang proposed a framework called recommendation via dual-autoencoder (ReDa). It learns the latent representation of items and users using autoencoders, and reduces the variations of training data [16]. Yi proposed a deep learning-based CF framework, named as deep matrix factorization (DMF), which can integrate side information efficiently in a model [17]. Lee proposed a novel scalable deep learning-based collaborative filtering algorithm (Scalable DL) by using normalized vectors as inputs to a neural network to prevent network from overfitting [18]. Wang presented a deep learning-based RS model with two stages called TDR. Two separate marginalized stacked denoising auto-encoder models are applied to the items and users' features at the first stage to learn the latent space, then the output is used as input for the deep neural network (DNN) component for optimizing the model at the second stage [19]. Khan proposed a hybrid RS model named as deep semantic based topic driven hybrid RS (DST-HRS); it uses item's semantics description that is influenced by its topics information [20]. Zhang proposed a probabilistic matrix factorization model based on backpropagation neural network ensemble learning, bagging, and fuzzy clustering (FCM-bagging-BP-PMF) [21]. Sarridis proposed a neural interaction matrix factorization (NIMF) method that is applied to the rating matrix. In order to extract user and item embeddings; it takes a normalized rating matrix as input to the neural network [22]. Mondal presented DeCS which addressed the cold start problem in RSs by using deep neural network framework to learn low-dimensional embeddings and side information of the user and item [23]. Boudiba developed a tag-based model, which is extracted from contextual BERT. The proposed model uses multi perceptron layers architecture and named as neural CF-MLP [24].

The problem with the reviewed deep learning-based CF models is that, they use the rates of all users in dataset to build the latent space which will be used later to predict the missing rates of each user. Despite the fact that, users have different interests, and these differences reduce the model predictions accuracy.

This research proposes a novel model that involves the users with most similar preferences in recommendations generation process, instead of involving all the users. In other words, it distributes the users over multiple models based on their similarity. The proposed system combines clustering technique such as K-means, with a deep learning model such as denoising autoencoder (DAE) to improve the prediction accuracy.

The proposed method has some important differences if it is compared with other autoencoder-based models:

a) Instead of using the whole users in the dataset to train the DAE model, it divides the users into k clusters based on their interest similarity by using K-means algorithm, then it uses the members of each cluster to train their own DAE model.
b) It uses explicit feedback (1-5 rates) in form of item-user interactions with added noise (corruption) to the input data, to learn latent representations of corrupted item-user preferences that can best reconstruct the full input.

The rest of the paper is structured as follows: Section 2 presented the theoretical background of the techniques that are used in the proposed method; proposed method is explained in section 3; results and performance evaluation are presented in section 4 and section 5 covers the paper conclusions.

## 2. Background and theories

Our method is based on the following techniques: clustering and autoencoders. In this section, we will briefly discuss these topics.

### 2.1 Clustering

K-Means is an unsupervised clustering algorithm. It is the simplest, most used, and computationally efficient clustering algorithm [25]. K-means is used for partitioning data into K clusters using clusters' centres (centroids). The centroid of each cluster is computed by taking the average of all data points in the cluster. Determining the number

of clusters before the model training is required in this method. The main 4 steps of the K-means algorithm are described as follow [26]:

1. Randomly select K data points as initial centroids.
2. Assign each data point Xi to the closest cluster by calculating its distance to all centroids using Euclidean distance.
3. Update the centroids by calculating the mean of the assigned points.
4. Repeat steps 2 and 3 until no convergence or maximum iteration is met.

The optimal number of clusters (k) can be determined by using Silhouette method. It validates the consistency of data within each cluster. The silhouette method measures how much a point is similar to its cluster compared to other clusters by calculates silhouette coefficients of each point [27].

The silhouette coefficient for a sample is computed as follows:

$$\text{Silhouette Coefficient} = \frac{(b-a)}{max(a,b)} \quad (1)$$

Where, a is the mean intra-cluster distance, and b is the mean nearest-cluster distance.

### 2.2 Autoencoder

Autoencoder (AE) was first presented in 1991 by Kramer [28]. It exploited feed-forward neural networks to learn the latent representation of an input with low dimensions [29]. The output of the AE aims to reconstruct the input. Then, the network is trained by back-propagating the loss score (e.g., mean squared error) during the reconstruction, it consists of two parts as below,

$$\text{Encoder } \varphi: x \rightarrow z \quad (2)$$

$$\text{Decoder } \Psi: z \rightarrow x \quad (3)$$

Where $\varphi, \Psi = \arg min_{\varphi, \Psi} \|x - (\varphi \,.\, \Psi)\, x\|2$. In the simplest case, there is only one hidden layer, where the encoder takes input x and maps it to z, then the decoder maps z into reconstruction x,

$$\text{Encoder: } z = \sigma(Wx + b) \quad (4)$$

$$\text{Decoder: } x = \sigma(W'z + b') \quad (5)$$

Where σ is a non-linear activation function, $x \in R^d$ is the input, $z \in R^d$ is the hidden node, $W \in R^{d.k}$ is weight matrix mapping input to hidden node, $W' \in R^{d.k}$ is the weight matrix mapping hidden node to reconstruction node, $b \in R^k$, $b' \in R^d$ as bias vectors [30].

### 2.3 Denoising autoencoder

Vincent [31] introduced the denoising autoencoder (DAE) to discover more robust features through autoencoders and learning the identity function. DAE applies, corrupted version of input x as $\tilde{x}$, and the network is trained to denoise and reconstruct input x. Many corruption ways can be used, but the most common choices are multiplicative mask-out/ drop-out noise and additive Gaussian noise. In this paper, the drop-out noise is used which randomly masks entries of the input by setting them to zero [32].

## 3. Proposed method

The proposed method consists of two main steps: users clustering and training DAE model. The first step divides the users into clusters based on their similar preferences. These preferences are extracted from the rates that are given by a user to items' features. MovieLens dataset is used in the proposed system, where movies represent the items and genres represent the items' features. Two tables have been used out of the dataset:

1) Ratings: it has all the rates given by users to movies on a scale between 1 and 5.
2) Movies: it has all movies with their genres, for example, action, drama, comedy, etc. there are 19 different genres.

### 3.1 Users clustering

The proposed system groups the users with similar interests together (clustering) by applying k-means algorithm. The similarities among users are found by extracting the average rates that is given by a user to each genre. Fig. 1 shows a part of the users-genres interaction matrix. This matrix is obtained by merging ratings with movies tables based on movies' ids. In order to find the best number of clusters (K) for the experimented dataset, silhouette method is applied by using Eq. (1). Silhouette score is computed by trying different values of K and the value with highest score is selected.

### 3.2 Training DAE model

In the training step, the dataset will be divided into K sets, based on the cluster value. Each set will

| user_id | Action | Adventure | Animation | Children's | Comedy | Crime | Documentary | Drama | Fantasy | Film-Noir | Horror | Musical | Mystery | Romance |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 4.200000 | 4.000000 | 4.111111 | 4.250000 | 4.142857 | 4.000000 | 0.000000 | 4.428571 | 4.00 | 0.000000 | 0.000000 | 4.285714 | 0.000000 | 3.666667 |
| 2 | 3.500000 | 3.736842 | 0.000000 | 0.000000 | 3.560000 | 3.583333 | 0.000000 | 3.898734 | 3.00 | 4.000000 | 3.000000 | 0.000000 | 3.333333 | 3.708333 |
| 3 | 3.956522 | 4.000000 | 4.000000 | 4.000000 | 3.766667 | 0.000000 | 0.000000 | 4.000000 | 4.50 | 0.000000 | 2.666667 | 4.000000 | 3.000000 | 3.800000 |
| 4 | 4.157895 | 3.833333 | 0.000000 | 4.000000 | 0.000000 | 5.000000 | 0.000000 | 4.166667 | 4.50 | 0.000000 | 4.333333 | 0.000000 | 0.000000 | 4.000000 |
| 5 | 2.612903 | 3.000000 | 4.000000 | 3.833333 | 3.410714 | 3.285714 | 3.666667 | 3.096154 | 0.00 | 4.000000 | 2.800000 | 3.333333 | 3.125000 | 3.100000 |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 6036 | 3.000000 | 2.987952 | 3.911765 | 3.444444 | 3.203065 | 3.528302 | 3.909091 | 3.505376 | 3.00 | 4.058824 | 2.986486 | 3.709677 | 3.411765 | 3.352459 |
| 6037 | 3.642857 | 4.000000 | 4.000000 | 3.666667 | 3.576271 | 3.833333 | 4.000000 | 3.877551 | 4.25 | 3.444444 | 4.111111 | 4.000000 | 3.692308 | 3.681818 |
| 6038 | 3.000000 | 4.000000 | 3.666667 | 3.000000 | 3.833333 | 0.000000 | 0.000000 | 3.888889 | 0.00 | 0.000000 | 2.500000 | 0.000000 | 0.000000 | 4.166667 |
| 6039 | 4.000000 | 4.100000 | 3.615385 | 3.529412 | 3.723077 | 4.000000 | 0.000000 | 4.000000 | 3.60 | 4.500000 | 4.000000 | 3.690476 | 4.176471 | 3.800000 |
| 6040 | 2.976190 | 2.818182 | 3.000000 | 4.000000 | 3.274510 | 3.920000 | 4.500000 | 3.821622 | 3.50 | 4.000000 | 2.590909 | 4.000000 | 4.454545 | 3.488889 |

Figure. 1 a part of the users-genres interaction matrix

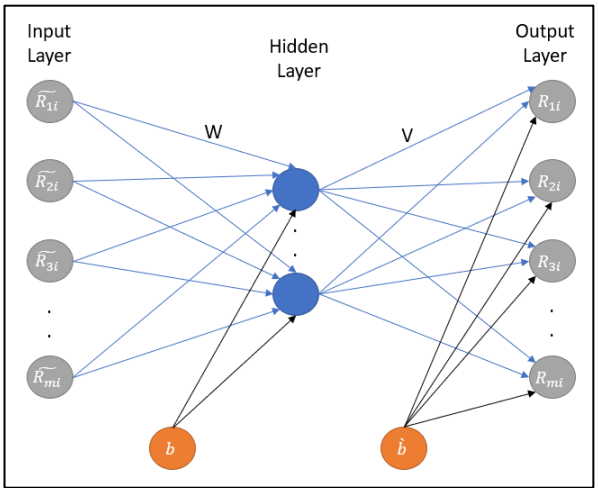


Figure. 2 The proposed model structure

be further divided into training and testing sets with 80% and 20% of data respectively. All the clusters use the same model structure.

The proposed model follows item-based structure, where it uses the items-users matrix as input ($r_i$), where items are represented as rows and users as columns, the inter-section between a row and a column represents the actual rate that is given by a user to an item. Item-based structure is used because the number of users is less than items, as a result of the clustering process, and that will reduce the number of nodes in the input and output layers. The proposed model structure is shown in Fig. 2.

### 3.2.1. DAE model design

The proposed DAE model is designed as follows:

1) Input layer

The number of nodes in the input layer equals the number of users, where the inputs nodes represent the interaction between an item $i$ and all users in a specific cluster, $r_i = \{R_{1i}, R_{2i}, R_{3i}, R_{mi}\}$ where $R$ is the rate value, m is the number of users and $i$ is an item.

Dropout noise will be applied to mask out the input data by setting them to zero randomly, with a noise ratio equal to (50%) to introduce a corrupted version of input data $\tilde{r}_i$.

2) Hidden layer

$\tilde{r}_i$ will be fully connected to the hidden layer which has (256) nods to represent the latent space of the input data. The activation function that is used in the hidden layer is the sigmoid function. The output of the hidden layer is computed as follows:

$$z = f(W * \tilde{r}_i + b) \tag{6}$$

Where, $f$ is the sigmoid activation function, $\tilde{r}_i$ is a corrupted input data, $W$ and $b$ are the weights and bias of the hidden layer respectively.

3) Output layer

The output layer is fully connected with the hidden layer where it has the same number of nods that is in the input layer with weights' values (V). The output of this layer is the predicted rates $\hat{r}_i$, it is computed as follows:

$$\hat{r}_i = f(V * z + \grave{b}) \tag{7}$$

Where, $f$ is the linear activation function, $z$ is the output of the hidden layer, $V$ and $\grave{b}$ are the weights and bias of the output layer respectively.

4) Loss function

The proposed model uses mean squared error (MSE) as a loss function. But, since it doesn't make sense to predict zeros in the item's representation

vector $R_{mi}$, the approach from [14] is flowed to optimize masked mean squared error (MMSE) loss [6].

$$\text{MMSE} = \frac{\sum_{i=1}^{i=n} m_i.(r_i - \hat{r}_i)^2}{\sum_{i=1}^{i=n} m_i} \quad (8)$$

Where $r_i$ is actual rating, $\hat{r}_i$ is reconstructed, or predicted rating, and $m_i$ is a mask function such that $m_i$ = 1 if $r_i \neq 0$ else $m_i$ = 0.

5) Optimization algorithm

Adam optimization algorithm has been used to update the network's weights. It combines 'gradient descent with momentum algorithm' and the 'root mean square propagation algorithm' by using the following equations:

$$p_t = m_1.p_{t-1} + (1 - m_1)\left[\frac{\partial f}{\partial w}\right] \quad (9)$$

Where, $p$ is aggregate of gradients, $m$ is the moving average parameter, $\partial f$ is derivative of loss function, and $\partial$w is derivative of weights.

$$q_t = m_2.q_{t-1} + (1 - m_2)\left[\frac{\partial f}{\partial w}\right] \quad (10)$$

Where, $q_t$ is the sum of square of past gradients.

$$\widehat{p_t} = \frac{p_t}{(1 - m_1^t)} \quad (11)$$

$$\widehat{q_t} = \frac{q_t}{(1 - m_2^t)} \quad (12)$$

Where, $\widehat{p_t}$ and $\widehat{q_t}$ are the bias corrected weight parameters.

$$W_t = W_{t-1} - a\,.\left(\frac{\widehat{p_t}}{\sqrt{\widehat{q_t} + \epsilon}}\right) \quad (13)$$

Where, $W_t$ is the new weight, $a$ is the learning rate, and $\epsilon$ is a small positive constant to avoid division by zero.

The main steps of the proposed model are illustrated in algorithm 1.

### 3.3. Evaluation metrics

Root mean squared error (RMSE) and mean absolute error (MAE) are used as evaluation matrices. They evaluate the accuracy of the predicted rates by comparing them with the actual testing data. RMSE has a straightforward relation with MMSE score.

Algorithm 1. Cluster-based denoising autoencoder RS

**Input**: Ratings data;
**Output**: Predicted rates;

1 k ← max cluster value
2 $i$ ← 0
3 **while** $i$ <= k do
4 Get all ratings of cluster $i$
5 m ← number of items rated in cluster $i$
6 n ← number of users in cluster $i$
7 Create m×n ratings array $r_i$
8 Split the array into train and test sets
9 Build AE model with input/output nodes = number of users in cluster $i$
10 **For** each item in train set do
11 $\tilde{r}_i$ ← dropout_noise($r_i$, 0.5)
12 Calculate z using Eq. (6)
13 Calculate $\hat{r}_i$ using Eq. (7)
14 Calculate MMSE using Eq. (8)
15 Update parameters using Eq. (9, 10, 11, 12, and 13)
16 **End for**
17 Generate predictions
19 $i$ ← $i$ +1
20 **End while**

$$\text{RMSE} = \sqrt{MMSE} \quad (14)$$

MAE computes the mean of the absolute differences between the actual rating $r_i$ and the reconstructed rating $\hat{r}_i$ as follows:

$$\text{MAE} = \frac{\sum_{i=1}^{i=n} m_i.|r_i - \hat{r}_i|}{\sum_{i=1}^{i=n} m_i} \quad (15)$$

Where, $m_i$ = 1 if $r_i \neq 0$ else $m_i$ = 0.

## 4. Results and discussion

Three of MovieLens (ML) datasets (ML-100K, ML-1M, and ML-10M) are applied to the proposed method. All of datasets have the same tables with the same attributes. But they have different numbers of users, movies, and ratings as shown in Table 1.

Table 2. Datasets statistics

| Dataset | Users | Movies | Ratings |
|---|---|---|---|
| ML-100K | 943 | 1,682 | 100,000 |
| ML-1M | 6,040 | 3,706 | 1,000,209 |
| ML-10M | 69,878 | 10,681 | 10,000,052 |

In order to determine the number of clusters (K) for each dataset, Silhouette method is used to compute the scores of different values of K (between 2 and 20) and the highest score is selected. The selected K values for the experimented datasets were as follows: 5, 11, and 23 clusters for ML-100K, ML-1M, and ML-10M datasets respectively. Where it is obvious that, as the size of data increases the number of clusters increases too.

MovieLens datasets are used to train and test the proposed model (cluster-based DAE). They are divided into 80% and 20% for training and testing respectively. The hyperparameters configuration is selected by trying different settings and they are evaluated based on the MAE and RMSE scores. The hyperparameters of the proposed model are:

- Sigmoid and linear activation functions for hidden and output layers respectively.
- Adam optimizer with learning rate (0.0001).
- The noise ratio is 50%.
- Regularization rate is (0.0001).
- Number of epochs is (500).
- The number of hidden nodes is (256) in ML-100K and ML-1M datasets models, and (512) in ML-10M dataset model.

The experiment was performed on a laptop PC equipped with an Intel(R) Core(TM) i7-11800H CPU@2.3GHz, NVIDIA GeForce RTX 3060 GPU, and 16 GB RAM. The proposed model is implemented by using python 3.9.12 programming language with keras 2.9.0, tensorflow 2.9.1, numpy 1.22.4, pandas 1.4.2, and scikit-learn 1.1.1.

The experimental results of the proposed model are performed over the three datasets by computing the average of MAE and RMSE of all clusters' models. Tables 2, 3, and 4 present the results of applying ML-100K, ML-1M and ML10M datasets respectively. Fig. 3 summarizes the performance results of the proposed model over the three datasets.

Table 2. Results of each cluster's model in ML-100K dataset

| Cluster | # Items (Samples) | #Users (Input nodes) | MAE | RMSE |
|---|---|---|---|---|
| 1 | 1570 | 234 | 0.6043 | 0.6760 |
| 2 | 1036 | 177 | 0.5041 | 0.5035 |
| 3 | 1346 | 134 | 0.4683 | 0.7480 |
| 4 | 1335 | 226 | 0.6339 | 0.4950 |
| 5 | 1022 | 172 | 0.6495 | 0.5149 |
| | | Total: 943 | Average: **0.5720** | Average: **0.6313** |

Table 3. Results of each cluster's model in ML-1M dataset

| Cluster | # Items (Samples) | #Users (Input nodes) | MAE | RMSE |
|---|---|---|---|---|
| 1 | 3194 | 742 | 0.5823 | 0.6760 |
| 2 | 2715 | 377 | 0.4690 | 0.5035 |
| 3 | 3268 | 652 | 0.6440 | 0.7480 |
| 4 | 2483 | 361 | 0.4574 | 0.4950 |
| 5 | 2386 | 375 | 0.4818 | 0.5149 |
| 6 | 2404 | 370 | 0.5355 | 0.5836 |
| 7 | 2281 | 326 | 0.4551 | 0.4925 |
| 8 | 3563 | 783 | 0.6617 | 0.7840 |
| 9 | 2616 | 510 | 0.5490 | 0.6034 |
| 10 | 2839 | 542 | 0.5795 | 0.6539 |
| 11 | 3533 | 1002 | 0.6300 | 0.7441 |
| | | Total: 6040 | Average: **0.5496** | Average: **0.6181** |

Table 4. Results of each cluster's model in ML-10M dataset

| Cluster | # Items (Samples) | #Users (Input nodes) | MAE | RMSE |
|---|---|---|---|---|
| 1 | 5946 | 5625 | 0.5013 | 0.5738 |
| 2 | 10232 | 7881 | 0.56 | 0.6712 |
| 3 | 4224 | 1708 | 0.467 | 0.5145 |
| 4 | 10632 | 6853 | 0.5877 | 0.7121 |
| 5 | 4196 | 1401 | 0.4688 | 0.5135 |
| 6 | 3526 | 2689 | 0.4523 | 0.4976 |
| 7 | 5462 | 3670 | 0.5197 | 0.5844 |
| 8 | 6823 | 1829 | 0.5039 | 0.5673 |
| 9 | 8765 | 7944 | 0.5336 | 0.6253 |
| 10 | 7003 | 2187 | 0.4739 | 0.5389 |
| 11 | 5574 | 1631 | 0.4804 | 0.5365 |
| 12 | 7866 | 1924 | 0.4627 | 0.5187 |
| 13 | 4389 | 2047 | 0.5138 | 0.5646 |
| 14 | 6118 | 2795 | 0.523 | 0.5872 |
| 15 | 4664 | 2098 | 0.5044 | 0.5589 |
| 16 | 3422 | 1175 | 0.4394 | 0.4812 |
| 17 | 6648 | 5304 | 0.5263 | 0.6006 |
| 18 | 3847 | 2473 | 0.4892 | 0.5381 |
| 19 | 4357 | 2570 | 0.4851 | 0.5392 |
| 20 | 5132 | 858 | 0.4654 | 0.5033 |
| 21 | 6478 | 2485 | 0.5097 | 0.571 |
| 22 | 5919 | 967 | 0.4697 | 0.518 |
| 23 | 7127 | 1764 | 0.4937 | 0.5548 |
| | | Total: 69878 | Average: **0.4970** | Average: **0.5596** |

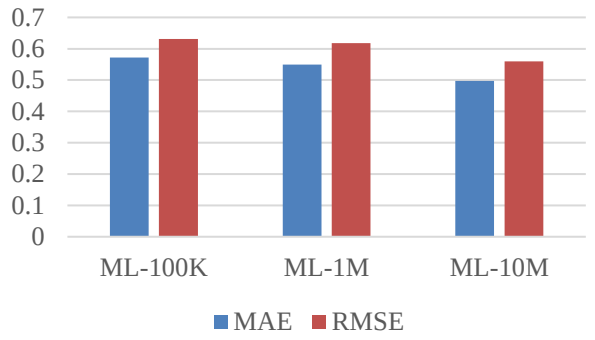


Figure. 3 Performance of the proposed model over different MovieLens datasets

Tables 2, 3, and 4 show the number of items and users that is used to train each cluster's model with their interactions over ML-100K, ML-1M, and ML-10M datasets respectively. The MAE and RMSE scores of all clusters' models are evaluated. The average MAE and RMSE of ML-100K dataset are (0.5720) and (0.6313) respectively. In ML-1M dataset the average MAE and RMSE are (0.5496) and (0.6181) respectively. Where the average MAE and RMSE of ML-10M are (0.4970) and (0.5596) respectively.

The proposed model is compared with other state-of-the-art deep learning-based rate prediction methods in terms of prediction quality (MAE and RMSE) using ML-100K, ML-1M, and ML-10M datasets. Tables 5, 6, and 7 show the MAE and RMSE results of deep learning-based models that are compared with our proposed model in ML-100K, ML-1M, and ML-10M datasets respectively.

Table 5 shows the effectiveness of the proposed method on ML-100K dataset, which outperformed the other compered methods in term of MAE (by a range between 21% and 12.9%) and RMSE (by a range between 33.9% and 24.2%). Table 6 proved that the proposed method has outperformed the other methods in term of MAE (by a range between 18.1% and 14.7%) and RMSE (by a range between 27.1% and 18%). The effectiveness of the proposed method on ML-10M dataset presented in Table 7, which outperformed the other compered methods in term of MAE (by 22.3%) and RMSE (by a range between 37% and 21.1%).

Table 5. Compassion between the proposed model and other models on ML-100k dataset

| Methods | MAE | RMSE |
|---|---|---|
| PMF * | 0.782 | 0.970 |
| ReDa [16] | 0.720 | 0.919 |
| Scalable DL [18] | - | 0.907 |
| NIMF [22] | - | 0.894 |
| DeCS [23] | 0.676 | 0.891 |
| DMF+ * | 0.655 | 0.889 |
| TDR [19] | 0.701 | 0.873 |
| Cluster-based DAE (proposed) | **0.572** | **0.631** |

*: Taken from [17].

Table 6. Compassion between the proposed model and other models on ML-1M dataset

| Methods | MAE | RMSE |
|---|---|---|
| PMF * | 0.697 | 0.889 |
| RBM ** | - | 0.854 |
| ReDa [16] | 0.665 | 0.849 |
| Scalable DL [18] | - | 0.848 |
| DST-HRS [20] | - | 0.846 |
| BiasedMF ** | - | 0.845 |
| DeCS [23] | 0.628 | 0.842 |
| TDR [19] | 0.655 | 0.835 |
| DMF+ * | 0.608 | 0.832 |
| AutoRec ** | - | 0.831 |
| CF-NADE [15] | - | 0.829 |
| NIMF [22] | - | 0.829 |
| FCM-bagging-BP-PMF [21] | 0.731 | 0.798 |
| Cluster-based DAE (proposed) | **0.550** | **0.618** |

*: Taken from [17].
**: Taken from [14].

Table 7. Compassion between the proposed model and other models on ML-10M dataset

| Methods | MAE | RMSE |
|---|---|---|
| Neural CF-MLP [24] | 0.720 | 0.930 |
| BiasedMF ** | - | 0.845 |
| RBM ** | - | 0.825 |
| AutoRec ** | - | 0.782 |
| NIMF [22] | | 0.781 |
| DST-HRS [20] | - | 0.779 |
| CF-NADE [15] | - | 0.771 |
| Cluster-based DAE (proposed) | **0.497** | **0.560** |

*: Taken from [17].
**: Taken from [14].

The reason that makes the proposed method outperforms the other compared methods is that, we utilized the similarity of users' preferences to distribute the users over K models (instead of single model), where the predictions are generated by similar users.

On the other hand, the training time of the proposed method is longer than training the same model without clustering. For example, in ML-1M

dataset the proposed method took (7 minutes and 45 seconds) to train cluster-based DAE. Whereas, it took (1 minutes and 50 seconds) to train a DAE model without clustering. Because, the proposed method trains multiple models instead of one.

## 5. Conclusion

This research proposed a cluster-based denoising autoencoder model for rate prediction recommender systems. It distributed the users over K models instead of a single model. The proposed system utilized k-means algorithm to divide the users into K clusters based on their similar interests. Each cluster's members cooperate to extract the latent space of items-users interactions to predict the missing rates using denoising autoencoder model. The proposed method was trained and tested with MovieLens (100K, 1M, and 10M) datasets where 80 % of the data are used for training and 20% for testing. The performance of the proposed method compared against other state-of-the-art methods that use deep learning to build rate prediction models. It outperformed the other compared methods in term of MAE (with 12.9%, 14.7%, and 22.3%) and RMSE (with 24.2%, 18%, and 21.1%) using 100K, 1M, and 10M datasets respectively. The proposed model requires more training time than using the same model without clustering which we will try to reduce it in future work. Moreover, the parallel computing of proposed method is also worth exploring.

## Conflicts of interest

The authors declare no conflict of interest.

## Author contributions

Ammar A. Al-Asadi provided the idea, technique, software, formal analysis, materials, data collection, and writing-original version preparation. Mahdi Nsaif Jasim provided supervision, revision, and editing the research.

## References

[1] A. Neamah and A. E. Ameer, "Design and Evaluation of a Course Recommender System Using Content-Based Approach", In: *Proc. of 2018 International Conference on Advanced Science and Engineering (ICOASE)*, pp. 1–6, 2018.

[2] S. Lazemi and H. E. Komleh, "Improving collaborative recommender systems via emotional features", In: *Proc. of 2016 IEEE 10th International Conference on Application of Information and Communication Technologies (AICT)*, pp. 1–5, 2016.

[3] S. Zhang, L. Yao, A. Sun, and Y. Tay, "Deep learning based recommender system: A survey and new perspectives", *ACM Computing Surveys (CSUR)*, Vol. 52, No. 1, pp. 1–38, 2019.

[4] G. Zhang, Y. Liu, and X. Jin, "A survey of autoencoder-based recommender systems", *Front Comput Sci*, Vol. 14, No. 2, pp. 430–450, 2020.

[5] Y. Koren, S. Rendle, and R. Bell, "Advances in collaborative filtering", *Recommender Systems Handbook*, pp. 91–142, 2022.

[6] O. Kuchaiev and B. Ginsburg, "Training deep autoencoders for collaborative filtering", *ArXiv Preprint ArXiv:1708.01715*, 2017.

[7] M. Mohamed, M. Khafagy, and M. Ibrahim, "Recommender systems challenges and solutions survey", In: *Proc. of 2019 International Conference on Innovative Trends in Computer Engineering (ITCE)*, 2019, pp. 149–155.

[8] R. Logesh, V. Subramaniyaswamy, D. Malathi, N. Sivaramakrishnan, and V. Vijayakumar, "Enhancing recommendation stability of collaborative filtering recommender system through bio-inspired clustering ensemble method", *Neural Comput Appl*, Vol. 32, No. 7, pp. 2141–2164, 2020.

[9] R. Rashidi, K. Khamforoosh, and A. Sheikhahmadi, "Proposing improved meta-heuristic algorithms for clustering and separating users in the recommender systems", *Electronic Commerce Research*, Vol. 22, No. 2, pp. 623–648, 2022.

[10] A. Mnih and R. Salakhutdinov, "Probabilistic matrix factorization", *Adv Neural Inf Process Syst*, Vol. 20, 2007.

[11] Y. Koren, R. Bell, and C. Volinsky, "Matrix factorization techniques for recommender systems", *Computer (Long Beach Calif)*, Vol. 42, No. 8, pp. 30–37, 2009.

[12] P. Covington, J. Adams, and E. Sargin, "Deep neural networks for youtube recommendations", In: *Proc of the 10th ACM Conference on Recommender Systems*, pp. 191–198, 2016.

[13] R. Salakhutdinov, A. Mnih, and G. Hinton, "Restricted Boltzmann machines for collaborative filtering", In: *Proc of the 24th International Conference on Machine Learning*, pp. 791–798, 2007.

[14] S. Sedhain, A. K. Menon, S. Sanner, and L. Xie, "Autorec: Autoencoders meet collaborative filtering", In: *Proc of the 24th International Conference on World Wide Web*, pp. 111–112,

2015.
[15] Y. Zheng, B. Tang, W. Ding, and H. Zhou, "A neural autoregressive approach to collaborative filtering", In: *Proc. of International Conference on Machine Learning*, pp. 764–773, 2016.
[16] F. Zhuang, Z. Zhang, M. Qian, C. Shi, X. Xie, and Q. He, "Representation learning via dual-autoencoder for recommendation", *Neural Networks*, Vol. 90, pp. 83–89, 2017.
[17] B. Yi, X. Shen, H. Liu, Z. Zhang, W. Zhang, S. Liu, and N. Xiong, "Deep matrix factorization with implicit feedback embedding for recommendation system", *IEEE Trans Industr Inform*, Vol. 15, No. 8, pp. 4591–4601, 2019.
[18] H. Lee and J. Lee, "Scalable deep learning-based recommendation systems", *ICT Express*, Vol. 5, No. 2, pp. 84–88, 2019.
[19] R. Wang, Y. Jiang, and J. Lou, "TDR: Two-stage deep recommendation model based on mSDA and DNN", *Expert Syst Appl*, Vol. 145, p. 113116, 2020.
[20] Z. Khan, N. Iltaf, H. Afzal, and H. Abbas, "DST-HRS: A topic driven hybrid recommender system based on deep semantics", *Comput Commun*, Vol. 156, pp. 183–191, 2020.
[21] Z. Zhang, G. Huang, Y. Zhang, S. Wei, B. Shi, J. Jiang, and B. Liang, "Research on PMF Model Based on BP Neural Network Ensemble Learning Bagging and Fuzzy Clustering", *Complexity*, Vol. 2021, 2021.
[22] I. Sarridis and C. Kotropoulos, "Neural Factorization Applied to Interaction Matrix for Recommendation", In: *Proc. of 2021 29th European Signal Processing Conference (EUSIPCO)*, pp. 1336–1340, 2021.
[23] R. Mondal and B. Bhowmik, "DeCS: A Deep Neural Network Framework for Cold Start Problem in Recommender Systems", In: *Proc. of 2022 IEEE Region 10 Symposium (TENSYMP)*, pp. 1–6, 2022.
[24] T. Boudiba and T. Dkaki, "Exploring Contextualized Tag-based Embeddings for Neural Collaborative Filtering", In: *Proc. of International Conference on Agents and Artificial Intelligence*, pp. 158–166, 2022.
[25] X. Ran, X. Zhou, M. Lei, W. Tepsan, and W. Deng, "A novel k-means clustering algorithm with a noise algorithm for capturing urban hotspots", *Applied Sciences*, Vol. 11, No. 23, p. 11202, 2021.
[26] T. Puraram, P. Chaovalit, A. Peethong, P. Tiyanunti, S. Charoensiriwath, and W. Kimpan, "Thai Food Recommendation System using Hybrid of Particle Swarm Optimization and K-Means Algorithm", In: *Proc. of 2021 6th International Conference on Machine Learning Technologies*, pp. 90–95, 2021.
[27] P. Rousseeuw, "Silhouettes: a graphical aid to the interpretation and validation of cluster analysis", *J Comput Appl Math*, Vol. 20, pp. 53–65, 1987.
[28] M. Kramer, "Nonlinear principal component analysis using autoassociative neural networks", *AIChE Journal*, Vol. 37, No. 2, pp. 233–243, 1991.
[29] M. Yu, T. Quan, Q. Peng, X. Yu, and L. Liu, "A model-based collaborate filtering algorithm based on stacked AutoEncoder", *Neural Comput Appl*, Vol. 34, No. 4, pp. 2503–2511, 2022.
[30] Y. Pan, F. He, and H. Yu, "A novel enhanced collaborative autoencoder with knowledge distillation for top-N recommender systems", *Neurocomputing*, Vol. 332, pp. 137–148, 2019.
[31] P. Vincent, H. Larochelle, Y. Bengio, and P. A. Manzagol, "Extracting and composing robust features with denoising autoencoders", In: *Proc of the 25th International Conference on Machine Learning*, pp. 1096–1103, 2008.
[32] Y. Xiong and R. Zuo, "Robust feature extraction for geochemical anomaly recognition using a stacked convolutional denoising autoencoder", *Math Geosci*, Vol. 54, No. 3, pp. 623–644, 2022.