

## An improved hybrid semi-stacked autoencoder for item-features of recommendation system (iHSARS)

Ahed Mleih Al-Sbou<sup>1,2</sup>, Noor Hafhizah Abd Rahim<sup>2</sup>

<sup>1</sup>Department of Computer Science, Faculty of Information Technology, Al-Hussein Bin Talal University, Ma'an, Jordan

<sup>2</sup>Faculty of Ocean Engineering Technology and Informatics, Universiti Malaysia Terengganu, Kuala Terengganu, Malaysia

---

### Article Info

#### Article history:

Received Sep 13, 2022

Revised Dec 5, 2022

Accepted Dec 10, 2022

---

#### Keywords:

Autoencoder

Deep learning

Deep-autoencoder

Recommendation systems

Semi-autoencoder

---

### ABSTRACT

In recent years, information overload has become a phenomenon where it makes people difficult to filter relevant information. To address issues such as high-dimensional data, cold start, and data sparsity, semi-autoencoder is one of the unsupervised deep learning methods used in the recommendation systems. It is particularly useful for reducing data dimensions, capturing latent representations, and flexibly reconstructing various parts of input data. In this article, we propose an improved hybrid semi-stacked autoencoder for item-features of recommendation system (iHSARS) framework. This method aims to show better performance of the hybrid collaborative recommendation via semi-autoencoder (HRSA) technique. Two novel elements for iHSARS's architecture have been introduced. The first element is an increase sources of side information of the input layer, while the second element is the number of hidden layers has been expanded. To verify the improvement of the model, MovieLens-100K and MovieLens-1M datasets have been applied to the model. The comparison between the proposed model and different state-of-the-art methods has been carried using mean absolute error (MAE) and root mean square error (RMSE) metrics. The experiments demonstrate that our framework improved the efficiency of the recommendation system better than others.

*This is an open access article under the [CC BY-SA](https://creativecommons.org/licenses/by-sa/4.0/) license.*



---

### Corresponding Author:

Noor Hafhizah Abd Rahim

Faculty of Ocean Engineering Technology and Informatics, Universiti Malaysia Terengganu

Kuala Terengganu, Malaysia

Email: noorhafhizah@umt.edu.my

---

## 1. INTRODUCTION

With the explosion of information on the Internet in recent years, the problem of information overload arises [1]. Therefore, it is increasingly difficult to efficiently filter relevant information from all the data available on the internet, which is a potential challenge for many Internet users [2]. One of the solutions to deal with the issue of information overload is the use of recommendation system techniques. This technique involves processing data to determine which items are necessary and important to users, as well as to better serve their needs [3]. This study becomes an important area of research after being launched in the middle of the 1990s [4], [5]. The study gives beneficial in providing recommendations in a wide range of applications such as music preferences, movies, news, books, social networks, research articles, products in general, and so on [6]-[8].

There are three primary categories of recommendation systems, namely content-based filtering (CBF), collaborative filtering (CF), and hybrid filtering [9], [10]. CBF technique is also called cognitive filtering [5]. It makes recommendations based on how closely related items relate to those that users liked previously [3], consequently, it also depends on the user-item profile and user profile. On the other hand, the CF technique recommends items for an active user that other users like based on their similarities, through their similar interests and tastes [11]. As for hybrid filtering technique, the technique mixes two or more different methods

to improve the performance of recommendations and to avoid some problems that the previous methods could not address [12].

In this paper, the focus is on the CF technique because it is the most widespread, popular, and effective technique in this field [13]. There are two categories into which it can be divided: model-based approaches and memory-based approaches [14]. Memory-based approaches attempt to find users similar to the active user by using all user rating data to generate the missing user rating for an item [15], [16]. Furthermore, this method is also based on how the data is processed, which can be divided into two main types: user-based and item-based. The user-based method makes missing rating predictions for the target user based on other similar users in rating preferences. On the other hand, the item-based method, unlike the user-based method, the prediction of the target user's rating depends on the similarity of the item [17]. For model-based approaches, it is based on creating a model that can describe user rating data based on a user rating matrix for a user prediction of unknown rating for a given item, such as Bayesian networks [18], clustering methods [19], and matrix factorization methods [20], [21]. In this paper, the model-based method is examined in-depth, as it handles the sparsity problem, which will be discussed later as this approach is better than memory-based approaches [22].

Recommendation systems suffer from many challenges such as the size of the database being processed, the cold start problem, the sparsity problem, and many others. The first challenge is these systems need powerful devices with high efficiency and strong algorithms to deal with big data. Another problem is cold start problem, which happens when a new item or a new user enters the system for the first time. Therefore, the recommendation system does not contain enough information about users or items for using in providing them with recommendations that the user prefers [23]. Finally, the last problem is sparsity occurs when the number of ratings already obtained by the user is too small in relation to the total of ratings that need to be predicted [24]. However, matrix factorization (MF) has effectively addressed the problems. MF model is the most widely used of the several collaborative filtering techniques, which reduce the sparsity issue found in the recommendation system database [25]. The purpose of MF is to reconstruct the user-item rating matrix as a set of two low dimensional matrices: one for the latent space matrix of the users and the other for the items, and then use the decomposition matrices to make additional predictions [26].

Deep learning is a subfield of machine learning that is based on methods that are inspired by how the human brain works and how it is constructed [27]. In recent years, deep learning has become popular in general, due to the development in increasing computing power and processing big data used in model training, and it has also been used and increasingly in recommendation systems. The reason is the deep learning models offer its superior performance in reducing data dimensions, extracting better features, and renewing data. Furthermore, It has also been used successfully in natural language processing and computer vision [28]. Many deep learning models have proven successful in areas of recommendation systems such as hybrid collaborative recommendation via semi-autoencoder (HRSA) [29], stacked denoising auto-encoders neural networks (SDAE) [30], restricted boltzmann machine (RBM) based CF [31], information retrieval generative adversarial network (IRGAN) [32], recurrent neural network (RNN) based CF [33], and others.

A feed-forward neural network called an autoencoder has three layers: input, hidden, and output. It is one of the deep learning techniques. It is widely used with recommendation systems because of its excellent efficiency in dimensionality reduction, latent feature extraction, and data reconstruction. As a result, it has effectively alleviated the issues of cold start and data sparsity in recommendation systems [12]. Ouyang *et al.* [34] a model called autoencoder-based CF (ACF) is proposed, in which autoencoder is used with CF to learn a complex nonlinear representation of users' explicit ratings on an item. Also, in [35] a model called AutoRec is proposed, which is a rating prediction model based on the autoencoder network structure. AutoRec has been very successful in rating user-item matrix prediction. However, there are still major gaps in the most recent autoencoder-based techniques used in recommendation systems. In most autoencoder-based techniques, the input layer and output layer dimensions must be the same, but some other techniques need to combine side information with the input layer to better understand the characteristics of users and items. Therefore, it leads to better accuracy of recommendations. As a result, the input layer's dimensions are larger than those of the output layer, resulting in a technique called semi-autoencoder. For example, [29] developed the semi-autoencoder and integrated it with a hybrid CF technique with the goal of rating and ranking prediction. To make personalized recommendations, it uses both content data and learning non-linear features.

In recent research, the effectiveness of semi-autoencoder in recommendation systems has been demonstrated. It alleviates the cold start and sparsity problems by integrating additional information for items and users, as well as their ratings. To alleviate the above problems and increase the accuracy of the recommendation system, we propose an improved hybrid semi-stacked autoencoder for item-features of recommendation system (iHSARS) framework. This proposed method is aimed to improve the performance of the HRSA technique used in rating prediction through several improvements to it. First, to increase the side information sources to the input layer, based on the information extracted from the relationship between the user-item matrix and user profiles, which will be explained later. This method is taken from the user-item matrix in order to capture the hidden

relationship between them efficiently. Second, the framework's hidden layer number can be increased to improve latent factor learning. Finally, to update the reconstruction loss function equation by adjusting the regularization term by adding  $l_2$  norm for all weights and biases, in order to avoid the problem of overfitting.

We investigate the efficacy of our proposed framework using two global datasets, Movielens-1M and Movielens-100K, that differ in size and density. The performance of this proposal has been compared with other well-known techniques with two evaluation metrics, mean absolute error (MAE) and root mean square error (RMSE). They are most commonly used metrics for evaluating the performance of models in the recommendation systems. The following are some of the contributions made by this paper:

- We propose the iHSARS Framework as a method of recommendation. This proposed method has some improvement features in a way to the performance of the HRSA rating prediction technique. One of the ways is increasing the side information sources of the input layer based on the information extracted from the relationship between the user-item matrix and user profiles. Moreover, the number of hidden layers in the framework is increased, and the latent factors are better learned. Finally, update the reconstruction loss function equation by adjusting the regularization term to avoid the problem of overfitting.
- Our proposed framework is also contributing the performance of the recommendation system and alleviating the problems of sparsity and cold start by increasing the sources of side information.
- The efficiency of the proposed iHSARS is demonstrated by comprehensive experiments on two global data sets, which differ in size and density.

The rest of this paper is structured as follows: In section 2, it highlights the overview of other studies that related our work. In section 3, it presents and explains the research frameworks in detail. Section 4 shows the results of experiments conducted on two data sets and the following section discusses and analyses the experimental results obtained in section 5. Finally, section 6 presents the paper's conclusion.

## 2. RELATED WORKS

Early CF approaches assume that a group of users with identical interests have similar interests. Memory-based approaches and model-based approaches are the two groups into which they are divided. Memory-based methods are not very effective since they require access to all the set's ratings. Moreover, their performance is unsatisfactory if the ratings are very sparse. Therefore, model-based methods have been suggested to solve these challenges. The basic principle behind model-based methods is to build a model that represents (user-item) interactions through factors that represent users' and items' latent features. There are many traditional model-based collaborative filtering methods. Matrix factorization approaches such as probabilistic matrix factorization [36] and alternating least squares (ALS) [26].

However, the vast amount of content, its complexity, and dynamics seem to present a challenge for most recommendation systems. To overcome these challenges, a numerous researchers have focused on incorporating deep learning methods into recommendation systems to improve them. In the field of CF, deep learning gained relatively little attention initially. Among the presented works, RBMs (restricted Boltzmann machines) were proposed by [31] to model (user-item) rating matrix. However, there was very little published work after that. Nevertheless, deep learning method has achieved great success in image and voice recognition [37] and it still has a long way to go in many areas. Deep learning can discover complex non-linear hidden features in heterogeneous data, as well as greatly alleviate the problem of sparse data. As a result, it is a promising technique used for recommendation systems. In addition, deep learning of recommendation systems is currently a hot topic of research.

Autoencoders have gained popularity in recent years as a popular architecture for recommendation systems. Tahmasebi *et al.* [38] presented a case for a social movie recommendation system called social recommender deep autoencoder network (SRDNet) that integrates the advantages of collaborative filtering with content-based methods to generate more accurate recommendations. Additionally, a deep autoencoder is employed to discover users' latent characteristics focused on social relationships gathered from the Open Movie and MovieTweatings datasets. Yu *et al.* [14] the Model-based collaborate filtering algorithm based on stacked autoencoder (MCFSAE) algorithm is introduced in order to deal with the sparsity issue in CF recommendation systems, where high-level features are extracted by stacked autoencoders (SAE), which are then used by the softmax layer to perform classification and estimate the missing ratings. There are many studies related to recommendation systems that have been carried out which take advantage of the autoencoder's capacity to learn the representations in low-dimensional spaces and thus become more compressed and efficient representations such as AutoRec [35], AutoSVD++ [39], and collaborative filtering neural network (CFN) [37].

Autoencoder (AE) is an unsupervised learning model that receives a rating data vector as input and tries to match the output vector to the input vector, which in the output layer is not labeled. AE is split into two sections. The first is an encoder that reduces the high-dimensions of the input layer into a lower-dimensions in the bottleneck layer. Another section is the decoder layer that reconstructs the input layer from the bottleneck layer. The input layer and output layer dimensions should be the same in most methods for autoencoder-based

recommendation systems. However, in order to better understand the features of items and users, there are many other methods that integrate side information with the input layer simultaneously, which leads to an improvement in the accuracy of recommendations. One of these methods is called semi-autoencoder, in which the input layer's dimensions are larger than those of the output layer. HSAR model is an example of this type of model [29]. To address the sparsity problem, this model integrates additional features of items and users, as well as their ratings. When compared with autoencoders, semi-autoencoders have two main advantages: First, by sampling multiple subsets of the inputs, representations of diverse features can be captured and reconstructed effectively, and secondly, semi-autoencoders have the function of integrating additional information into the input layer easily.

Several techniques to achieving goals similar to those discussed in this section can be found in the literature. As an example, [40] proposed a recommendation via dual-autoencoder (ReDa) model for learning representation and recommendation, Which uses the dual architecture of a traditional autoencoder called "Dual-Autoencoder". One is used to find out representations of hidden features for users, and the other is used to find out hidden feature representations of items simultaneously. The ReDa has some drawbacks, such as the input layer's dimensions being the same as the output layer's dimensions. Thus, if some additional information from different sources is used in this model, it is added to the input layer, which increases the input layer's dimensions and, in consequence, increases the output layer's dimensions. This will lead to important problems [41]. Initially, when additional information from different sources is combined into the input layer, reconstruction this information in the output layer results in information loss. Second, as additional information is provided from other sources, the model is difficult to expand. As in semi-autoencoder model.

To overcome the disadvantages of autoencoder integrating the input layer's side information without affecting the output layer's dimensions, [29] developed a hybrid CF approach for rating prediction based on the semi-autoencoder structure. In this model, it overcomes the sparsity problem by integrating additional information for items and users with their user-item rating matrix. Additionally, it helps in learning the improved hidden representation of the recommendation. There are some shortcomings in some aspects of this model, which will be addressed in the proposed framework of this article and that lead to an increase in its performance and an improvement in scalability. First, there is little additional information used in this model which leads to poor capturing of feature representations of the items efficiently. Thus, in the proposed framework, new side information is added to the previous side information in the modified model, which is extracted from the relationship between the user-item matrix and user profiles which will be detailed later. Second, this model has only three layers, namely, the input layer, the hidden layer, and the output layer, which compresses the high dimensions of the input data in the input layer to the low dimensions of the hidden layer. Thus, having a single hidden layer does not reduce dimensions sufficiently and effectively [42]. Therefore, in our proposed framework, additional hidden layers have been added to further reduce the dimensions and better optimize the weights of the neural network [34], [43].

Finally, the proposed iHSARS model can increase side information sources that increase the accuracy of the HRSA model's performance. In conjunction with the addition of many hidden layers that adjust the weights in the network while training the model with better accuracy. It is conceivable to use this technique in the future to improve the efficiency of other successful models.

### 3. RESEARCH FRAMEWORKS

#### 3.1. Preliminaries and notations

In this subsection, we first introduce some often-used notations, as shown in Table 1. The notations consist of some preliminary knowledge that will be applied to our suggested framework. These notations are helpful for understanding the formulas that are be used in this paper.

#### 3.2. Problem definition

The rating prediction issue in a recommendation system is defined as follows. Given the user-item rating matrix  $R \in R^{M \times N}$ , where  $M$  and  $N$  are the numbers of users and items, respectively, and  $r^{ui} \in R$  represents the rating given by user  $u \in \{1, \dots, M\}$  to item  $i \in \{1, \dots, N\}$ . In this framework,  $r^i = \{r^{1i}, \dots, r^{Mi}\}$  represents the rating vector of the observed partial item  $i$ , which corresponds to the rating-matrix columns, to represent each item  $i$ . For ease of use, we denote to the partial observed vectors for all items as  $r^I \in R^{N \times M}$ . The attributes information vector of item  $i$  are represented by the formula  $a^i \in R^{P_i}$ , whereas the attributes information vector for all items is referred to as  $A^I \in R^{N \times P_i}$ . The (item-user) characteristics information vector of item  $i$  are represented by the formula  $c^i \in R^{Q_i}$ , whereas the characteristics information vector for all items is referred to as  $C^I \in R^{N \times Q_i}$ . The ratings are typically expressed in either explicit integers with a range of (1 to 5) stars or implicit binary numbers (0, 1), where 0 indicates dislike and 1 indicates like. Finally, the set of observed ratings is denoted by  $\Omega$ .

Table 1. Notations and their descriptions used in this article

Notations	Definitions	Notations	Definitions
$R$	The rating matrix $R \in R^{m \times n}$	$Q_i$	The number of item characteristics information
$\hat{R}$	The prediction matrix $\hat{R} \in R^{m \times n}$	$a^i$	The additional attributes of item $i$
$M$	Number of users	$A^I$	All items' additional attributes
$N$	Number of items	$c^i$	The additional characteristics of item $i$
$r^{ui}$	The rating given by user $u$ to item $i$	$C^I$	All items' additional characteristics
$\hat{r}^{ui}$	The predicted value of $r^{ui}$	$k$	Number of hidden layers
$r^i$	The rating matrix's column	$W, \hat{W}, W_1$ to $W_{k+1}$	Matrices for weights in neural networks
$r^I$	All items' partial observed vectors	$b, \hat{b}, b_1$ to $b_{k+1}$	Neural network bias terms
$\hat{r}^i$	The prediction rating matrix's column	$x$	Raw data set for input
$\hat{r}^I$	The prediction rating vectors for all items	$\hat{x}$	Output dataset restructuring
$\Omega$	Observed set of ratings	$h$	The item's latent representation
$P_i$	The number of item attributes information	$sub(x)$	Raw data set for input without additional side information

### 3.3. Method

In this subsection, it provides structure and details about our proposed iHSARS framework, which enhances the performance of the recommendation system in terms of rating prediction and alleviating the problem of sparsity. Figure 1 provides an illustration of the entire framework. The proposed iHSARS framework, as shown in Figure 1, uses two types of side information at the input layer: item attributes and (item-user) characteristics, which will be introduced in detail in this subsection. In addition, the network is trained using  $k$  hidden layers, and  $k$  must be an odd number so that the number of hidden layers from the right and left of the middle layer is equal.

Our framework is provided with side information by two parts, one item attributes and the other are (item-user) characteristics. First, the item attributes vector  $i$  is denoted by the notation  $a^i \in R^{P_i}$  ( $i = 1, \dots, N$ ), whereas  $A^I \in R^{N \times P_i}$  refers to the vector of attribute information for all items. For instance, if the dataset being used is a movie, the item attributes can be movie genre and year. Second, the (item-user) characteristics vector  $i$  is denoted by the notation  $c^i \in R^{Q_i}$  ( $i = 1, \dots, N$ ), whereas  $C^I \in R^{N \times Q_i}$  refers to (item-user) characteristics information vector for all items. The (item-user) characteristics of a side information vector it has three vectors. The first vector consists of the ratio of the number of users with the same attribute features in a user profile who rated the same item to the number of all users who rated the same item. For example, such as the age attribute vector and the occupation attribute vector for users in the movie dataset. The equation for calculating one of them, the occupation attribute vector, will be illustrated. The occupation attribute vector of all users is computed by the ratio of all users who rated the same movie and who have the same occupation to all users who rated the same movie, regardless of their occupations. The second vector contains one value, which is the ratio of the total number of users who rated the item to the total number of all users in the data set. Finally, the final vector also has one value, which is the average of all users' ratings for the same item. The item's partial rating vector for all  $N$  items  $r^I$ , the item attribute vector  $A^I$  for all  $N$  items, and the (item-user) characteristics vector for all  $N$  items  $C^I$  are combined to represent concatenated vectors  $cat(r^I; A^I; C^I) \in R^{N \times (M+P_i+Q_i)}$  as an input to the input layer of the network. In (1)-(3) represent the encoding procedure for a network.

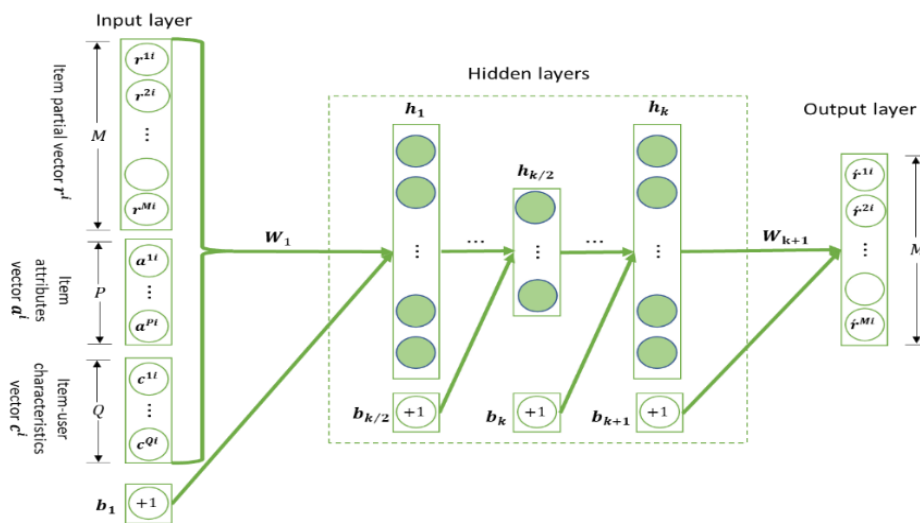


Figure 1. The proposed iHSARS framework structure

$$h_1 = g(\text{cat}(r^I; A^I; C^I) \cdot W_1 + b_1) \quad (1)$$

$$h_j = g(h_{j-1} \cdot W_j + b_j) \quad (2)$$

$$h_{k/2} = g(h_{(k/2)-1} \cdot W_{k/2} + b_{k/2}) \quad (3)$$

Where  $W_1$  to  $W_{k/2}$  are weight matrices and  $k/2$  is rounded to the nearest higher integer value,  $b_j$  is a representation of the  $j$ -th hidden layer's bias vector,  $h_j$  denotes the input's hidden latent representation at the  $j$ -th hidden layer,  $h_{k/2}$  is an item's latent representation, and where  $g$  is an active function such as Identity. Using (4)-(6). The latent representation  $h_{k/2}$  is then mapped using the decoding mapping to a reconstruction  $\hat{r}^I$ :

$$h_{(k/2)+1} = f(h_{k/2} \cdot W_{(k/2)+1} + b_{(k/2)+1}) \quad (4)$$

$$h_j = f(h_{j-1} \cdot W_j + b_j) \quad (5)$$

$$\hat{r}^I = f(h_k \cdot W_{k+1} + b_{k+1}) \quad (6)$$

where  $W_{(k/2)+1}$  to  $W_k$  are weight matrices  $k/2$  is rounded to the nearest higher integer value,  $b_j$  is a representation of the  $j$ -th hidden layer's bias vector,  $h_j$  denotes the input's hidden latent representation at the  $j$ -th hidden layer,  $\hat{r}^I$  is the reconstruction of the input from the latent hidden representation  $h_{k/2}$ , and where  $f$  is an active function such as sigmoid. The objective function of the network is to reconstruct the input data  $\text{cat}(r^i; a^i; c^i)$  in order to minimize the difference between the output and the subset of the input, where  $\text{sub}(x) = r^i$ , and minimizing the gaps between reconstruction  $\hat{r}^i$  and  $r^i$ , as given in (7).

$$\mathcal{L}(r^i, \hat{r}^i) = \min_{W_1, \dots, W_{k+1}, b_1, \dots, b_{k+1}, r^{ui} \in \Omega} \frac{1}{N} \sum_{i=1}^N \|\hat{r}^i - r^i\|_2^2 + \frac{\lambda}{2} (\|W_1\|_2^2 + \dots + \|W_{k+1}\|_2^2) \quad (7)$$

The proposed framework for optimizing the parameters of (7) uses stochastic gradient descent (SGD). Our experiments have shown that Adam's method is better than others because of its rapid convergence. The proposed iHSARS algorithm's pseudo-codes are summarized in Algorithm 1.

**Algorithm 1. The iHSARS Algorithm**

**Input:** The rating matrix  $R$ , total number of hidden layers  $k$ , dimensions of the item's vector attribute  $P_i$ , dimensions of the (item-user) characteristics vector  $Q_i$ , parameter  $\lambda$ .

**Output:** The prediction matrix  $\hat{R}$ .

1. Get the item's attribute information vector ( $a^i \in R^{P_i}$ ).
2. Get the  $c^i \in R^{Q_i}$  characteristic of the information vector (item-user) for each item.
3. Get the concatenation vectors  $\text{cat}(r^I; A^I; C^I)$  for the item's rating vector  $r^I$ , the item attribute vector  $A^I$ , and the (item-user) characteristic vector  $C^I$ ;
4. Set  $b_1$  to  $b_{k+1}$  vectors to 0 and initialize  $W_1$  to  $W_{k+1}$  vectors by truncating a normal-distributed random number.
5. Input  $\text{cat}(r^I; A^I; C^I)$  to iHSARS network and then train the network by calculating the values using Eq. (1) to Eq. (6);
6. Using an Adam stochastic gradient descent optimization, minimize Eq. (7) until convergence;
7. Return: The predicated rating matrix  $\hat{R}$

## 4. EXPERIMENTS

In our experiments, we used two open real-world datasets of various sizes and densities to systematically evaluate the performance of the iHSARS framework for recommendations. We use MovieLens as dataset with two different sizes. The first dataset consists of 100K data, which is called as MovieLens-100K, while the second dataset consists of 1M data, which is called as MovieLens-1M.

### 4.1. Datasets

Both datasets are used to compare the proposed framework with the others. In the first dataset, it has 1,682 movies and 943 users with a 100,000 rating and a 6.3% rating density, while in the second dataset, there are 6,040 users and 3,706 movies with a rating of more than one million and a rating density of 4.468%. In both, the minimum number of movies each user must rate from (1 to 5) in the data set is 20, which indicates that the higher the value of the score, the more the user wants to enjoy watching the movie; 1 indicates dislike, and 5 indicates really like it. Additionally, they contain side information about items and users that will be included in our experiment. It includes user attributes such as occupation and age while also including item information such as year of release and genre.

### 4.2. Metrics for evaluation

In our experiment, we evaluate the efficiency of our suggested framework using two commonly used metrics: MAE [34] and RMSE [37]. The accuracy of predicted ratings is evaluated using these metrics. MAE calculates the absolute error across all N pairs by determining the gap between the predicted values of  $\hat{r}^i$  and their actual ratings of  $r^i$ . When compared to MAE, RMSE gives predictions with larger errors more weight. Thus, as the MAE and RMSE values decrease, the performance of the method also improves. The MAE and RMSE of the framework are calculated as given in (8) and (9) respectively.

$$MAE = \frac{1}{N} \sum_1^N |r^i - \hat{r}^i| \tag{8}$$

$$RMSE = \sqrt{\frac{1}{N} \sum_1^N (r^i - \hat{r}^i)^2} \tag{9}$$

### 4.3. Evaluation results

We evaluate our framework using different training ratios, randomly selecting 50%, 60%, 70%, 80%, and 90% of the data used as training datasets, respectively, leaving the rest as test dataset. With  $k = 3$ , the number of hidden layers is determined,  $k_1, k_2, k_3$  contain (700, 400, 700) hidden neurons respectively. To achieve the desired result, the performance is compared using a variety of parameters, including the specific learning rate  $\eta$  to  $10^{-4}$ , optimization algorithm as Adam, the batch size to 2,000, the epochs to 400, the activation function identity and sigmoid, which map encoding and decoding, respectively. The proposed network weight matrix is randomly generated from the mean and standard deviation's normal distribution, and the bias vector's initial value is set to 0. To extract the results, we implemented our framework's Python code on Google Colab. The effectiveness of the proposed framework is evaluated using RMSE and MAE average results. Tables 2 and 3 list all the results from the two datasets on RMSE and MAE. The method performs better the smaller the MAE and RMSE values are.

Table 2. Average (MAE and RMSE) for Movielens-100k with 50%, 60, 70%, 80%, and 90% of the training data

Methods	MAE					RMSE				
	50%	60%	70%	80%	90%	50%	60%	70%	80%	90%
BPMF [44]	N/A	0.9650 ±0.0157	0.9040 ±0.0048	0.8810 ±0.0094	0.8626 ±0.0129	N/A	1.2319 ±0.0220	1.1532 ±0.0068	1.1274 ±0.0118	1.1032 ±0.0162
PMF [36]	N/A	0.8395 ±0.0822	0.7919 ±0.0407	0.7823 ±0.0228	0.7882 ±0.0280	N/A	1.0253 ±0.0780	0.9787 ±0.0403	0.9701 ±0.0266	0.9750 ±0.0304
PRA [45]	N/A	0.7656 ±0.0031	0.7632 ±0.0027	0.7594 ±0.0041	0.7595 ±0.0040	N/A	0.9753 ±0.0039	0.9710 ±0.0058	0.9657 ±0.0042	0.9649 ±0.0052
SVD++ [46]	N/A	0.75220 ±0.0006	0.7400 ±0.0005	0.7260 ±0.0005	0.7222 ±0.0021	N/A	0.9652 ±0.0001	0.9502 ±0.0006	0.9318 ±0.0009	0.9240 ±0.0005
HRSA [29]	0.73289 ±0.002	0.72219 ±0.002	0.71457 ±0.003	0.70409 ±0.004	0.69976 ±0.006	0.92659 ±0.002	0.91709 ±0.002	0.90690 ±0.004	0.896 ±0.003	0.890 ±0.007
ReDa [40]	N/A	0.7332 ±0.0047	0.7248 ±0.0067	0.7203 ±0.0043	0.7153 ±0.0094	N/A	0.9329 ±0.0053	0.9231 ±0.0081	0.9190 ±0.0056	0.9114 ±0.0093
<b>iHSARS</b>	<b>0.71708</b> ±0.001	<b>0.71272</b> ±0.001	<b>0.70173</b> ±0.001	<b>0.69405</b> ±0.001	<b>0.68320</b> ±0.001	<b>0.91679</b> ±0.001	<b>0.90940</b> ±0.001	<b>0.89559</b> ±0.001	<b>0.88714</b> ±0.001	<b>0.87249</b> ±0.001

Table 3. Average (MAE and RMSE) for Movielens-1M with 50%, 60, 70%, 80%, and 90% of the training data

Methods	MAE					RMSE				
	50%	60%	70%	80%	90%	50%	60%	70%	80%	90%
MCFSAE [14]	N/A	0.7820	0.7700	0.7730	0.7600	N/A	1.1230	1.1000	1.0770	1.0490
BPMF [44]	N/A	0.6962 ±0.0001	0.6882 ±0.0001	0.6802 ±0.0001	0.6758 ±0.0009	N/A	0.8946 ±0.0003	0.8832 ±0.0001	0.8738 ±0.0001	0.8688 ±0.0003
PMF [36]	N/A	0.7055 ±0.0035	0.6971 ±0.0020	0.6900 ±0.0019	0.6849 ±0.0031	N/A	0.8998 ±0.0038	0.8891 ±0.0028	0.8805 ±0.0027	0.8748 ±0.0061
PRA [45]	N/A	0.7108 ±0.0010	0.7122 ±0.0005	0.7142 ±0.0001	0.7155 ±0.0001	N/A	0.9002 ±0.0010	0.9026 ±0.0005	0.9053 ±0.0002	0.9071 ±0.0001
SVD++ [46]	N/A	0.6782 ±0.0001	0.6736 ±0.0003	0.6680 ±0.0005	0.6656 ±0.0008	N/A	0.8656 ±0.0003	0.8586 ±0.0003	0.8508 ±0.0004	0.8478 ±0.0019
HRSA [29]	0.69400 ±0.001	0.68687 ±0.001	0.67935 ±0.001	0.67285 ±0.001	0.67180 ±0.001	0.88200 ±0.002	0.87413 ±0.001	0.86643 ±0.001	0.85855 ±0.001	0.85643 ±0.001
ReDa [40]	N/A	0.6789 ±0.0040	0.6731 ±0.0029	0.6646 ±0.0029	0.6647 ±0.0051	N/A	0.8659 ±0.0042	0.8573 ±0.0033	0.8485 ±0.0024	0.8474 ±0.0046
<b>iHSARS</b>	<b>0.67971</b> ±0.001	<b>0.67135</b> ±0.001	<b>0.66600</b> ±0.001	<b>0.66154</b> ±0.001	<b>0.65735</b> ±0.001	<b>0.86983</b> ±0.001	<b>0.85831</b> ±0.001	<b>0.85306</b> ±0.001	<b>0.84700</b> ±0.001	<b>0.84115</b> ±0.001

### 5. DISCUSSION OF FINDINGS

To highlight the efficiency of our proposed framework, we compared it to the traditional matrix factorization techniques and autoencoder-based techniques mentioned below:

- BPMF: Bayesian probabilistic matrix factorization [44], which combines the Bayesian method and probabilistic matrix factorization method to provide the best recommendation.
- PMF: Probabilistic matrix factorization [36], which factors the user-item matrix using matrix factorization in order to discover the hidden features of users and items.
- PRA: Probabilistic rating autoencoder [45], which creates latent user feature profiles using autoencoder.
- SVD++: Singular value decomposition++ [46] is a composite method that integrates the neighborhood model and the latent factor model into a single model, making use of explicit and implicit user feedback.
- MCFSAE: A model-based collaborate filtering algorithm based on stacked autoencoder [14] is introduced in order to deal with the sparsity issue in CF recommendation systems, where high-level features are extracted by stacked autoencoders (SAE), which are then used by the softmax layer to perform classification and estimate the missing ratings.
- HRSA [29] has developed a hybrid CF technique for recommendations based on a semi-autoencoder. This technique obtains the prediction matrix directly from the semi-autoencoder model's output layer. ReDa [40], which uses a dual-autoencoder, to learn hidden representations of both users and items at the same time.

The averages (RMSE and MAE) of iHSARS are shown in Tables 2 and 3 along with comparisons between models trained on the Movielens-100k and Movielens-1M data sets using different ratios of the training data. Their experimental results for BPMF, PMF, PRA, SVD++, MCFSAE, and ReDa were gathered from published paper [40], whereas the experimental results for HRSA obtained from our implementation of the source code. Typically, all techniques work better as the percentage of training data samples increases. We can clearly see from the experimental results that our framework outperforms all other comparison techniques in respect of rating prediction, especially when the data set is sparse. This demonstrates the ability of our framework to learn the robust features of the items. However, on 70%, 80%, and 90% of the 1M dataset, our framework slightly outperformed the ReDa technique regarding the RMSE and MAE evaluation criteria, as well as with SVD++ at 90%. Comparing our framework with HRSA for rating prediction, the proposed framework is much better than HRSA. This is because it has limited additional information is used in HRSA while more additional information in the proposed framework is extracted from item attributes and user characteristics and added to previous information in HRSA. In addition, the HRSA network has been developed by adding many hidden layers to be better trained. The graph in Figures 2-5 intuitively shows that our iHSARS framework significantly outperforms other comparison techniques in the two data sets.

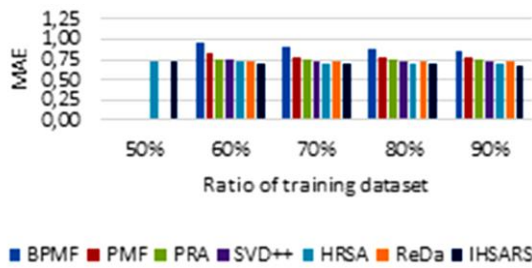


Figure 2. MAE graph of Movielens-100K training data

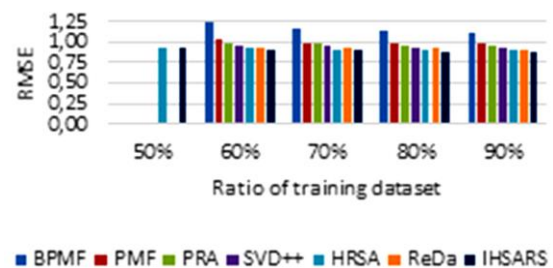


Figure 3. RMSE graph of Movielens-100K training data

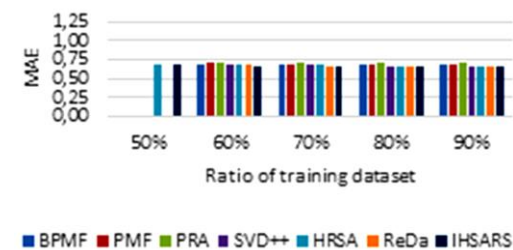


Figure 4. MAE graph of Movielens-1M training data

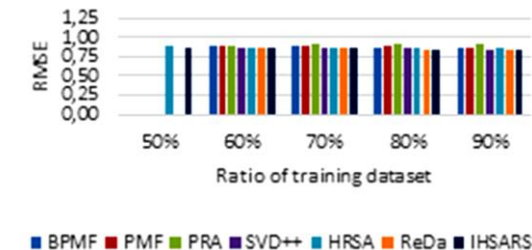


Figure 5. RMSE graph of Movielens-1M training data



## 6. CONCLUSION

As a conclusion, in this work, the iHSARS framework is proposed which is based on the development and improvement of the performance of HRSAs technique. The framework can improve the prediction results for recommendation for the issue of data sparsity. The improvements that have been made come in two ways. First, we have increased the additional information sources for the items, while the second improvement is the increased number of hidden layers of the network for better training. In this study too, an experimental investigation has been conducted utilizing two real-world datasets with various percentages of the training data to demonstrate the effectiveness of our framework. Evaluations and comparisons of iHSARS with the most popular techniques indicate that the proposed framework reduces both RMSE and MAE. For future works, more experiments will be conducted to increase the performance in terms of accuracy, such as incorporating more explicit or implicit auxiliary information for items and users into the framework. In addition, the experiments will be run to solve complexity problems. Besides that, the experiments will be conducted on huge datasets such as Movielens-10M, Movielens-20M, and Movielens-25M.

## ACKNOWLEDGEMENT

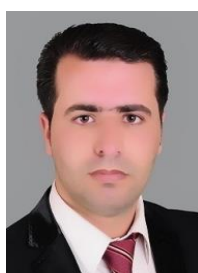
We would like to thank the Research Management Office, Universiti Malaysia Terengganu and UMTAS2021 for providing us financial support.




## REFERENCES

- [1] B. Dong, Y. Zhu, L. Li, and X. Wu, "Hybrid collaborative recommendation via dual-autoencoder," *IEEE Access*, vol. 8, pp. 46030–46040, 2020, doi: 10.1109/ACCESS.2020.2979255.
- [2] F. O. Isinkaye, Y. O. Folajimi, and B. A. Ojokoh, "Recommendation systems: Principles, methods and evaluation," *Egypt. Informatics J.*, vol. 16, no. 3, pp. 261–273, 2015, doi: 10.1016/j.eij.2015.06.005.
- [3] D. K. Behera, M. Das, S. Swetanisha, and B. Naik, "Collaborative filtering using restricted boltzmann machine and fuzzy c-means," *In Progress in Computing, Analytics and Networking*, vol. 710, pp. 723–731, 2018.
- [4] W. Hill, L. Stead, M. Rosenstein, and G. Furnas, "Recommending and evaluating choices in a virtual community of use," *Conf. Hum. Factors Comput. Syst. - Proc.*, vol. 1, pp. 194–201, 1995, doi: 10.1145/223904.223929.
- [5] L. Shah, H. Gaudani, and P. Balani, "Survey on recommendation system," *Int. J. Comput. Appl.*, vol. 137, no. 7, pp. 43–49, 2016, doi: 10.5120/ijca2016908821.
- [6] M. K. Najafabadi, A. H. Mohamed, and M. N. Mahrin, "A survey on data mining techniques in recommender systems," *Soft Comput.*, vol. 23, no. 2, pp. 627–654, 2019, doi: 10.1007/s00500-017-2918-7.
- [7] Z. Batmaz, A. Yurekli, A. Bilge, and C. Kaleli, "A review on deep learning for recommender systems: challenges and remedies," *Artif. Intell. Rev.*, vol. 52, no. 1, pp. 1–37, Jun. 2019, doi: 10.1007/s10462-018-9654-y.
- [8] C. Panagiotakis, H. Papadakis, A. Papagrorgiou, and P. Fragopoulou, "Improving recommender systems via a dual training error based correction approach[Formula presented]," *Expert Syst. Appl.*, vol. 183, 2020, p. 115386, 2021, doi: 10.1016/j.eswa.2021.115386.
- [9] H. Zhang, R. K. Wong, and V. W. Chu, "Hybrid variational autoencoder for recommender systems," *ACM Trans. Knowl. Discov. Data*, vol. 16, no. 2, 2022, doi: 10.1145/3470659.
- [10] M. Nilashi, O. Ibrahim, and K. Bagherifard, "A recommender system based on collaborative filtering using ontology and dimensionality reduction techniques," *Expert Syst. Appl.*, vol. 92, pp. 507–520, 2018, doi: 10.1016/j.eswa.2017.09.058.
- [11] A. A. Sbou and N. H. Abd Rahim, "Performance comparison of three different types of autoencoders using recommendation systems," *J. Theor. Appl. Inf. Technol.*, vol. 100, no. 5, pp. 1675–1683, 2022.
- [12] 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, doi: 10.1007/s11704-018-8052-6.
- [13] M. H. Chen, C. H. Teng, and P. C. Chang, "Applying artificial immune systems to collaborative filtering for movie recommendation," *Adv. Eng. Informatics*, vol. 29, no. 4, pp. 830–839, 2015, doi: 10.1016/j.aei.2015.04.005.
- [14] 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, doi: 10.1007/s00521-021-05933-8.
- [15] N. Idrissi and A. Zellou, "A systematic literature review of sparsity issues in recommender systems," *Soc. Netw. Anal. Min.*, vol. 10, no. 1, 2020, doi: 10.1007/s13278-020-0626-2.
- [16] B. T. Betru, C. A. Onana, and B. Bernabe, "Deep learning methods on recommender system: A survey of state-of-the-art," *Int. J. Comput. Appl.*, vol. 162, no. 10, pp. 975–8887, 2017, doi: 10.5120/ijca2017913361.
- [17] S. K. Raghuvanshi and R. K. Pateriya, "Collaborative filtering techniques in recommendation systems," in *Data, Engineering and Applications*, Singapore: Springer Singapore, pp. 11–21, 2019.
- [18] L. M. De Campos, J. M. Fernández-Luna, J. F. Huete, and M. A. Rueda-Morales, "Combining content-based and collaborative recommendations: A hybrid approach based on Bayesian networks," *Int. J. Approx. Reason.*, vol. 51, no. 7, pp. 785–799, 2010, doi: 10.1016/j.ijar.2010.04.001.
- [19] M. Eirinaki, C. Lamos, S. Paulakis, and M. Vazirgiannis, "Web personalization integrating content semantics and navigational patterns," *Proc. International Work. Web Inf. Data Manag.*, pp. 72–79, 2004, doi: 10.1145/1031453.1031468.
- [20] D. Billsus, D. Billsus, M. J. Pazzani, and M. J. Pazzani, "Learning collaborative information filters," *Proc. Fifteenth Int. Conf. Mach. Learn.*, vol. 54, p. 47, 1998, [Online]. Available: <http://www.aaai.org/Papers/Workshops/1998/WS-98-08/WS98-08-005.pdf>.
- [21] M. Ranjbar, P. Moradi, M. Azami, and M. Jalili, "An imputation-based matrix factorization method for improving accuracy of collaborative filtering systems," *Eng. Appl. Artif. Intell.*, vol. 46, pp. 58–66, 2015, doi: 10.1016/j.engappai.2015.08.010.
- [22] R. Katarya, "Movie recommender system with metaheuristic artificial bee," *Neural Comput. Appl.*, vol. 30, no. 6, pp. 1983–1990, 2018, doi: 10.1007/s00521-017-3338-4.
- [23] B. Lika, K. Kolomvatsos, and S. Hadjiefthymiades, "Facing the cold start problem in recommender systems," *Expert Syst. Appl.*, vol. 41, no. 4 PART 2, pp. 2065–2073, 2014, doi: 10.1016/j.eswa.2013.09.005.
- [24] D. Bokde, S. Girase, and D. Mukhopadhyay, "Matrix factorization model in collaborative filtering algorithms: A survey," *Procedia Comput. Sci.*, vol. 49, no. 1, pp. 136–146, 2015, doi: 10.1016/j.procs.2015.04.237.




- [25] A. Pujahari and D. S. Sisodia, "Pair-wise preference relation based probabilistic matrix factorization for collaborative filtering in recommender system," *Knowledge-Based Syst.*, vol. 196, no. xxxx, p. 105798, 2020, doi: 10.1016/j.knsys.2020.105798.
- [26] Y. Koren, R. Bell, and C. Volinsky, "Matrix factorization techniques for recommender systems," *Computer (Long. Beach. Calif.)*, vol. 42, no. 8, pp. 30–37, Aug. 2009, doi: 10.1109/MC.2009.263.
- [27] A. Bashar, "Survey on evolving deep learning neural network," *In Progress in Computing, Analytics and Networking*, no. July, 2020, doi: 10.36548/jaicn.2019.2.003.
- [28] N. Sachdeva, E. Ritacco, G. Manco, and V. Pudi, "Sequential variational autoencoders for collaborative filtering," *WSDM 2019 - Proc. 12th ACM Int. Conf. Web Search Data Min.*, pp. 600–605, 2019, doi: 10.1145/3289600.3291007.
- [29] S. Zhang, L. Yao, X. Xu, S. Wang, and L. Zhu, "Hybrid collaborative recommendation via semi-autoencoder," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 10634 LNCS, pp. 185–193, 2017, doi: 10.1007/978-3-319-70087-8\_20.
- [30] F. Strub *et al.*, "Collaborative filtering with stacked denoising autoencoders and sparse inputs," *NIPS Work. Mach. Learn. eCommerce, Dec 2015*, pp. 1–9, 2015.
- [31] R. Salakhutdinov, A. Mnih, and G. Hinton, "Restricted boltzmann machines for collaborative filtering," in *Proceedings of the 24th international conference on Machine learning - ICML '07*, vol. 10, no. 5, 2007, pp. 791–798, doi: 10.1145/1273496.1273596.
- [32] J. Wang *et al.*, "IRGAN: A minimax game for unifying generative and discriminative information retrieval models," in *Proceedings of the 40th International ACM SIGIR conference on Research and Development in Information Retrieval*, pp. 515–524, 2017, doi: 10.1145/3077136.3080786.
- [33] C. Wu, J. Wang, J. Liu, and W. Liu, "Recurrent neural network based recommendation for time heterogeneous feedback," *Knowledge-Based Syst.*, vol. 109, pp. 90–103, 2016, doi: 10.1016/j.knsys.2016.06.028.
- [34] Y. Ouyang *et al.*, "Autoencoder-based collaborative filtering," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 8836, pp. 284–291, 2014, doi: 10.1007/978-3-319-12643-2\_35.
- [35] S. Sedhain, A. K. Menon, S. Sanner, and L. Xie, "AutoRec: Autoencoders meet collaborative filtering," in *Proceedings of the 24th international conference on World Wide Web*, pp. 111–112, 2015, doi: 10.1145/2740908.2742726.
- [36] R. Salakhutdinov and A. Mnih, "Probabilistic matrix factorization," *Advances in neural information processing systems*, vol. 20, pp. 1–8, 2007.
- [37] F. Strub, J. Mary, and R. Gaudel, "Hybrid collaborative filtering with autoencoders," 2016, [Online]. Available: <http://arxiv.org/abs/1603.00806>.
- [38] H. Tahmasebi, R. Ravanmehr, and R. Mohamadrezaei, "Social movie recommender system based on deep autoencoder network using Twitter data," *Neural Comput. Appl.*, vol. 33, no. 5, pp. 1607–1623, 2021, doi: 10.1007/s00521-020-05085-1.
- [39] S. Zhang, L. Yao, and X. Xu, "Autosvd++: An efficient hybrid collaborative filtering model via contractive auto-encoders," *SIGIR 2017 - Proc. 40th Int. ACM SIGIR Conf. Res. Dev. Inf. Retr.*, pp. 957–960, 2017, doi: 10.1145/3077136.3080689.
- [40] 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, doi: 10.1016/j.neunet.2017.03.009.
- [41] B. Dong, Y. Zhu, L. Li, and X. Wu, "Hybrid collaborative recommendation of co-embedded item attributes and graph features," *Neurocomputing*, vol. 442, pp. 307–316, 2021, doi: 10.1016/j.neucom.2021.01.129.
- [42] C. C. Tan and C. Eswaran, "Performance comparison of three types of autoencoder neural networks," *Proc. - 2nd Asia Int. Conf. Model. Simulation, AMS, 2008*, pp. 213–218, doi: 10.1109/AMS.2008.105.
- [43] R. Mu, "A survey of recommender systems based on deep learning," *IEEE Access*, vol. 6, pp. 69009–69022, 2018, doi: 10.1109/ACCESS.2018.2880197.
- [44] R. Salakhutdinov and A. Mnih, "Bayesian probabilistic matrix factorization using markov chain Monte Carlo," *Proc. 25th Int. Conf. Mach. Learn.*, 2008, pp. 880–887, doi: 10.1145/1390156.1390267.
- [45] H. Liang and T. Baldwin, "A probabilistic rating auto-encoder for personalized recommender systems," *Int. Conf. Inf. Knowl. Manag. Proc.*, vol. 19-23-Oct-, 2015, pp. 1863–1866, doi: 10.1145/2806416.2806633.
- [46] Y. Koren, "Factorization meets the neighborhood: a multifaceted collaborative filtering model," in *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2008, pp. 426–434, doi: 10.1145/1401890.1401944.

## BIOGRAPHIES OF AUTHORS



**Ahed Mleih Al-Sbou**    is currently a lecturer in the Information Technology School of Computer Science at the University of Al-Hussein Bin Talal where he has been a faculty member since 2014. He received the B.Sc. degree in computer science from Al-Hussein Bin Talal University, Ma'an, Jordan, in 2006, and the M.Sc. degree in computer science from Al-Balqa Applied University, Salt, Jordan, in 2012, is currently a Ph.D. student in the Department of Computer Sciences, University Malaysia Terengganu, Kuala Terengganu, Malaysia. His research interests include the applications of artificial intelligence, deep learning, data mining, and recommendation systems. He can be contacted at email: [ahed\\_alsbou@ahu.edu.jo](mailto:ahed_alsbou@ahu.edu.jo).



**Noor Hafhizah Abd Rahim**    has Master degree in computer science from University of Malaya, Malaysia and a PhD degree from University of Bristol, UK in artificial intelligence field. She is currently a senior lecturer at Faculty of Ocean Engineering Technology and Informatics, Universiti Malaysia Terengganu. Her research interests are semantic web, machine learning, and natural language processing. She can be contacted at email: [noorhafhizah@umt.edu.my](mailto:noorhafhizah@umt.edu.my).