Neurocomputing 488 (2022) 557–571

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

A deep learning based trust- and tag-aware recommender system

Sajad Ahmadian [a,*], Milad Ahmadian [b], Mahdi Jalili [c]

[a] Faculty of Information Technology, Kermanshah University of Technology, Kermanshah, Iran
[b] Department of Computer Engineering, Razi University, Kermanshah, Iran
[c] School of Engineering, RMIT University, Melbourne, Australia

ARTICLE INFO

Article history:
Received 19 April 2021
Revised 27 October 2021
Accepted 21 November 2021
Available online 27 November 2021

Keywords:
Recommender systems
Collaborative filtering
Deep neural networks
Sparse autoencoder
Trust
Tag

ABSTRACT

Recommender systems are popular tools used in many applications, such as e-commerce, e-learning, and social networks to help users select their desired items. Collaborative filtering is a widely used recommendation technique that employs previous ratings of users to predict their future interests. Lack of sufficient ratings often reduces the performance of collaborative filtering recommendation methods. Additional side resources, such as trust relationships and tag information can be employed to enhance the recommendation accuracy. However, trust and tag data are often heavily sparse as users mainly provide insufficient information about these side resources. Moreover, such additional resources have generally large dimensions which result in increasing the computational complexity of recommendation models in calculating similarity values. To cope with these problems, a new recommendation model is proposed that utilizes deep neural networks to model the representation of trust relationships and tag information. To this end, a sparse autoencoder is used to extract latent features from user-user trust relationships and user-tag matrices. Then, the extracted latent features are utilized to calculate similarity values between users, which are then used to form the nearest neighbors of the target user and predict unseen items. The proposed method can tackle the data sparsity problem and reduce the computational complexity of recommender systems as the extracted latent features have smaller dimensions in comparison to the original data. Experimental results on two benchmark datasets reveal the effectiveness of the proposed method and show its outperformance over state-of-the-art recommender systems.

1. Introduction

Recently, with the massive spread of Internet users, the total amount of digital information has grown exponentially. This makes users to be confused in processing such huge information, a challenge that is known as the information overload problem [1,2]. Intelligent systems can help users in finding their relevant information and speeding up the search process by utilizing machine learning algorithms. Recommender systems (RSs) are one of the most popular types of such systems that have been successfully applied to various applications, such as recommending movies, games, music, news, and articles [3,4]. The aim of RSs is to use previous preferences of users, model their tastes, and employ it to predict their likely ratings on items. To this end, one needs to analyze the user-item interaction data, which is indeed the ratings given by users on items [5]. RSs based on user-item interaction data are known as Collaborative Filtering (CF) recommenders [6,7]. Another type of RSs is based on contextual information of items and/or users that matches the relevant items to users. These RSs are known as content-based recommenders [8]. There are some hybrid RSs that use both contextual information and user-item interaction data [9,10].

CF methods rely on the fact that the preferences of users remain stable over time and users with known preferences in the past will most probably have similar interests in the future. One often requires collecting sufficient data before performing the CF recommendation process to provide accurate and dependable recommendations to users. Performance of CF methods usually declines due to lack of enough ratings from users to items, a problem known as data sparsity problem [11–13]. A possible solution to overcome the data sparsity problem in RSs is to use additional information in the recommendation process [14,15]. Some research works have considered additional data such as trust relationships between users or tag information [16–18]. The main idea behind using trust relationships in RSs is that the existence of a trust relationship between two users could be linked to the similarity of their interests [19–22]. On the other hand, tag information provides supplementary data by summarizing the characteristics

* Corresponding author.
E-mail addresses: s.ahmadian@kut.ac.ir (S. Ahmadian), m.ahmadian@stu.razi.ac.ir (M. Ahmadian), mahdi.jalili@rmit.edu.au (M. Jalili).

https://doi.org/10.1016/j.neucom.2021.11.064
0925-2312/© 2021 Elsevier B.V. All rights reserved.

of items and reflecting user preferences through tagging behaviors [23].

Although utilizing supplementary information often leads to improving the accuracy of RSs, these supplementary data are even much sparser than the user-item rating data. In other words, users often provide insufficient information about additional resources, such as trust and tag data. Thus, they are heavily sparse. On the other hand, the matrices corresponding to trust relationships between users and tag information have large dimensions as there are generally numerous users, items, and tags in real-world applications. Therefore, considering such additional information in the recommendation process might largely impact the computational complexity of the algorithms and make them less practical for large-scale systems [3,24]. Matrix factorization technique has been successfully used in RSs as a useful tool to learn latent factors from the user-item rating matrix and supplementary information [19,20,25,26]. This helps to reduce the computational complexity by using these latent factors in predicting unknown ratings. Nevertheless, the sparsity problem in the user-item rating matrix and supplementary information is still a challenge that might make the learned latent factors to be ineffective. Also, the quality of the learned latent factors is a crucial point impacting the performance of recommendation methods in improving the prediction accuracy.

To address the above-mentioned problems, this paper presents a novel and effective recommender system based on deep neural networks to alleviate the data sparsity problem in RSs. The proposed method utilizes deep neural networks to model the representation of supplementary information, including trust relationships and user-tag matrices. This representation is obtained by employing sparse autoencoders to extract latent features from trust relationships and user-tag matrices. The extracted latent features with low dimensions are then used to calculate similarity values between users and produce recommendations by predicting unknown items. A list of the main contributions of the proposed method is represented as follows:

- The proposed method takes the advantages of latent features of both trust and tag information obtained by deep neural networks to improve the performance of recommendation methods. Although several recommendation methods have been previously proposed based on deep neural networks by employing trust and tag information, to the best of our knowledge, this work is the first attempt to use deep neural networks to construct low dimensional latent features based on both trust relationships between users and tag information.
- To address the data sparsity problem, the proposed method utilizes deep sparse autoencoders by considering sparse coding regularization to obtain the latent features related to the input raw data. This boosts the performance of the proposed method, especially when it faces sparse data. We experimentally prove that the proposed method alleviates the data sparsity problem better than state-of-the-art recommendation models.
- Three different scenarios are proposed to calculate similarity values between users based on the latent features extracted from the user-user trust relationships matrix, user-tag matrix, and the combination of these two types of latent features.
- In contrast to classical collaborative filtering recommendation methods that utilize original input matrices with large dimensions to compute the similarity values, the proposed method computes them based on the obtained latent features of trust and tag information with low dimensions. Therefore, it has lower computational complexity than classical recommendation methods in calculating similarity values between users.
- Several experiments have been performed on two benchmark datasets and their results demonstrate the advantages of the proposed recommender system in providing more accurate recommendations than state-of-the-art recommenders.

The rest of the paper is structured as follows. The related works are summarized in section 2. Section 3 introduces the proposed method. The discussion of performed experiments and the analysis of experimental results are presented in section 4. The paper is concluded in section 5.

2. Related works

2.1. Trust-aware recommender systems

Trust-aware recommender systems rely on the fact that users usually tend to accept recommendations from their trusted friends [27–29]. Trust relationships between users have been used in the literature as the most effective supplementary information to find users' interests and improve the quality of recommenders. [30] proposed a trust-aware recommender by integrating explicit and implicit trust relationships into the recommendation process. To this end, the explicit trust relationships are collected through social networks and the implicit trust relationships are measured based on users' implicit feedbacks. In [31], a trust-based recommender system was proposed based on a trust propagation model to provide more accurate recommendations. An adaptive trust-aware recommender was proposed in [32], which is based on a new trust measurement inferred by a user-item bipartite network. In [33], an effective neighbor selection model was proposed to enhance the accuracy of social recommenders by removing ineffective users from the nearest neighbors set leading to provide more reliable recommendations.

Gohari et al. proposed a recommendation method based on a new hybrid model of local and global trust [34]. The impact of these two trust models is dynamically evaluated using a reliability measurement. [35] tackled the data sparsity issue by introducing a reliability measure to improve users' rating profiles that their ratings and trust relationships are not sufficient to produce accurate recommendations. A CF-based recommender was proposed in [36] using trust and emotion measurements to deal with the data sparsity and shilling attacks problems. The trust measurement is utilized to obtain the nearest neighbors using subjective and objective trust models. Then, the nearest neighbors are evaluated according to emotional consistency among users to exclude malicious users. Guo et al. used trust relationships that are obtained using implicit feedbacks of users to propose three factored similarity models [37]. [38] proposed a trust-aware matrix factorization approach by integrating several resources into the recommendation method to alleviate the data sparsity and cold start problems. In [39], first a trust propagation mechanism is introduced based on the domain knowledge covered by the experts to calculate trust relationships between them in social networks. Then, the subjective and objective weights are obtained by integrating the calculated trust values to generate recommendations for users. Guo et al. [40] proposed a trust-based recommender system by fusing three heterogeneous graphs including user-item interaction graph, user-user trust relation graph, and item-item knowledge graph. To this end, the feature vectors obtained by these heterogeneous graphs are utilized to apply the fusion mechanism and produce recommendations for users.

In [41], a cross-domain recommendation algorithm is proposed based on a deep neural matrix factorization strategy and trust relationships between users to address the cold start challenge in recommender systems. To this end, similarity values between the users related to the source and target domains are computed using the latent features obtained through the deep neural matrix factorization model. Moreover, ant colony optimization is utilized to make a bipartite trust graph and determine the nearest neighbors set for users. Ma et al. developed a trust-based cross-domain recommendation method by utilizing a latent space representation strategy [42]. In particular, a deep neural network model is employed to find the latent features of users based on user and item matrixes constructed by the probabilistic matrix factorization model. In [43], the authors incorporated three important factors: trust, geographic influence, and temporal influence into point of interest collaborative filtering to improve the efficiency of the network representation learning model in calculating similarity values between users. Cold start and data sparsity challenges have been addressed in [44] by incorporating sparse trust relationships into the recommendation process. These sparse trust relationships are determined using the decomposition of explicit trust values into fine-grained features which are then combined using the support vector machine approach. Chen et al. utilized the generative adversarial network and recurrent neural network to develop a trust-based recommendation model [45]. Moreover, a discriminative model is proposed to distinguish between real ratings and the ratings generated by the generative adversarial network. In [46], a tensor factorization method is introduced based on trust relationships between users to address the data sparsity issue in recommender systems. To this end, trust relationships and implicit feedback are integrated to improve the accuracy of recommendations provided for users. Also, trust relationships between users are represented as two different types named unilateral trust and mutual trust leading to better use of trust relationships.

2.2. Tag-aware recommender systems

Tag-aware RSs use social tag information as an important resource to provide the users ease to organize, manage, share, and search their interests [47]. Mauro et al. proposed a tag-based CF method by integrating frequently co-occurring interests in information search logs to enhance the quality of RSs [48]. In [49], a tag-based method was proposed based on exploiting correlations between tags derived by co-occurrence of interests. In [50], a co-SVD model was proposed based on enriching user preferences by both rating data and tag information, where relevance between tags and item features is explored to enhance the single data source and alleviate the over-fitting. A tag-aware recommendation method was proposed in [23] based on similarities between user and item profiles via discovering frequent user-generated tag patterns. Xie et al. introduced a tag-based resource recommender by utilizing the tag data associated with different resources [51]. The method considers the users' feedback to dynamically adjust recommendations during the recommendation process. [17] proposed a tag-aware music recommender method based on semantic labels to construct a highly sparse ratings matrix. This method considers temporal information by employing a Gaussian state-space model. In [52], a tag-aware recommender system is introduced which utilizes a multi-layer perceptron neural network to extract a latent space for items based on tag vectors. Moreover, an attention network is applied to obtain the representations of users. Then, the latent features of users and items are integrated to predict unknown ratings and generate recommendations for users. Banerjee et al. [53] introduced a recommender system that integrates three different data resources including the user-item rating matrix, trust relationships between users, and tag information to make recommendations for users. The main advantage of this recommendation method is to provide more reliable input data for recommender systems to address the data sparsity problem.

In [54], a collaborative tag-based recommendation approach is presented to make recommendations in collaborative tagging systems. Particularly, a three-mode tensors model is proposed to calculate the correlations between three components: users, tags, and items. In addition, content-based information of items is utilized in the recommendation process to increase the quality of recommendations. Guan et al. introduced a document recommendation model based on tagging data [55]. To this end, firstly, semantic spaces of users, tags, and documents are achieved using a graph-based representation learning model. Then, the obtained representations are utilized to find the relationships between users, tags, and documents, and produce recommendations. [56] addressed the personalized tag recommendation problem by representing tagging data and their relationships as a heterogeneous graph and developing a ranking-based recommendation algorithm. Accordingly, the constructed heterogeneous graph is used to make a ranked list of tags that can be provided for users as recommendations. In [57], a point of interest (POI) recommendation algorithm is developed which employs tag data to make embedding models for providing recommendations to users. For this purpose, the Cosine similarity function is utilized to compute similarity values between users and the representation of POIs obtained using the embedding models. Quintanilla et al. proposed a personalized tag recommender system based on deep neural networks to learn the representations of users' preferences [58]. To do this, a joint training model is developed to merge the representations of users' preferences and tag data leading to an improvement in the recommendation efficiency. Moreover, the adversarial learning model is utilized to predict tags for users according to the previous user-generated tags in the system. In [59], a tag-based recommendation approach is developed based on a neural attention network to learn hidden tag information and determine the weights of tags for users. In addition, the data sparsity issue is addressed by integrating tag data and user-item interactions leading to an enhancement in the accuracy of recommendations.

2.3. Deep neural networks-based recommender systems

Deep neural networks play an important role in a large number of applications, including recommender design [60], classification [61], and forecasting [62–65]. They are powerful tools to effectively learn user and item representations to utilize in the recommendation process [66]. Recently, various research works have been developed to design deep neural networks to enhance the accuracy of RSs. In [67], a deep learning model was proposed to learn an effective representation of content and tag related to items. The model extracts implicit relationships between users and items by learning effective latent representations of supplementary information in an unsupervised manner. In [68], the data sparsity and cold start problems were addressed by applying deep neural networks to find latent features related to users and items. Guan et al. proposed a multi-view model to use multiple sources of content for designing a recommendation model, where stacked autoencoder networks were applied to learn latent features [69]. Zhang et al. applied a deep nonlinear structure to obtain latent features related to users and items and integrated them with the matrix factorization technique [70]. In [71], a recommendation method is proposed based on the representations of latent features for users and items which are obtained by employing deep convolutional neural network. The vector dot product approach is utilized in the recommendation process to calculate the unknown ratings based on the obtained latent features of users and items. In [72], a deep recommendation method is proposed based on two effective attention mechanisms to generate accurate recommendations for users. To this end, the first attention mechanism is introduced for deep convolutional neural networks to extract the latent features of users according to the reviews. Moreover, the second attention mechanism is developed to obtain the most effective latent features of items based on the user-item rating matrix.

An effective recommendation approach was proposed in [73] based on a feed-forward deep neural network for balancing popularity and novelty of recommended items. In [74], a recommendation method was proposed based on feed-forward deep neural networks to simulate correlation between users and items. Wu et al. integrated the matrix factorization method and deep neural networks to propose an effective model to find description documents related to both users and items [75]. [24] applied a sparse autoencoder to obtain latent features related to tag information. Although this method can effectively extract latent features of tag information, it fails to consider other important information, such as trust relationships. In [76], a tag-based recommender system is proposed based on a deep learning model to extract latent features of users and items. [77] addressed the cold start problem by proposing a model according to the combination of CF recommender and deep learning neural networks, where the content features of items are extracted by deep neural networks. Moreover, the timeSVD++ model is improved to produce more accurate predictions for cold start items. [78] utilized deep learning-based approaches in collaborative filtering to combine the representation and matching function learning models. In [79], back propagation network and attention mechanism are utilized to learn latent features of target users and their nearest neighbors. Xue et al. introduced a recommender by using the integration of matrix factorization model and deep neural network to enhance the recommendation quality [80]. A general framework is proposed in [81] to replace the inner product in matrix factorization models with a neural network architecture to improve the performance of recommendation methods.

3. Trust- and tag-aware deep neural networks based recommender

This section includes details of the proposed trust- and tag-aware recommendation model which is called TTDNN. This model includes three main steps. First, the original raw data including user-item tag and user-user trust matrices are modeled as suitable vectors to make the suitable inputs for deep networks. Then, in the second step, an autoencoder is applied to extract latent features of tag data and trust relations. In the last step, the extracted features are utilized to compute the similarity values and predict the unseen ratings. Table 1 describes the notations used in the proposed method. Details of the proposed method are provided in the following.

3.1. Data representation

In the proposed method, two information resources are used as inputs to deep networks: trust relationships and users tag matrices. The trust relationships matrix contains explicit social relations between the users that are collected from social networks. Suppose that U, I, and T are the set of users, items, and tags in the recommender system, respectively. We represent the trust relationships matrix as $S_{|U|\times|U|}$, where $S_{u,v}=1$ shows that a trust relationship exists between users u and v, otherwise; $S_{u,v}=0$. The users tag matrix contains the tag information, which is usually in the format $F=(U,I,T)$. To apply the users tag matrix as the input of the deep network, the original 3D matrix is projected onto two 2D matrices $R_{|U|\times|I|}$ and $Y_{|U|\times|T|}$. For the user-item matrix R, if $\sum_{t\in T}F_{u,i,t}>0$ then $R_{u,i}=1$, otherwise $R_{u,i}=0$, where $u\in U$, $i\in I$ and $t\in T$. Moreover, for the user-tag matrix, if $t\in T$ and $\sum_{i\in I}F_{u,i,t}>0$ then $Y_{u,t}=1$, otherwise $Y_{u,t}=0$. After the pre-processing of the original information, trust relationships and tag matrices are used as the inputs to the deep network to learn latent features. It is worth noting that the user-item rating matrix R is employed to calculate unknown ratings in the proposed method. Therefore, deep neural networks are not applied on the user-item ratings matrix and these ratings are directly used in the recommendation process of the proposed method. Fig. 1 shows the representations of three input matrices S, R, and Y.

Table 1
Description of notations used in this work.

Notation	Description
$\boldsymbol{U}$	Set of users
$\boldsymbol{I}$	Set of items
$\boldsymbol{T}$	Set of tags
$\boldsymbol{S}$	Trust relationships matrix
$\boldsymbol{R}$	User-item rating matrix
$\boldsymbol{Y}$	User-tag matrix
$\boldsymbol{\delta}$	Activation function of sparse autoencoder
$\boldsymbol{X}$	p-dimensional input vector of sparse autoencoder
$\boldsymbol{W}$	Connection weights matrix in sparse autoencoder
$\boldsymbol{b}$	Bias vector in hidden layer of sparse autoencoder
$\boldsymbol{c}$	Bias vector in output layer of sparse autoencoder
$\boldsymbol{X}$	p-dimensional output vector of sparse autoencoder
$\boldsymbol{h}$	q-dimensional vector of activation values in hidden layer
$\boldsymbol{m}$	Number of input samples
$\boldsymbol{\lambda}$	Weight decay parameter of sparse autoencoder
$\tilde{\mathbf{S}}$	Trust-based latent features of users
$\tilde{\mathbf{Y}}$	Tag-based latent features of users
$\boldsymbol{k}$	Number of latent features
$\boldsymbol{\beta}$	Non-negative real value in the combined similarity function
$\boldsymbol{N_u}$	Nearest neighbors set of user u
$\boldsymbol{P_{u,i}}$	Predicted rating for user u on item i

3.2. Latent features extraction

Autoencoders are known as one of the artificial neural network models, which have been frequently applied to learn latent features of input data. The aim of this type of neural network is data dimensionality reduction by learning a representation (encoding) of input data in an unsupervised manner. The sparse autoencoder is a specific type of autoencoders that is based on a sparsity constraint, leading to enhance the representation quality, especially in sparse input data. Therefore, sparse autoencoder can obtain a high-quality representation of input data, especially when recommendation methods face sparse input data [82]. Generally, an input layer, a hidden layer, and an output layer are considered in a sparse autoencoder. Moreover, an encoder and a decoder form its two main parts. The encoder is formed as an input layer and a hidden layer which is used to represent input data as latent features. On the other hand, the decoder is formed as hidden layer and an output layer employed to translate feature vector into input space and generate a reconstructed space.

Suppose that $X=\left[x_1,x_2,\cdots,x_p\right]^T\in R^p$ is a p-dimensional input vector. In the encoder of the sparse autoencoder, nonlinear representations can be obtained for the vectors of input samples [83]. The representation of a given input vector X is modelled as follows:

$$\begin{aligned} h_1 &= \delta\left(W_{11}x_1+W_{12}x_2+\cdots+W_{1p}x_p+b_1\right) \\ h_2 &= \delta\left(W_{21}x_1+W_{22}x_2+\cdots+W_{2p}x_p+b_2\right) \\ h_q &= \delta\left(W_{q1}x_1+W_{q2}x_2+\cdots+W_{qp}x_p+b_q\right) \end{aligned} \tag{1}$$

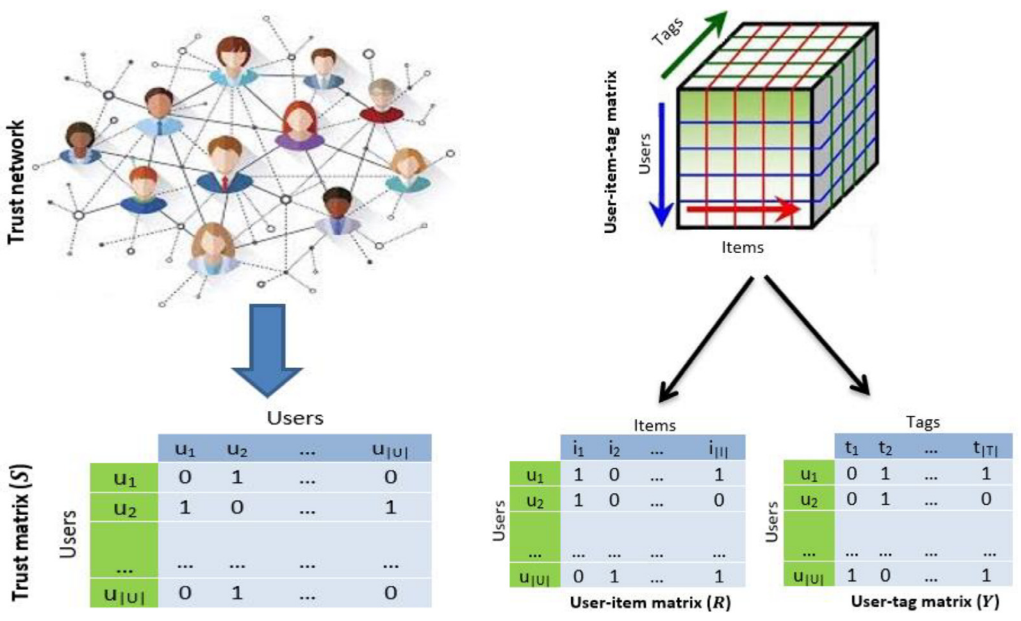

Fig. 1. Representation of input matrices: (a) representation of trust matrix (S) based on trust relationships, (b) representation of user-item matrix (R) and user-tag matrix (Y) based on user-item-tag matrix.

where h_i is the activation value of unit i in the hidden layer, q is the number of neurons in the hidden layer, δ indicates the activation function, W_{ij} refers to the connection weight between unit i in the hidden layer and unit j in the input layer, and b_i refers to the bias from unit i in the hidden layer. According to Eq. (1), the activation value of unit i in the hidden layer can be calculated as follows:

$$h_i = \delta\left(\sum_{j=1}^{p} W_{ij}x_j + b_i\right), i = 1, 2, \cdots, q \tag{2}$$

In the decoder of the sparse autoencoder, the output values of neurons in the output layer can be obtained. The following equation is used to calculate the values of output neurons in the sparse autoencoder:

$$x_1 = \delta\left(W_{11}h_1 + W_{12}h_2 + \cdots + W_{1q}h_q + c_1\right)$$

$$x_2 = \delta\left(W_{21}h_1 + W_{22}h_2 + \cdots + W_{2q}h_q + c_2\right)$$

$$x_p = \delta\left(W_{p1}h_1 + W_{p2}h_2 + \cdots + W_{pq}h_q + c_p\right) \tag{3}$$

where x_i is the value of unit i in the output layer, δ indicates the activation function, W_{ij} refers to the connection weight between unit i in the output layer and unit j in the hidden layer, and c_i refers to the bias from unit i in the output layer. According to Eq. (3), the value of unit i in the output layer can be calculated as follows:

$$x_i = \delta\left(\sum_{j=1}^{q} W_{ij}h_j + c_i\right), i = 1, 2, \cdots, p \tag{4}$$

The objective of the sparse autoencoder is to reconstruct the output vector X as similar as possible the input vector X by performing the learning process. The following equation is used as the objective function in the sparse autoencoder:

$$\min_{W,b,c} \| X - X\|^2 \tag{5}$$

Suppose that $A = \{a_1, a_2, \cdots, a_m\}$ is a set of input samples containing m p-dimensional input vectors and $h(a_i) \in R^k$ is a new sparse feature vector obtained by a sparse autoencoder, where k is the number of latent features. According to Eq. (2), the representation of a given input vector a_i can be modelled as follows:

$$h(a_i) = \delta(Wa_i + b) \tag{6}$$

where W denotes the weight matrix between the input and output layers of the sparse autoencoder and b is the bias of the input layer. According to Eqs. (4) and (5), the decoder minimizes the error between the input and output using the following objective function:

$$\min_{W,b,c} \sum_{i=1}^{m} \|\delta\left(W^T h(a_i) + c\right) - a_i\|^2 \tag{7}$$

where W^T denotes the transpose of W, c indicates the bias of output layer, and m represents the number of input samples.

To optimize the convex objective function (Eq. (7)), the L-BFGS approach [84] is used as a convex optimization tool. The sparse coding of sparse autoencoder represents input data as a weighted linear combination of vectors. To this end, the sparse autoencoder with sparse penalties is used to generate latent features by using $L1$ norm regularization as the sparsity penalty [85]. The $L1$ norm regularization is added to the objective function of the sparse autoencoder as follows:

$$J(W, b, A) = \frac{1}{m}\sum_{i=1}^{m} \|\delta\left(W^T h(a_i) + c\right) - a_i\|^2 + \lambda P \tag{8}$$

where $J(W, b, A)$ denotes the objective function of the sparse autoencoder, λ is the weight decay parameter to control the weight of sparsity penalty term preventing over-fitting, and P is the sparsity penalty based on $L1$ norm regularization which is calculated as follows:

$$P = \frac{1}{m}\sum_{i=1}^{m}\sum_{j=1}^{q} \|h_j(a_i)\|_1 \tag{9}$$

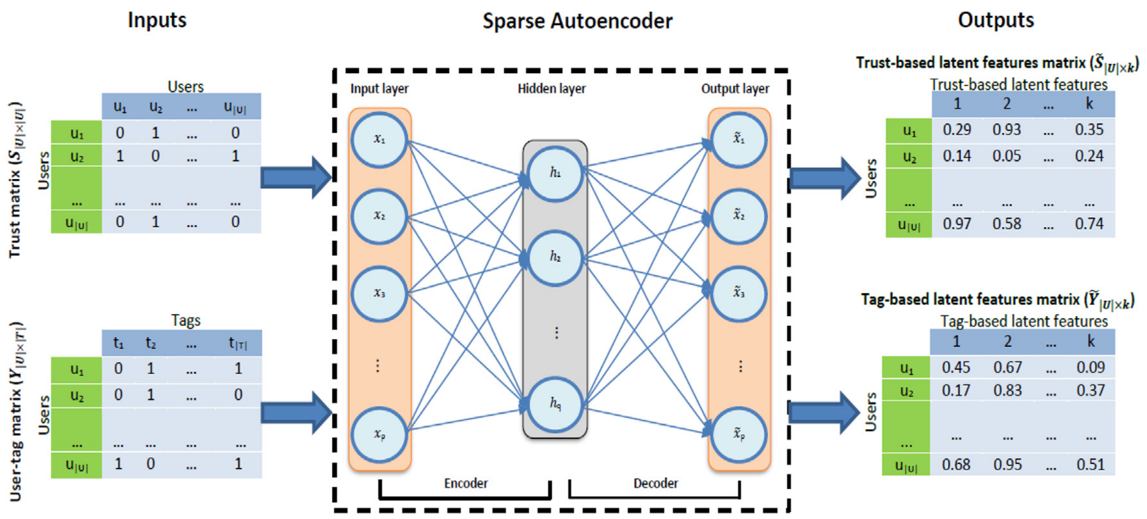

Fig. 2. The latent features extraction step in the proposed approach.

where m is the number of input examples and q is the number of hidden units. It is worth noting that the most straightforward gradient-based models are difficult to apply to the objective function in Eq. (8) as it is not continuously differentiable [86]. Therefore, a proximal gradient descent model is applied for structured sparsity optimization. Accordingly, the $L1$ norm regularization can be calculated using the form of proximation as follows [85]:

$$\|h_j(a_i)\|_1 = \sqrt{h_j^2(a_i) + \varepsilon}, \varepsilon = 1e - 4 \tag{10}$$

Eq. (8) represents the objective function of the sparse autoencoder with only one hidden layer. To establish a deep autoencoder, more hidden layers might be required in the training phase. In this paper, we use the method proposed by Hinton et al. [87] as a learning model to train the deep sparse autoencoder. In this training algorithm, the first hidden layer can be trained using the input data, and then the second hidden layer can be trained using the outputs obtained by the first hidden layer, and so on. After training the sparse autoencoder, the weights and biases of the hidden layers are considered as the latent features.

It is worth noting that both trust relationships and user-tag matrices are often heavily sparse as there are many users with insufficient tag information and trust relationships available. Therefore, in the proposed method, we use a sparse autoencoder [85] to learn latent features to cope with the data sparsity problem. To extract the latent features, the input values should be represented as 1D vectors for applying to the autoencoder. Therefore, for each user u, two vectors are formed based on trust relationships matrix S and user-tag matrix Y using the following equations:

$$S_u = \left(S_{u,1}, S_{u,2}, \cdots, S_{u,|U|}\right), u \in U \tag{11}$$

and,

$$Y_u = \left(Y_{u,1}, Y_{u,2}, \cdots, Y_{u,|T|}\right), u \in U \tag{12}$$

where S_u and Y_u are the vectors of user u based on S and Y, respectively. Indeed, S_u and Y_u correspond to the u^{th} rows in S and Y. Then, these vectors are applied as the inputs of the sparse autoencoder to learn the latent features. The sparsity constraint in the sparse autoencoder forces the model to choose only k neurons with the highest activation values in the hidden layer as the latent features and ignore other neurons. Therefore, by considering S_u and Y_u as the inputs of the sparse autoencoder, the corresponding latent features can be considered as follows:

$$\tilde{S}_u = \left(\tilde{S}_{u,1}, \tilde{S}_{u,2}, \cdots, \tilde{S}_{u,k}\right), u \in U \tag{13}$$

and,

$$\tilde{Y}_u = \left(\tilde{Y}_{u,1}, \tilde{Y}_{u,2}, \cdots, \tilde{Y}_{u,k}\right), u \in U \tag{14}$$

where $\tilde{S}_u \in R^k$ and $\tilde{Y}_u \in R^k$ are the latent features of user u obtained based on the trust and tag vectors, respectively. k denotes the number of latent features which is usually much less than the number of users and tags (i.e. $k \ll |U| and\ k \ll |T|$).

The proposed method is able to make representations of input data with lower-dimensional in comparison to the raw input data by obtaining the latent features using the sparse autoencoder. Thus, the proposed method alleviates the data sparsity problem and reduces the computational complexity of the recommendation process. Fig. 2 graphically represents the latent features extraction step in the proposed approach.

3.3. Recommendation

In this step, the recommendation process is applied based on the extracted latent features of users. To this end, the latent features are utilized to measure similarity values between users. As discussed in the previous step, two different latent vectors are extracted based on trust relationships and user-tag matrices. Therefore, three scenarios are considered in the proposed method to provide the recommendations. The details of these scenarios are discussed as follows:

Scenario 1: In this scenario, the similarity values are calculated using the latent features extracted from the trust relationships matrix (i.e. Eq. (13)). Here, we use Cosine similarity function as follows:

$$Trust\_Sim_{u,v} = \frac{\sum_{i=1}^{k} \tilde{S}_{u,i} \times \tilde{S}_{v,i}}{\sqrt{\sum_{i=1}^{k} \tilde{S}_{u,i}^{2}}\sqrt{\sum_{i=1}^{k} \tilde{S}_{v,i}^{2}}} \tag{15}$$

where $Trust\_Sim_{u,v}$ denotes the similarity value of the users u and v, $\tilde{S}_u$ is the latent features vector of user u, $\tilde{S}_{u,i}$ is the i th latent feature of user u, and k indicates the number of features in the latent features vector $\tilde{S}_u$.

Scenario 2: In this scenario, the extracted latent features from the user-tag matrix (i.e. Eq. (14)) are utilized to measure the similarity values as follows:

$$Tag\_Sim_{u,v} = \frac{\sum_{i=1}^{k} \tilde{Y}_{u,i} \times \tilde{Y}_{v,i}}{\sqrt{\sum_{i=1}^{k} \tilde{Y}_{u,i}^{2}}\sqrt{\sum_{i=1}^{k} \tilde{Y}_{v,i}^{2}}} \tag{16}$$

where $Tag\_Sim_{u,v}$ denotes the similarity value of users u and v according to the latent features extracted from the user-tag matrix, $\tilde{Y}_u$ is the latent features vector of user u, $\tilde{Y}_{u,i}$ is the i th latent feature of user u, and k indicates the number of features in the latent features vector $\tilde{Y}_u$.

Scenario 3: This scenario combines both similarity values obtained by the latent features of the trust relationships and user-tag matrices. That is to combine the similarity values $Trust\_Sim_{u,v}$ and $Tag\_Sim_{u,v}$ to obtain the final similarity value. To this end, the F-beta score is utilized to compute the combined similarities as follows:

$$Trust\_Tag\_Sim_{u,v} = \frac{(1+\beta^2) \times (Trust\_Sim_{u,v} \times Tag\_Sim_{u,v})}{(\beta^2 \times Trust\_Sim_{u,v}) + Tag\_Sim_{u,v}} \tag{17}$$

where β is a non-negative real value to measure the importance of trust-based and tag-based similarity values. In other words, $\beta < 1$ gives more weight to $Trust\_Sim_{u,v}$ while $\beta > 1$ gives more weight to $Tag\_Sim_{u,v}$. In the case of $\beta = 1$, both $Trust\_Sim_{u,v}$ and $Tag\_Sim_{u,v}$ have the same importance.

After obtaining the similarity values based on one of the above scenarios, some users with the highest similarity values are determined as the nearest neighbors of the target user. Then, the rating of unseen items can be obtained as follows:

$$P_{u,i} = \frac{\sum_{v \in N_u} Sim_{u,v} \times R_{v,i}}{\sum_{v \in N_u} Sim_{u,v}} \tag{18}$$

where $P_{u,i}$ refers to the rating predicted for user u on item i, N_u denotes the nearest neighbors set of user u, $R_{v,i} \in R$ indicates the rating of the item i assigned by user v, and $Sim_{u,v}$ is one of the similarity functions $Trust\_Sim_{u,v}$, $Tag\_Sim_{u,v}$, and $Trust\_Tag\_Sim_{u,v}$ calculated above. Finally, the recommendations list to the target user is formed by selecting relevant items with the highest ratings. The pseudo-code of the proposed method is represented in Algorithm 1.

Algorithm 1. Trust- and Tag-aware recommender based on Deep Neural Networks (TTDNN).

```
Inputs: Trust relationships matrix $\boldsymbol{S}$, user-item-tag matrix $\boldsymbol{F}$,
    number of latent features $\boldsymbol{k}$, and $\boldsymbol{top\_N}$.
Output: Top-$\boldsymbol{N}$ recommendations list.
1: Divide data into two sets including training set $\boldsymbol{Tr}$ and test
    set $\boldsymbol{Te}$;
2: Project the input 3D matrix $\boldsymbol{F}$ onto two 2D matrices
    including user-item matrix $\boldsymbol{R}$ and user-tag matrix $\boldsymbol{Y}$;
3: Let $\boldsymbol{U}$ and $\boldsymbol{I}$ be the set of users and items in the system,
    respectively;
4: for each user $\boldsymbol{u} \in \boldsymbol{U}$ do
5: Form the trust relationships vector $\boldsymbol{S_u} \in \boldsymbol{R}^{|\boldsymbol{U}|}$ based on trust
    relationships matrix $\boldsymbol{S}$ using Eq. (11);
6: Form the user-tag vector $\boldsymbol{Y_u} \in \boldsymbol{R}^{|\boldsymbol{T}|}$ based on user-tag matrix
    $\boldsymbol{Y}$ using Eq. (12);
7: end for
8: for each user $\boldsymbol{u} \in \boldsymbol{U}$ do
9: Apply sparse autoencoder training phase to learn latent
    features based on the trust vector $\boldsymbol{S_u} \in \boldsymbol{R}^{|\boldsymbol{U}|}$ using Eqs. (6)-
    (10);
10: Choose $\boldsymbol{k}$ neurons with the highest activation values in the
    hidden layer as the latent features vector $\tilde{\boldsymbol{S}}_{\boldsymbol{u}} \in \boldsymbol{R}^{\boldsymbol{k}}$;
11: for each user $\boldsymbol{u} \in \boldsymbol{U}$ do
12: Apply sparse autoencoder training phase to learn latent
    features based on the user-tag vector $\boldsymbol{Y_u} \in \boldsymbol{R}^{|\boldsymbol{T}|}$ using Eqs.
    (6)-(10);
13: Choose $\boldsymbol{k}$ neurons with the highest activation values in the
    hidden layer as the latent features vector $\tilde{\boldsymbol{Y}}_{\boldsymbol{u}} \in \boldsymbol{R}^{\boldsymbol{k}}$;
14: end for
15: for each pair of users $\boldsymbol{u}$ and $\boldsymbol{v} \in \boldsymbol{U}$ do
16: Calculate the similarity value between users $\boldsymbol{u}$ and $\boldsymbol{v}$
    based on one of the three proposed scenarios using Eqs.
    (15)-(17);
17: end for
18: for all $\boldsymbol{r_{u,i}} \in \boldsymbol{Te}$ where $\boldsymbol{u} \in \boldsymbol{U}$ and $\boldsymbol{i} \in \boldsymbol{I}$ do
19: Calculate the unknown rating $\boldsymbol{P_{u,i}}$ using Eq. (18);
20: end for
21: Recommend $\boldsymbol{top\_N}$ items as the recommendations list;
```

3.4. Computational complexity analysis

There are three steps in the proposed method including: (1) data representation, (2) latent features extraction, and (3) recommendation. In the first step, the trust matrix is constructed using social relationships between users, which its complexity is $O\left(|U|^2\right)$. Moreover, the user-item-tag matrix is divided into user-item and user-tag matrices, and it also requires a linear scan with a complexity of $O(|U||I||T|)$. Therefore, the first step requires a computational complexity of $O\left(|U|^2 + |U||I||T|\right)$. In the second step, a sparse autoencoder is employed to extract latent features of trust

relationships and tag information. We use L-BFGS as an iterative approach to optimize the parameters of the sparse autoencoder, which its complexity should be determined according to the number of iterations. If $\widehat{iter}$ is the average number of iterations, then L-BFGS requires a computational complexity of $O\left(|T| \times \widehat{iter} \times \sum_{i=1}^{n} z_i\right)$ for learning the latent features obtained based on tag data where n denotes the number of hidden layers and z_i refers to the number of neurons in the i th hidden layer. In the case of learning the latent features of trust relationships, the complexity is $O\left(|U| \times \widehat{iter} \times \sum_{i=1}^{n} z_i\right)$. Therefore, the overall computational complexity related to the second step is $O\left((|T| + |U|) \times \widehat{iter} \times \sum_{i=1}^{n} z_i\right)$. In the third step, the similarity values are calculated which its complexity is $O\left(k|U|^2\right)$ where k denotes the number of the latent features. Moreover, these similarity values should be sorted to obtain user neighborhood N_u, for which the complexity is $O\left(|U|^2 \log|N_u|\right)$. Then, Eq. (18) is used to predict unknown ratings and top_N items are recommended to the target user; its complexity is $O(|I||N_u| \log top\_N)$ where top_N denotes the recommendations list length. Therefore, the complexity of the third step is $O\left(k|U|^2 + |U|^2 \log|N_u| + |I||N_u| \log top\_N\right)$. Finally, the total complexity of the proposed method is $O\left(|U|^2 + |U||I||T| + (|T| + |U|) \times \widehat{iter} \times \sum\right.$
$\left. i = 1^n z_i + k|U|^2 + |U|^2 \log|N_u| + |I||N_u| \log top\_N\right)$ which can be reduced to $O\left(|U|^2 + |U||I||T|\right)$.

4. Experimental results

This section presents the experimental results performed on two datasets to evaluate the performance of the proposed method in comparison to other recommendation methods. A brief description of the recommenders used to compare with the proposed one is represented as follows:

- SoReg [88]: A matrix factorization-based recommendation model which utilizes social network information as social regularization to improve the accuracy of trust-based recommenders.
- SocialMF [14]: This method employs trust propagation mechanism to enhance matrix factorization techniques.
- TrustSVD [20]: This method incorporates both explicit and implicit trust relations into the recommendation process.
- CETrust [3]: This method uses a matrix factorization model to uncoverlatentfeaturesrelatedtotrustrelationshipsbetweenusers.
- TrustPMF [19]: A matrix factorization-based recommender which employs users' trust relationships to learn low-dimensional latent feature spaces.
- DeepSoR [89]: A deep neural network based recommendation method to learn nonlinear features from social relations and integrate into probabilistic matrix factorization for predicting unknown ratings in the recommendation process.
- SDAE [16]: This method utilizes a deep learning model according to stacking autoencoders to learn both user preferences and social influence of friends when generating recommendations.
- AESR [90]: A recommendation method based on deep autoencoder with top-k semantic social information to learn an implicit representation of semantic social information.
- DSPR [91]: This recommendation method employs a deep neural network to maximize deep-semantic similarities between users and their target items by learning latent features of tag data.
- TRSDL [76]: A tag-aware recommendation method based on word embedding techniques and deep neural networks to represent tag data.
- AIRec [52]: A tag-aware recommendation approach that utilizes deep learning models to obtain a latent representation from tag data, and applies it to factorization machines for predicting unknown ratings.
- NLRDMF [92]: A trust-aware recommendation approach that employs deep matrix factorization and deep marginalized denoising autoencoder models to generate accurate recommendations for users.
- TTDNN: The proposed method which is based on extracting latent features of trust relationships and user-tag matrices to use in the recommendation process.

4.1. Datasets

We use Last.fm and Delicious as two real datasets. Last.fm dataset contains 1892 users who are interconnected in a social network based on friendship relationships[1]. The users can tag music tracks and artists, and a list of relevant music tracks is provided for each user. In this dataset, the artists are considered as the items to be recommended to the target user. Delicious dataset contains 1867 users who can tag and share their personal bookmarks (URLs)[2]. Moreover, the users are interconnected in a social network based on mutual fan relationships and each user has bookmarks and tag assignments. In this dataset, the URLs are considered as the items to be recommended to the target user. We use the social information of the Last.fm and Delicious datasets as the trust relationships between users. Also, we remove the tags with the occurrence less than 5 times in the Last.fm and less than 15 times in the Delicious to reduce the amount of calculation in the experiments. Both datasets are available on the website of HetRec 2011[3]. Table 2 represents the demographic information of these datasets.

4.2. Evaluation metrics

Three evaluation metrics including precision, recall, and NDCG (normalized discounted cumulative gain) are employed to evaluate the performance of the algorithms. Precision is measured according to the ratio of relevant items in the recommendation list provided for users. Recall is defined according to the ratio of relevant items in the test set recommended to users as recommendation list. NDCG metric is used to evaluate recommenders by analyzing the ranks of relevant items in the recommendation list. This metric is calculated as follows:

$$NDCG = \frac{DCG}{DCG_{max}} \tag{19}$$

where,

$$DCG = rel_1 + \sum_{i=2}^{top\_N} \frac{rel_i}{log_2(i+1)} \tag{20}$$

where rel_i is a function to determine the relevancy of item i; if item i is relevant then $rel_i = 1$; otherwise $rel_i = 0$. top_N denotes the number of items in recommendations list and DCG_{max} denotes the maximum value of discounted cumulative gain which is defined as follows:

$$DCG_{max} = 1 + \sum_{i=2}^{top\_N} \frac{1}{log_2(i+1)} \tag{21}$$

[1] http://www.last.fm.com
[2] http://delicious.com
[3] http://ir.ii.uam.es/hetrec2011/datasets.html

Table 2
Demographical information of the datasets considered in this work.

Dataset	#Users	#Items	#Tags	#Trust
Last.fm	1892	17,632	11,946	12,717
Delicious	1867	69,226	53,388	7668

4.3. Experimental setup

In the experiments, we use fivefold cross-validation approach. To this end, the datasets are randomly split into five folds that the training set is obtained by merging four folds in each run and the test set comprises the remaining data. The final result is obtained by averaging the results of five runs performed on all of the folds. Several parameters are used in the experiments which their values should be determined. In the latent features extraction step, we use different sparse autoencoders with one, two, or three hidden layers for both Last.fm and Delicious datasets. To this end, the number of neurons in the first, second, and third hidden layer of the sparse autoencoder is set as 1000, 800, and 400, respectively. Also, the number of trust-based and tag-based latent features (k) extracted by the sparse autoencoder is set to $k = 70$ and $k = 80$ for the Last.fm and Delicious datasets, respectively. λ is a parameter used in Eq. (8) to control the impact of the sparsity penalty and to prevent over-fitting in the sparse autoencoder. The value of this parameter is set to $\lambda = 0.02$, which is obtained by a greedy search approach for both Last.fm and Delicious datasets. It should be noted that the input and output layers of the sparse autoencoder contain equal number of neurons, which their values are equal to $|U|$ and $|T|$ for the trust-based and tag-based input vectors, respectively. The sparse autoencoders have been trained in batch mode. $|N_u|$ is the size of neighborhood of user u, which is used in Eq. (18) to calculate the unknown ratings. This parameter impacts the prediction accuracy of the recommendation methods. The lower value of $|N_u|$ decreases the ability of recommendation methods in predicting unseen items as they incorporate fewer ratings of the user neighborhood. On the other hand, choosing large number of users as the user neighborhood reduces the prediction accuracy as the model might incorporate irrelevant ratings into the recommendation process. Therefore, one should determine a good value for $|N_u|$ to make a balance between different criteria. In the proposed method, we use $|N_u| = 90$ for all algorithms and for both Last.fm and Delicious datasets as suggested by [24]. β is another parameter used in Eq. (17) to determine the importance of trust-based and tag-based similarities in the combined similarity function. The value of the parameter β is set to $\beta = 1$ and $\beta = 1.4$ for the Last.fm and Delicious datasets, respectively. Python programming language is used to implement recommendation models and all experiments are performed on a PC configured with a 3.1 GHz CPU and 4 GB of RAM.

4.4. TTDNN under different scenarios to calculate similarities

Three scenarios are considered to calculate the similarities, including trust-based similarity (Scenario 1), tag-based similarity (Scenario 2), and trust- and tag-based similarity (Scenario 3). In this section, several experiments are conducted on Last.fm and Delicious datasets to evaluate the effectiveness of different recommendation scenarios. Table 3 reports the experimental results according to precision, recall and NDCG metrics on Last.fm dataset with different recommendation list lengths $top\_N = 5, 10, 15$. These results indicate that Scenario 3 outperforms Scenario 1 and Scenario 2 based on all evaluation metrics. This is expected as Scenario 3 is richer than the other two scenarios by considering both trust and tag information in the similarity calculation. Table 4 shows the same results for the Delicious dataset, where again Scenario 3 outperforms other two scenarios. In these two datasets, Scenario 1 obtains better performance than Scenario 2, indicating that the latent features of trust relationships produce more accurate recommendations than those of tag information.

4.5. Performance of TTDNN for different structures of deep neural network

Performance of deep neural networks depends on the number of hidden layers in their structure. Several experiments are conducted in this section to assess the role of deep network structures on the performance. To this end, autoencoders with one, two, and three hidden layers are used to extract the latent features. Figs. 3 and 4 indicate the results respectively for the Last.fm and Delicious datasets according to precision, recall, and NDCG measurements. As we can see from these figures, deep neural networks with three hidden layers obtain the best performance in all cases. In the first, second, and third hidden layers, the number of neurons is set to 1000, 800, and 400, respectively. Decreasing the number of neurons in the layers leads to reduce the dimensionality of the latent features extracted by sparse autoencoders in comparison to the dimensionality of the input matrices. Moreover, the latent features provide more abstract

Table 3
Experimental results on Last.fm dataset for different scenarios of the proposed method and different recommendations list lengths (*top_N*).

Scenarios	Precision			Recall			NDCG		
	top_N = 5	top_N = 10	top_N = 15	top_N = 5	top_N = 10	top_N = 15	top_N = 5	top_N = 10	top_N = 15
Scenario 1	0.498 ± 0.01	0.462 ± 0.03	0.425 ± 0.04	0.573 ± 0.02	0.619 ± 0.03	0.642 ± 0.01	0.458 ± 0.01	0.412 ± 0.03	0.384 ± 0.01
Scenario 2	0.457 ± 0.01	0.411 ± 0.02	0.398 ± 0.01	0.528 ± 0.03	0.564 ± 0.04	0.591 ± 0.02	0.415 ± 0.04	0.393 ± 0.02	0.362 ± 0.04
Scenario 3	**0.583 ± 0.02**	**0.556 ± 0.01**	**0.529 ± 0.02**	**0.657 ± 0.01**	**0.692 ± 0.02**	**0.731 ± 0.03**	**0.516 ± 0.03**	**0.482 ± 0.01**	**0.459 ± 0.01**

Table 4
Experimental results on Delicious dataset for different scenarios of the proposed method and different recommendations list lengths (*top_N*).

Scenarios	Precision			Recall			NDCG		
	top_N = 5	top_N = 10	top_N = 15	top_N = 5	top_N = 10	top_N = 15	top_N = 5	top_N = 10	top_N = 15
Scenario 1	0.428 ± 0.02	0.391 ± 0.02	0.356 ± 0.03	0.507 ± 0.02	0.526 ± 0.04	0.559 ± 0.03	0.336 ± 0.03	0.298 ± 0.04	0.274 ± 0.02
Scenario 2	0.397 ± 0.01	0.364 ± 0.03	0.341 ± 0.04	0.472 ± 0.04	0.503 ± 0.03	0.518 ± 0.04	0.312 ± 0.02	0.281 ± 0.01	0.265 ± 0.03
Scenario 3	**0.473 ± 0.02**	**0.451 ± 0.01**	**0.421 ± 0.02**	**0.559 ± 0.03**	**0.608 ± 0.01**	**0.625 ± 0.02**	**0.392 ± 0.02**	**0.378 ± 0.02**	**0.341 ± 0.01**

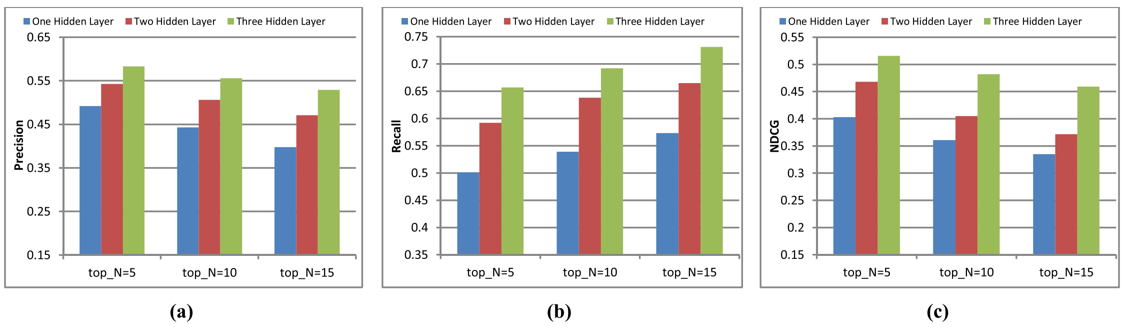

Fig. 3. The results of experiments based on Last.fm dataset for different network structures and different recommendation list lengths (*top_N*): (a) Precision metric, (b) Recall metric, (c) NDCG metric.

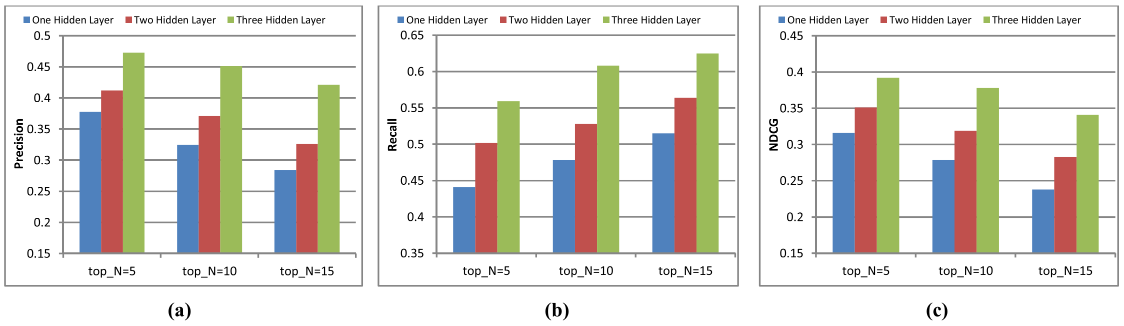

Fig. 4. The results of experiments based on Delicious dataset for different network structures and different recommendation list lengths (*top_N*): (a) Precision metric, (b) Recall metric, (c) NDCG metric.

Table 5
Experimental results on Last.fm dataset for different recommendations list lengths (*top_N*).

Algorithms	Precision			Recall			NDCG		
	top_N = 5	top_N = 10	top_N = 15	top_N = 5	top_N = 10	top_N = 15	top_N = 5	top_N = 10	top_N = 15
SoReg	0.329 ± 0.06	0.301 ± 0.08	0.285 ± 0.05	0.416 ± 0.05	0.448 ± 0.03	0.467 ± 0.05	0.293 ± 0.05	0.262 ± 0.04	0.251 ± 0.03
SocialMF	0.345 ± 0.04	0.317 ± 0.05	0.293 ± 0.07	0.402 ± 0.07	0.451 ± 0.08	0.474 ± 0.06	0.304 ± 0.03	0.276 ± 0.05	0.265 ± 0.06
TrustSVD	0.394 ± 0.05	0.382 ± 0.03	0.357 ± 0.03	0.473 ± 0.05	0.498 ± 0.03	0.512 ± 0.04	0.351 ± 0.06	0.338 ± 0.03	0.309 ± 0.03
CETrust	0.368 ± 0.07	0.341 ± 0.05	0.319 ± 0.04	0.435 ± 0.06	0.468 ± 0.04	0.481 ± 0.07	0.335 ± 0.04	0.316 ± 0.06	0.284 ± 0.07
TrustPMF	0.479 ± 0.04	0.447 ± 0.04	0.415 ± 0.06	0.515 ± 0.07	0.553 ± 0.05	0.579 ± 0.08	0.398 ± 0.05	0.374 ± 0.03	0.349 ± 0.06
DeepSoR	0.492 ± 0.05	0.461 ± 0.06	0.428 ± 0.07	0.562 ± 0.05	0.595 ± 0.06	0.617 ± 0.06	0.436 ± 0.04	0.415 ± 0.03	0.387 ± 0.05
SDAE	0.531 ± 0.03	0.515 ± 0.03	0.475 ± 0.02	0.586 ± 0.06	0.625 ± 0.05	0.643 ± 0.03	0.472 ± 0.03	0.446 ± 0.02	0.414 ± 0.03
AESR	0.545 ± 0.03	0.519 ± 0.02	0.482 ± 0.04	0.613 ± 0.04	0.658 ± 0.03	0.672 ± 0.05	0.491 ± 0.02	0.459 ± 0.01	0.442 ± 0.01
DSPR	0.528 ± 0.04	0.508 ± 0.03	0.472 ± 0.02	0.596 ± 0.02	0.641 ± 0.04	0.658 ± 0.03	0.485 ± 0.03	0.451 ± 0.02	0.426 ± 0.02
TRSDL	0.552 ± 0.02	0.534 ± 0.01	0.487 ± 0.03	0.618 ± 0.03	0.665 ± 0.02	0.689 ± 0.02	0.494 ± 0.02	0.463 ± 0.01	0.445 ± 0.02
AIRec	0.554 ± 0.04	0.532 ± 0.02	0.491 ± 0.02	0.623 ± 0.04	0.667 ± 0.05	0.692 ± 0.03	0.489 ± 0.02	0.461 ± 0.02	0.441 ± 0.05
NLRDMF	0.557 ± 0.03	0.535 ± 0.03	0.493 ± 0.04	0.628 ± 0.02	0.671 ± 0.04	0.698 ± 0.05	0.497 ± 0.03	0.469 ± 0.01	0.447 ± 0.03
TTDNN	**0.583 ± 0.02**	**0.556 ± 0.01**	**0.529 ± 0.02**	**0.657 ± 0.01**	**0.692 ± 0.02**	**0.731 ± 0.03**	**0.516 ± 0.03**	**0.482 ± 0.01**	**0.459 ± 0.01**

and representative version of the input space. This can somehow overcome the sparsity issue of the original data. However, further hidden layers need higher computing cost in the training phase, and one always makes a trade-off between performance and complexity. In the proposed method, we use a sparse autoencoder with three hidden layers.

4.6. Performance comparison with other recommenders

This section discusses the results of performed experiments in order to compare the quality of the proposed method with other recommendation models. Table 5 reports the results based on the Last.fm dataset for different recommendation list lengths. As it is seen, TTDNN obtains the best results according to all evaluation measurements. For example, its precision, recall, and NDCG for $top\_N = 10$ are 0.556, 0.692, and 0.482, respectively. NLRDMF is the second-best performer where its precision, recall, and NDCG values for $top\_N = 10$ are 0.535, 0.671, and 0.469, respectively. Table 6 reports the experimental results based on the Delicious dataset, where again TTDNN is the top-performer recommender. NLRDMF is again the second-best performer based on all evaluation metrics and different lengths of the recommendation list. As we can see from the results, the proposed method obtains better results than other recommenders according to different evaluation metrics (Tables 5 and 6). Therefore, we can conclude that utilizing the latent features of both trust relationships and tag information improves significantly the quality of recommendations. Also, these results show that sparse autoencoders can effectively extract the latent features of side information leading to calculate more accurate similarity values.

Table 6
Experimental results on Delicious dataset for different recommendations list lengths (*top_N*).

Algorithms	Precision			Recall			NDCG		
	top_N = 5	top_N = 10	top_N = 15	top_N = 5	top_N = 10	top_N = 15	top_N = 5	top_N = 10	top_N = 15
SoReg	0.256 ± 0.03	0.231 ± 0.02	0.205 ± 0.02	0.342 ± 0.04	0.371 ± 0.03	0.398 ± 0.05	0.212 ± 0.04	0.194 ± 0.03	0.163 ± 0.02
SocialMF	0.271 ± 0.02	0.256 ± 0.04	0.224 ± 0.04	0.356 ± 0.03	0.398 ± 0.04	0.425 ± 0.03	0.219 ± 0.03	0.205 ± 0.02	0.187 ± 0.03
TrustSVD	0.342 ± 0.04	0.315 ± 0.03	0.287 ± 0.05	0.402 ± 0.05	0.435 ± 0.04	0.457 ± 0.06	0.264 ± 0.04	0.239 ± 0.05	0.225 ± 0.05
CETrust	0.305 ± 0.03	0.269 ± 0.06	0.231 ± 0.03	0.381 ± 0.04	0.415 ± 0.05	0.442 ± 0.03	0.232 ± 0.05	0.217 ± 0.06	0.208 ± 0.04
TrustPMF	0.361 ± 0.05	0.338 ± 0.04	0.309 ± 0.06	0.429 ± 0.06	0.457 ± 0.04	0.461 ± 0.05	0.281 ± 0.03	0.243 ± 0.04	0.229 ± 0.05
DeepSoR	0.403 ± 0.04	0.382 ± 0.05	0.356 ± 0.04	0.475 ± 0.05	0.508 ± 0.03	0.521 ± 0.03	0.325 ± 0.05	0.298 ± 0.05	0.272 ± 0.04
SDAE	0.445 ± 0.02	0.418 ± 0.03	0.387 ± 0.03	0.524 ± 0.03	0.547 ± 0.04	0.556 ± 0.04	0.357 ± 0.02	0.321 ± 0.04	0.309 ± 0.03
AESR	0.441 ± 0.02	0.423 ± 0.01	0.395 ± 0.02	0.506 ± 0.03	0.549 ± 0.03	0.563 ± 0.05	0.349 ± 0.03	0.328 ± 0.04	0.315 ± 0.02
DSPR	0.436 ± 0.03	0.402 ± 0.04	0.375 ± 0.03	0.493 ± 0.04	0.528 ± 0.02	0.541 ± 0.04	0.331 ± 0.04	0.316 ± 0.03	0.297 ± 0.04
TRSDL	0.448 ± 0.02	0.426 ± 0.02	0.401 ± 0.01	0.539 ± 0.02	0.561 ± 0.03	0.582 ± 0.03	0.364 ± 0.02	0.341 ± 0.03	0.319 ± 0.02
AIRec	0.451 ± 0.03	0.428 ± 0.04	0.398 ± 0.02	0.541 ± 0.03	0.568 ± 0.02	0.589 ± 0.04	0.359 ± 0.05	0.348 ± 0.03	0.317 ± 0.02
NLRDMF	0.455 ± 0.04	0.431 ± 0.02	0.406 ± 0.03	0.546 ± 0.02	0.572 ± 0.03	0.592 ± 0.02	0.372 ± 0.03	0.351 ± 0.02	0.323 ± 0.01
TTDNN	**0.473 ± 0.02**	**0.451 ± 0.01**	**0.421 ± 0.02**	**0.559 ± 0.03**	**0.608 ± 0.01**	**0.625 ± 0.02**	**0.392 ± 0.02**	**0.378 ± 0.02**	**0.341 ± 0.01**

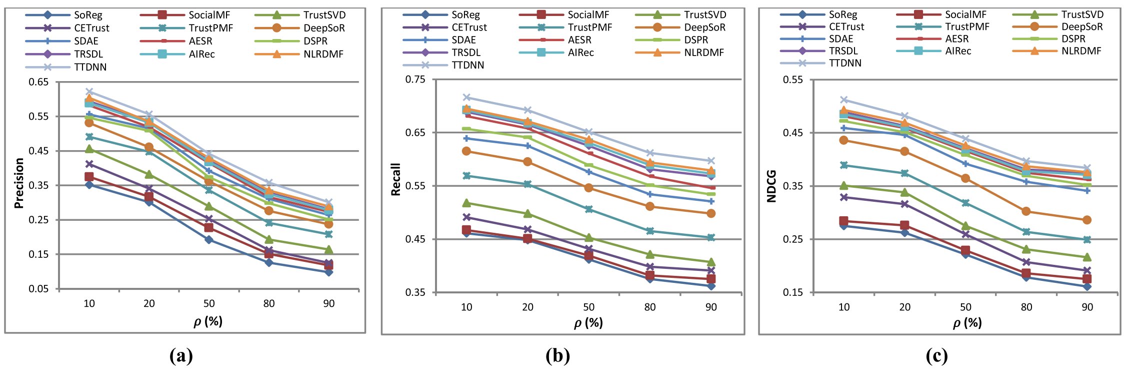

Fig. 5. The results of experiments based on Last.fm dataset for different sparsity levels (ρ): (a) Precision metric, (b) Recall metric, (c) NDCG metric. The recommendation list length is set to $top\_N = 10$.

4.7. Impact of different sparsity levels

Recommender systems have often major challenge with data sparsity. Therefore, some experiments are considered here to evaluate the quality of the recommendation models in different sparsity levels. The sparsity level refers to the ratio of data used as the training set in the recommendation approaches. In other words, we use the sparsity level as a measure to determine how much data is used in the training process of the recommendation approaches. The sparsity level is inversely related to the amount of data used in the training process. Therefore, the lower sparsity level shows that a higher amount of data is used in the training process. To this end, different sparsity levels as $\rho = 10\%, 20\%, 50\%, 80\%, 90\%$ are considered where $\rho = 20\%$ indicates that the training set is obtained by selecting 80% of all input data and the test set comprises the remaining data. Fig. 5 shows the results based on precision, recall, and NDCG metrics and different sparsity levels for Last.fm dataset. As it is seen, the performance of all algorithms declines when the sparsity level is increased. It is expected as increasing the sparsity levels leads to consider less data in the training process. These results also show that the proposed method obtains the highest recommendation

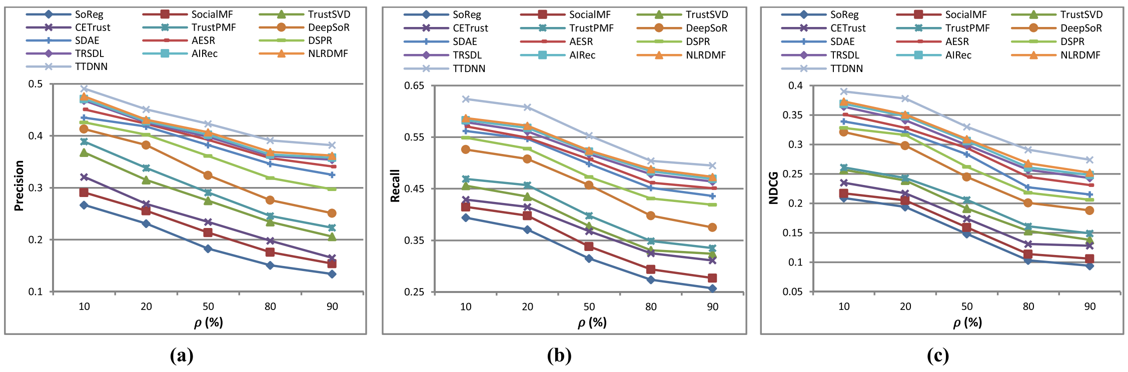

Fig. 6. The results of experiments based on Delicious dataset for different sparsity levels (ρ): (a) Precision metric, (b) Recall metric, (c) NDCG metric. The recommendation list length is set to $top\_N = 10$.

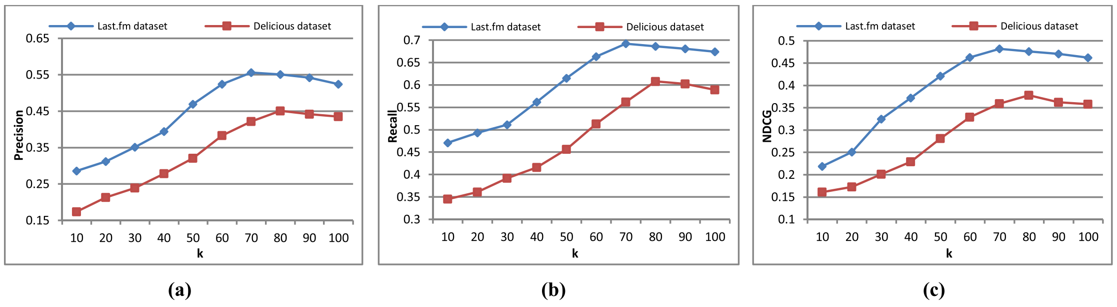

Fig. 7. The results of experiments based on Last.fm and Delicious datasets for different number of latent features (k): (a) Precision metric, (b) Recall metric, (c) NDCG metric. The recommendation list length is set to $top\_N = 10$.

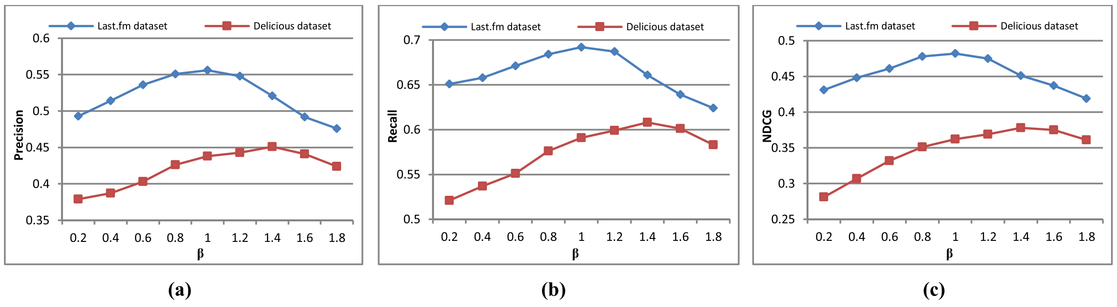

Fig. 8. The results of experiments based on Last.fm and Delicious datasets for different weights (β) in the combined similarity: (a) Precision metric, (b) Recall metric, (c) NDCG metric. The recommendation list length is set to $top\_N = 10$.

quality for overall sparsity levels. The experiments are repeated for Delicious dataset and their results are shown in Fig. 6. As we can see from this figure, the values of precision, recall, and NDCG metrics decrease when the sparsity level increases. Also, the proposed method outperforms other recommenders according to different sparsity levels. Thus, these results clearly demonstrate the advantage of the proposed method in alleviating the data sparsity problem.

4.8. Impact of the number of latent features

In this section, the impact of the number of latent features (k) on the performance of the proposed method is investigated by performing some experiments. It is worth noting that the number of latent features is one of the critical parameters used in the proposed method. Fig. 7 shows the experimental results based on the different number of latent features for both Last.fm and Delicious datasets. As we can see from these results, the precision, recall, and NDCG values increase in most cases by increasing the number of latent features. Thus, increasing the number of extracted latent features improves the quality of recommendations in the proposed method. However, these results show that the performance of the proposed method declines after a threshold value of the number of latent features. For example, precision, recall, and NDCG values decline when the number of latent features is higher than 70 for the Last.fm dataset. Also, when the number of latent features exceeds 80, the values of precision, recall, and NDCG metrics decrease for the Delicious dataset. Based on these performed experiments, the values of $k = 70$ and $k = 80$ are respectively used for the Last.fm and Delicious datasets as the optimal values for the proposed method.

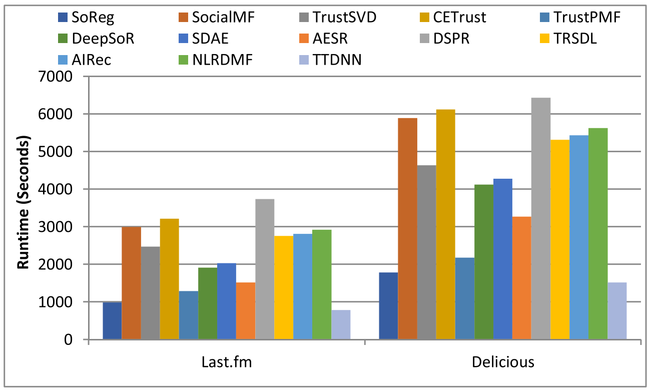

Fig. 9. Runtime comparison of recommendation models for Last.fm and Delicious datasets.

4.9. Impact of different weights in the combined similarity

Some experiments are conducted in this section to investigate the effect of different weights of trust- and tag-based similarity values in the combined similarity function in Eq. (17). According to Eq. (17), β controls the effect of trust- and tag-based similarity values in calculating the combined similarity value. Fig. 8 illustrates the performance of the proposed method in terms of different values of β. These results show that the proposed method achieves the best performance for the Last.fm dataset based on all evaluation metrics when $\beta = 1$. In other words, the best performance for the Last.fm dataset is achieved when the trust- and tag-based similarities have the same importance in calculating the combined similarity function. On the other hand, the best values of precision, recall, and NDCG metrics based on the Delicious dataset are obtained for the proposed method when $\beta = 1.4$. According to Eq. (17), $\beta > 1$ indicates that the tag-based similarity has a higher weight than the trust-based similarity. Therefore, in the case of the Delicious dataset, the best performance is obtained for the proposed method when the tag-based similarity is more important than the trust-based similarity in the combined similarity function.

4.10. Runtime comparison of recommendation models

In this section, the runtimes of different recommendation models are evaluated. Fig. 9 shows the runtimes of all compared recommendation models for Last.fm and Delicious datasets. As we can see from this figure, the proposed method has the lowest runtime for both Last.fm and Delicious datasets in comparison to other methods. Therefore, the proposed model obtains the best time computational complexity among all recommendation models. Moreover, SoReg is the second-best performer while DSPR obtains the worst runtime among the compared models. It is worth noting that the proposed method uses the extracted latent features instead of original input matrices to calculate similarity values between users. As the dimensionality of the latent features is much lower than that of original input matrices, therefore calculating the similarity values according to these latent features has a positive effect on decreasing the time computational complexity.

5. Conclusions

Considering additional side information, such as trust relationships and tag data can significantly improve the performance of RSs especially in dealing with data sparsity problem. However, users mainly generate insufficient information about these sources, and trust and tag data are often heavily sparser than user-item ratings data. In addition, there are generally numerous users, items, and tags in recommendation systems leading to form corresponding matrices with large dimensions. Therefore, calculating similarity values between users for the recommendation process can be a task with high computational complexity. This paper proposed an effective recommendation method by utilizing deep neural networks to learn latent features of trust relationships and user-tag matrices. These features are used as main inputs of the recommendation process to predict unseen ratings and provide recommendations to users. To this end, three different scenarios have been used to measure similarity values based on the latent features extracted by trust relations and tag information. It is worth noting that the dimensionality of latent features is much lower than the dimensionality of original trust and tag matrices. Therefore, calculating similarity values between users based on the extracted latent features has lower computational complexity than the approach that uses the original trust and tag matrices. Moreover, data sparsity problem can be addressed by calculating similarity values based on the latent features instead of using the original sparse matrices. The results of performed experiments demonstrated the superiority of the proposed method in comparison to others especially for alleviating the data sparsity problem. Other side information such as distrust relationships between users and review information can be used to improve the proposed method as future works. Also, integrating the proposed deep learning-based method with matrix factorization models can be useful to improve the recommendation accuracy. In addition, other models like convolution neural networks can be applied to learn the representation of latent features.

CRediT authorship contribution statement

Sajad Ahmadian: Conceptualization, Methodology, Writing – original draft. **Milad Ahmadian:** Conceptualization, Methodology, Writing – original draft. **Mahdi Jalili:** Conceptualization, Writing – review & editing, Investigation.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

Australian Research Council supported this research through project No. DP170102303.

References

[1] S. Ahmadian, M. Meghdadi, M. Afsharchi, Incorporating reliable virtual ratings into social recommendation systems, Appl. Intell. 48 (2018) 4448–4469.

[2] F. Rezaeimehr, P. Moradi, S. Ahmadian, N.N. Qader, M. Jalili, TCARS: Time-and community-aware recommendation system, Fut. Gener. Comput. Syst. 78 (2018) 419–429.

[3] Z. Zhang, G. Xu, P. Zhang, Y. Wang, Personalized recommendation algorithm for social networks based on comprehensive trust, Appl. Intell. 47 (3) (2017) 659–669.

[4] H. Ficel, M.R. Haddad, H.B. Zghal, A graph-based recommendation approach for highly interactive platforms, Expert Syst. Appl. 185 (2021) 115555.

[5] S. Ahmadian, N. Joorabloo, M. Jalili, M. Meghdadi, M. Afsharchi, and Y. Ren, "A temporal clustering approach for social recommender systems," in *International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*, Barcelona, Spain, 2018, pp. 1139-1144.

[6] M. Jalili, S. Ahmadian, M. Izadi, P. Moradi, M. Salehi, Evaluating collaborative filtering recommender algorithms: A survey, IEEE Access 6 (2018) 74003–74024.

[7] H. Tang, G. Zhao, X. Bu, X. Qian, Dynamic evolution of multi-graph based collaborative filtering for recommendation systems, Knowl.-Based Syst. 228 (2021) 107251.

[8] H. Zhang, Y. Sun, M. Zhao, T.W.S. Chow, Q.M.J. Wu, Bridging user interest to item content for recommender systems: An optimization model, IEEE Trans. Cybernetics (2019) 1–13.

[9] Q. Zhao, C. Wang, P. Wang, M. Zhou, C. Jiang, A novel method on information recommendation via hybrid similarity, IEEE Trans. Syst. Man Cybernet. Syst. 48 (3) (2018) 448–459.

[10] Y. Zhang, Z. Liu, C. Sang, Unifying paragraph embeddings and neural collaborative filtering for hybrid recommendation, Appl. Soft Comput. 106 (2021) 107345.

[11] S. Ahmadian, M. Afsharchi, M. Meghdadi, A novel approach based on multi-view reliability measures to alleviate data sparsity in recommender systems, Multimedia Tools Appl. 78 (2019) 17763–17798.

[12] F. Tahmasebi, M. Meghdadi, S. Ahmadian, K. Valiallahi, A hybrid recommendation system based on profile expansion technique to alleviate cold start problem, Multimedia Tools Appl. 80 (2021) 2339–2354.

[13] S. Ahmadian, N. Joorabloo, M. Jalili, M. Ahmadian, Alleviating data sparsity problem in time-aware recommender systems using a reliable rating profile enrichment approach, Expert Syst. Appl. 187 (2022) 115849.

[14] M. Jamali and M. Ester, "A matrix factorization technique with trust propagation for recommendation in social networks," in *RecSys '10 Proceedings of the fourth ACM conference on Recommender systems*, Barcelona, Spain, 2010, pp. 135-142.

[15] S. Ahmadian, N. Joorabloo, M. Jalili, Y. Ren, M. Meghdadi, M. Afsharchi, A social recommender system based on reliable implicit relationships, Knowl.-Based Syst. 192 (2020) 1–17.

[16] D. Rafailidis and F. Crestani, "Recommendation with social relationships via deep learning," in *Proceedings of the ACM SIGIR International Conference on Theory of Information Retrieval*, Amsterdam, The Netherlands, 2017, pp. 151-158.

[17] E. Zheng, G.Y. Kondo, S. Zilora, Q.i. Yu, Tag-aware dynamic music recommendation, Expert Syst. Appl. 106 (2018) 244–251.

[18] D. Rafailidis, P. Daras, The TFC model: tensor factorization and tag clustering for item recommendation in social tagging systems, IEEE Trans. Syst. Man Cybernet. Syst. 43 (3) (2013) 673–688.

[19] B. Yang, Y. Lei, J. Liu, W. Li, Social collaborative filtering by trust, IEEE Trans. Pattern Anal. Mach. Intell. 39 (2016) 1633–1647.

[20] G. Guo, J. Zhang, N. Yorke-Smith, TrustSVD: collaborative filtering with both the explicit and implicit influence of user trust and of item ratings, in: Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, 2015, pp. 123–129.

[21] M. Eirinaki, M.D. Louta, I. Varlamis, A trust-aware system for personalized user recommendations in social networks, IEEE Trans. Syst. Man. Cybernet. Syst. 44 (4) (2014) 409–421.
[22] P. Moradi, S. Ahmadian, F. Akhlaghian, An effective trust-based recommendation method using a novel graph clustering algorithm, Physica A 436 (2015) 462–481.
[23] H. Movahedian, M.R. Khayyambashi, A semantic recommender system based on frequent tag pattern, Intell. Data Anal. 19 (1) (2015) 109–126.
[24] Y.i. Zuo, J. Zeng, M. Gong, L. Jiao, Tag-aware recommender systems based on deep neural networks, Neurocomputing 204 (2016) 51–60.
[25] D. Rafailidis, A. Nanopoulos, Modeling users preference dynamics and side information in recommender systems, IEEE Trans. Syst. Man. Cybernet. Syst. 46 (6) (2016) 782–792.
[26] H.A. Rahmani, M. Aliannejadi, S. Ahmadian, M. Baratchi, M. Afsharchi, F. Crestani, LGLMF: local geographical based logistic matrix factorization model for POI recommendation, in: Asia Information Retrieval Symposium, 2019, pp. 66–78.
[27] S. Ahmadian, P. Moradi, F. Akhlaghian, An improved model of trust-aware recommender systems using reliability measurements, in: 6th Conference on Information and Knowledge Technology (IKT), Shahrood, Iran, 2014, pp. 98–103.
[28] P. Moradi, F. Rezaimehr, S. Ahmadian, M. Jalili, A trust-aware recommender algorithm based on users overlapping community structure, in: Sixteenth International Conference on Advances in ICT for Emerging Regions (ICTer), 2016, 2016, pp. 162–167.
[29] P. Moradi, S. Ahmadian, A reliability-based recommendation method to improve trust-aware recommender systems, Expert Syst. Appl. 42 (2015) 7386–7398.
[30] Y. Liu, L. Han, Z. Gou, Y. Yang, Personalized recommendation via trust-based diffusion, IEEE Access 7 (2019) 94195–94204.
[31] M. Ghavipour, M.R. Meybodi, Stochastic trust network enriched by similarity relations to enhance trust-aware recommendations, Appl. Intell. 49 (2) (2019) 435–448.
[32] T. Yu, J. Guo, W. Li, H.J. Wang, L. Fan, Recommendation with diversity: An adaptive trust-aware model, Decis. Support Syst. 123 (2019) 113073, https://doi.org/10.1016/j.dss.2019.113073.
[33] S. Ahmadian, M. Meghdadi, M. Afsharchi, A social recommendation method based on an adaptive neighbor selection mechanism, Inf. Process. Manage. 54 (4) (2018) 707–725.
[34] F.S. Gohari, F.S. Aliee, H. Haghighi, A dynamic local–global trust-aware recommendation approach, Electron. Commer. Res. Appl. 34 (2019) 100838, https://doi.org/10.1016/j.elerap.2019.100838.
[35] S. Ahmadian, M. Afsharchi, M. Meghdadi, An effective social recommendation method based on user reputation model and rating profile enhancement, J. Inform. Sci. 45 (5) (2019) 607–642.
[36] L. Guo, J. Liang, Y. Zhu, Y. Luo, L. Sun, X. Zheng, Collaborative filtering recommendation based on trust and emotion, J. Intell. Inform. Syst. 53 (1) (2019) 113–135.
[37] G. Guo, J. Zhang, F. Zhu, X. Wang, Factored similarity models with social trust for top-N item recommendation, Knowl.-Based Syst. 122 (2017) 17–25.
[38] G. Guo, J. Zhang, N. Yorke-Smith, A novel recommendation model regularized with user trust and item ratings, IEEE Trans. Knowl. Data Eng. 28 (2016) 1607–1620.
[39] Y. Liu, C. Liang, F. Chiclana, J. Wu, A knowledge coverage-based trust propagation for recommendation mechanism in social network group decision making, Appl. Soft Comput. 101 (2021) 107005.
[40] J. Guo, Y. Zhou, P. Zhang, B. Song, C. Chen, Trust-aware recommendation based on heterogeneous multi-relational graphs fusion, Inform. Fusion 74 (2021) 87–95.
[41] A. Ahmed, K. Saleem, O. Khalid, U. Rashid, On deep neural network for trust aware cross domain recommendations in E-commerce, Expert Syst. Appl. 174 (2021) 114757.
[42] G. Ma, Y. Wang, X. Zheng, X. Miao, Q. Liang, A trust-aware latent space mapping approach for cross-domain recommendation, Neurocomputing 431 (2021) 100–110.
[43] W. Wang, J. Chen, J. Wang, J. Chen, J. Liu, Z. Gong, Trust-enhanced collaborative filtering for personalized point of interests recommendation, IEEE Trans. Ind. Inf. 16 (2020) 6124–6132.
[44] Z. Hu, G. Xu, X.i. Zheng, J. Liu, Z. Li, Q.Z. Sheng, W. Lian, H. Xian, SSL-SVD: Semi-supervised learning-based sparse trust recommendation, ACM Trans. Internet Technol. 20 (1) (2020) 1–20.
[45] H. Chen, S. Wang, N. Jiang, Z. Li, N.a. Yan, L. Shi, Trust-aware generative adversarial network with recurrent neural network for recommender systems, Int. J. Intell. Syst. 36 (2) (2021) 778–795.
[46] J. Zhao, W. Wang, Z. Zhang, Q. Sun, H. Huo, L. Qu, et al., TrustTF: A tensor factorization model using user trust and implicit feedback for context-aware recommender systems, Knowl.-Based Syst. 209 (2020) 106434.
[47] T. Zhu, G. Li, W. Zhou, P. Xiong, C. Yuan, Privacy-preserving topic model for tagging recommender systems, Knowl. Inf. Syst. 46 (1) (2016) 33–58.
[48] N. Mauro, L. Ardissono, Extending a tag-based collaborative recommender with co-occurring information interests, in: Proceedings of the 27th ACM Conference on User Modeling, Adaptation and Personalization, Larnaca, Cyprus, 2019, pp. 181–190.
[49] H. Han, M. Huang, Y. Zhang, U.A. Bhatti, An extended-tag-induced matrix factorization technique for recommender systems, Information 9 (2018) 1–15.
[50] L. Luo, H. Xie, Y. Rao, F.L. Wang, Personalized recommendation by matrix co-factorization with tags and time information, Expert Syst. Appl. 119 (2019) 311–321.
[51] Q. Xie, F. Xiong, T. Han, Y. Liu, L. Li, Z. Bao, Interactive resource recommendation algorithm based on tag information, World Wide Web 21 (2018) 1655–1673.
[52] B.o. Chen, Y. Ding, X. Xin, Y. Li, Y. Wang, D. Wang, AIRec: Attentive intersection model for tag-aware recommendation, Neurocomputing 421 (2021) 105–114.
[53] S. Banerjee, P. Banjare, B. Pal, M. Jenamani, A multistep priority-based ranking for top-N recommendation using social and tag information, J. Ambient Intell. Hum. Comput. 12 (2021) 2509–2525.
[54] A. Nanopoulos, Item recommendation in collaborative tagging systems, IEEE Trans. Syst. Man Cybern. Part A: Syst. Humans 41 (4) (2011) 760–771.
[55] Z. Guan, C. Wang, J. Bu, C. Chen, K. Yang, D. Cai, et al., Document recommendation in social tagging services, in: Proceedings of the 19th international conference on World Wide Web, 2010, pp. 391–400.
[56] W. Zhao, Z. Guan, Z. Liu, Ranking on heterogeneous manifolds for tag recommendation in social tagging services, Neurocomputing 148 (2015) 521–534.
[57] S. Agrawal, D. Roy, M. Mitra, Tag embedding based personalized point of interest recommendation system, Inf. Process. Manage. 58 (2021) 102690.
[58] E. Quintanilla, Y. Rawat, A. Sakryukin, M. Shah, M. Kankanhalli, Adversarial learning for personalized tag recommendation, IEEE Trans. Multimedia 23 (2020) 1083–1094.
[59] R. Huang, N. Wang, C. Han, F. Yu, L.i. Cui, TNAM: A tag-aware neural attention model for Top-N recommendation, Neurocomputing 385 (2020) 1–12.
[60] S. Zhang, L. Yao, A. Sun, Y.i. Tay, Deep learning based recommender system: A survey and new perspectives, ACM Comput. Surv. 52 (1) (2019) 1–38.
[61] S.M.J. Jalali, M. Ahmadian, S. Ahmadian, A. Khosravi, M. Alazab, S. Nahavandi, An oppositional-Cauchy based GSK evolutionary algorithm with a novel deep ensemble reinforcement learning strategy for COVID-19 diagnosis, Appl. Soft Comput. 111 (2021) 107675.
[62] S.M.J. Jalali, S. Ahmadian, A. Khosravi, M. Shafie-khah, S. Nahavandi, J.P.S. Catalão, A novel evolutionary-based deep convolutional neural network model for intelligent load forecasting, IEEE Trans. Ind. Inf. 17 (2021) 8243–8253.
[63] S.M.J. Jalali, S. Ahmadian, M. Khodayar, A. Khosravi, V. Ghasemi, M. Shafie-khah, et al., Towards novel deep neuroevolution models: chaotic levy grasshopper optimization for short-term wind speed forecasting, Eng. Comput. (2021) 1–25.
[64] S.M.J. Jalali, S. Ahmadian, A. Kavousi-Fard, A. Khosravi, S. Nahavandi, Automated deep CNN-LSTM architecture design for solar irradiance forecasting, in: IEEE Transactions on Systems, Man, and Cybernetics: Systems, 2021, pp. 1–12.
[65] S. M. J. Jalali, M. Khodayar, S. Ahmadian, M. Shafie-khah, A. Khosravi, S. M. S. Islam, *et al.*, "A new ensemble reinforcement learning strategy for solar irradiance forecasting using deep optimized convolutional neural network models," in *2021 International Conference on Smart Energy Systems and Technologies (SEST)*, Vaasa, Finland, 2021, pp. 1-6.
[66] J. Liu, C. Wu, Deep learning based recommendation: A Survey, in: International Conference on Information Science and Applications, Macau, China, China, Macau, 2017, pp. 451–458.
[67] M. He, Q. Meng, S. Zhang, Collaborative additional variational autoencoder for top-N recommender systems, IEEE Access 7 (2019) 5707–5713.
[68] W. Yan, D. Wang, M. Cao, J. Liu, Deep auto encoder model with convolutional text networks for video recommendation, IEEE Access 7 (2019) 40333–40346.
[69] Y. Guan, Q. Wei, G. Chen, Deep learning based personalized recommendation with multi-view information integration, Decis. Support Syst. 118 (2019) 58–69.
[70] W. Zhang, X. Zhang, H. Wang, D. Chen, A deep variational matrix factorization method for recommendation on large scale sparse dataset, Neurocomputing 334 (2019) 206–218.
[71] J. Ni, Z. Huang, J. Cheng, S. Gao, An effective recommendation model based on deep representation learning, Inf. Sci. 542 (2021) 324–342.
[72] A. Da'u, N. Salim, R. Idris, An adaptive deep learning method for item recommendation system, Knowl.-Based Syst. 213 (2021) 106681.
[73] W. Zhang, Y. Du, T. Yoshida, Y. Yang, DeepRec: A deep neural network approach to recommendation with item embedding and weighted loss function, Inf. Sci. 470 (2019) 121–140.
[74] M. Fu, H. Qu, Z. Yi, L. Lu, Y. Liu, A novel deep learning-based collaborative filtering model for recommendation system, IEEE Trans. Cybern. 49 (2018) 1084–1096.
[75] H. Wu, Z. Zhang, K. Yue, B. Zhang, J. He, L. Sun, Dual-regularized matrix factorization with deep neural networks for recommender systems, Knowl.-Based Syst. 145 (2018) 46–58.
[76] N. Liang, H. Zheng, J. Chen, A. Sangaiah, C. Zhao, TRSDL: tag-aware recommender system based on deep learning–intelligent computing systems, Applied Sciences 8 (2018) 1–15.
[77] J. Wei, J. He, K. Chen, Y. Zhou, Z. Tang, Collaborative filtering and deep learning based recommendation system for cold start items, Expert Syst. Appl. 69 (2017) 29–39.
[78] Z.H. Deng, L. Huang, C.D. Wang, J.H. Lai, P.S. Yu, DeepCF: A unified framework of representation learning and matching function learning in recommender system, in: The thirty-third AAAI conference on Artificial Intelligence, AAAI 2019, Honolulu, Hawaii, USA, 2019, pp. 61–68.
[79] W.D. Xi, L. Huang, C.D. Wang, Y.Y. Zheng, J. Lai, "BPAM,, Recommendation based on BP neural network with attention mechanism, in: Proceedings of the

twenty-eighth International Joint Conference on Artificial Intelligence (IJCAI-19), pp. 3905–3911.
[80] H.J. Xue, X. Dai, J. Zhang, S. Huang, J. Chen, Deep matrix factorization models for recommender systems, in: Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence (IJCAI-17), 2017, pp. 3203–3209.
[81] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T. S. Chua, "Neural collaborative filtering," in *Proceedings of the 26th International Conference on World Wide Web (WWW '17)*, 2017, pp. 173–182.
[82] A. Makhzani and B. Frey, "K-sparse Autoencoders," *arXiv:1312.5663,* 2013.
[83] Y. Bengio, A. Courville, P. Vincent, Representation learning: A review and new perspectives, IEEE Trans. Pattern Anal. Mach. Intell. 35 (2013) 1798–1828.
[84] D.C. Liu, J. Nocedal, On the limited memory BFGS method for large scale optimization, Math. Program. 45 (1-3) (1989) 503–528.
[85] N. Jiang, W. Rong, B. Peng, Y. Nie, Z. Xiong, An empirical analysis of different sparse penalties for autoencoder in unsupervised feature learning, in: International Joint Conference on Neural Networks (IJCNN), Killarney, Ireland, 2015, pp. 1–8.
[86] H. Lee, A. Battle, R. Raina, A.Y. Ng, Efficient sparse coding algorithms, in: Proceedings of 20th Annual Conference on Neural Information Processing Systems, 2006, pp. 801–808.
[87] G.E. Hinton, R.R. Salakhutdinov, Reducing the dimensionality of data with neural networks, Science 313 (5786) (2006) 504–507.
[88] H. Ma, D. Zhou, C. Liu, M.R. Lyu, I. King, Recommender systems with social regularization, in: Proceedings of the fourth ACM international conference on Web search and data mining, Hong Kong, China, 2011, pp. 287–296.
[89] W. Fan, Q. Li, M. Cheng, Deep modeling of social relations for recommendation, in: The Thirty-Second AAAI Conference on Artificial Intelligence (AAAI-18), New Orleans, Louisiana, USA, 2018, pp. 8075–8076.
[90] N.CC, A. Mohan, A social recommender system using deep architecture and network embedding, Appl. Intell. 49 (2019) 1937–1953.
[91] Z. Xu, T. Lukasiewicz, C. Chen, Y. Miao, X. Meng, Tag-aware personalized recommendation using a hybrid deep model, in: Proceedings of the 26th International Joint Conference on Artificial Intelligence, 2017, pp. 3196–3202.
[92] L. Wan, F. Xia, X. Kong, C.-H. Hsu, R. Huang, J. Ma, Deep matrix factorization for trust-aware recommendation in social networks, IEEE Trans. Network Sci. Eng. 8 (1) (2021) 511–528.

Sajad Ahmadian received his B.S. degree in computer engineering from Razi University, Kermanshah, Iran, in 2011, and the M.Sc. degree in computer engineering, artificial intelligence from University of Kurdistan, Sanandaj, Iran, in 2014. Moreover, he has been received his Ph.D. degree in computer engineering, artificial intelligence from University of Zanjan, Zanjan, Iran, in 2018. He conducted a part of his Ph.D. research work in the laboratory of complex networks, RMIT University, Melbourne, Australia, form February 2018 to July 2018. He is currently an Assistant Professor at the Faculty of Information Technology, Kermanshah University of Technology, Kermanshah, Iran. His research interests include recommender systems, social networks analysis, data mining, and machine learning.

Milad Ahmadian is currently an M.Sc. student in computer engineering at the Department of Computer Engineering, Razi University, Kermanshah, Iran. He received his B.S. degree in Computer Engineering from Razi University, Kermanshah, Iran, in 2019. His research interests include recommender systems and deep learning.

Mahdi Jalili received the Ph.D. degree in computer and communications sciences from the Swiss Federal Institute of Technology Lausanne, Lausanne, Switzerland, in 2008. He is currently an Associate Professor at the School of Engineering, RMIT University, Australia. He was previously an Australian Research Council DECRA Fellow and RMIT Vice-Chancellor Research Fellow. He is an Associate Editor of Complex Adaptive Systems Modeling, Complexity and Mathematical Problems in Engineering. His current research interests include network science and engineering.