Hybrid high-order semantic graph representation learning for recommendations

Canta Zheng¹ · Wenming Cao¹

Received: 11 June 2021 / Accepted: 10 August 2021

Published online: 19 August 2021

© The Author(s) 2021

OPEN

Abstract

The amount of Internet data is increasing day by day with the rapid development of information technology. To process massive amounts of data and solve information overload, researchers proposed recommender systems. Traditional recommendation methods are mainly based on collaborative filtering algorithms, which have data sparsity problems. At present, most model-based collaborative filtering recommendation algorithms can only capture first-order semantic information and cannot process high-order semantic information. To solve the above issues, in this paper, we propose a hybrid graph neural network model based on heterogeneous graphs with high-order semantic information extraction, which models users and items respectively by learning low-dimensional representations for them. We introduced an attention mechanism to allow the model to understand the corresponding edge weights adaptively. Simultaneously, the model also integrates social information in the data to learn more abundant information. We performed some experiments on related datasets. Our method achieved better results than some current advanced models, which verified the proposed model's effectiveness.

Keywords Recommendation · GNN · Social network · High-order semantic

1 Introduction

The recommendation system aims to provide users with product information and suggestions, help users decide what products to buy, and simulate sales staff to help customers complete the purchase process through the Internet. The emergence and popularization of the Internet have brought a large amount of information to users, meeting users' needs in the information age. However, the amount of online news has increased significantly, and the scale of e-commerce has continued to expand. When faced with a large amount of information, users cannot obtain useful information and the efficiency of the use of information is reduced, which is called an information overload problem. Customers need to spend a lot of time to find the products that they want to buy, causing the loss of consumers submerged in information overload. Therefore, a personalized recommendation system came into being, which is an advanced and smart business platform based on massive data mining to help e-commerce websites provide completely personalized decision support and information services for their customers. It is also a customized information recommendation system that provides information and products of interest for users according their information needs and interests. Compared with search engines, the recommendation system conducts personalized calculations by studying the user's interest preferences. Besides, it discovers users' interest, thereby guiding them to find their own information needs. A sound recommendation

✉ Wenming Cao, wmcao@szu.edu.cn; Canta Zheng, 1900432031@email.szu.edu.cn | ¹College of Electronics and Information Engineering, Shenzhen University, Shenzhen 518060, China.

system can provide users with personalized services and establish a close relationship with users, allowing users to rely on recommendations.

Current limitations: According to social networks or buying behaviors, the recommendation system provides accurate and fast services for users, such as friends, merchandise items or services, etc., and offers personalized content to each user to improve the experience and create for attracting new users. The recommendation system was first used by Amazon to recommend products. Suppose a service network has N users and M items and then the recommendation algorithm aims to fill in the missing items in the $N * M$ evaluation matrix when there are existing connections between users and items.

Traditional recommendation algorithms mainly include content-based methods and collaborative filtering methods [1], which have been already applied to major social service platforms. The latest research works mainly contain network embedding representation, deep learning and deep geometric learning based on graph neural networks. However, these methods currently have several shortcomings and limitations. Although some models have solved the cold start problem, they cannot be applied to large-scale data. Besides, the influence of social relations on product sales has been recognized a long time ago. Some researchers believe that social influence is even more important than the similarity of historical behavior information in the recommendation system. Integrating social network data in the recommendation system can improve the recommendation's performance because of additional information between users and users. In recent years, the explosion of graph deep learning bring new vigor and vitality into various research fields, such as anomaly detection [2], community detection [3, 4] and recommendation. Deep learning-based recommendation techniques can capture more accurate feature information for better performance. However, the current recommendation algorithms based on deep learning [5–7] can only capture first-order semantics information while ignoring the high-order semantic details of the data, causing the loss of data and thereby limiting the recommendation algorithm's performance.

In this work, we propose a hybrid neural network model based on heterogeneous and high-order semantic graphs to learn users' and items' representations, which can solve the problems mentioned above.

Rich semantic information: As is shown in Fig. 1, the recommended data of the e-commerce platform is an alternative information network, including two calculator types: user and item. The routine relationships include user–item and user–user. The heterogeneous data can be divided into two types of graphs: the bipartite graph and the social graph. Figure 1 are the user–item bipartite graph (Fig. 1-middle) and the user–user social graph (Fig. 1-right). For example, for u_3 and u_4 , both of them have purchased item i_4 for the friend relationship, then if other products i_2 and i_5 purchased by u_3 are the same types or related items as i_4 , u_4 may also be interested in these two items.

Multi-view and multi-layer aggregation: We have considered three perspectives for a recommendation system to learn effective user representations and item representations. They are the user-centric aggregation of item features, the user-centric aggregation of friend features, and the item-centric aggregation of user features, making full use of original

Fig. 1 (Left) user–item bipartite and social graph, (middle) user–item bipartite graph, (right) user social relationship graph

information. Besides, we also sample 2-order and 3-order semantic information and design a multi-layer aggregation method to deal with the high-order semantic information.

The contributions of our model are shown as follows:

- 1) We propose a method that can aggregate rich semantic information, realize multi-view and high-order information fusion, and emphasize high-order semantic's importance.
- 2) We propose a new recommendation model with hybrid graph neural network including homogeneous GNNs and heterogeneous GNNs, based on a high-order semantic and heterogeneous graph with an attention mechanism, which is used to model users' hobbies and consumption habits, item category features and product features. And it can be effectively applied on the large-scale data.
- 3) Many experiments have been carried out on two public datasets. The experimental results show that our proposed method, combined with high-order semantic information and hybrid graph neural network, can learn more effective representations than the best existing models in terms of metrics.

2 Related work

2.1 Recommendation

The technology used in traditional recommendation systems is usually based on content filtering and collaborative filtering [1]. On one hand, The content-based filtering [8] methods recommend new items according to their similarity to the items already in the user's profile. Therefore, it requires more detailed information about each item. On the other hand, collaborative filtering methods [9] are "agnostic" to the items. Instead, they use user-assigned ratings to find other similar users (or items) based on their rating patterns. The universal nature of collaborative filtering systems is the reason for their widespread success because they recommend various products, such as movies, music, news, books, research articles, search queries, social tags, etc.

Generally speaking, collaborative filtering methods are divided into three types. The first is user-based collaborative filtering, the second is item-based collaborative filtering and the third is model-based collaborative filtering. User-based collaborative filtering mainly considers the similarity between users and users. As long as you find items that similar users like, and predict the target user's rating on the corresponding object, you can find the highest scores and recommend items to the user. Item-based collaborative filtering is similar to user-based collaborative filtering, except that we turn to see the similarity between items and items. Only when the target user's rating of specific items is found can we predict similar items with high similarity and recommend several similar items with the user's highest scores.

The classic collaborative filtering method mainly uses matrix factorization. For example, MF [10] learns the latent representation of users and items in the public subspace by decomposing the user-item evaluation matrix. As a baseline for comparison in many researches, it is a typical latent factor model. SVD++ [11] uses the users' specific ratings on items to model user preferences and is an improved algorithm of SVD. PMF [12] uses the user-item evaluation matrix to carry out probability matrix decomposition and uses Gaussian distribution to model users' and items' potential factors. Some are combined with trust information. For instance, SoRec [13] factorizes the user-item evaluation matrix and the user-user social relationship matrix while SoReg [14] uses social network information as a regularization term to constrain the matrix factorization framework. SocialMF [15] takes trust information and the spread of trust information into the recommendation system's matrix factorization model. TrustMF [16] optimizes user embedding to retrieve the trust matrix. TrustSVD [17] integrates the embedding vector of the target user's friends into the predicted score. The TrustMF [18] method uses matrix factorization technology to decompose the trusted network according to trust and maps users to two low-dimensional spaces: the truster space and the trustee space. In 2017, some researchers applied neural networks in recommendations. The NeuMF [19] method is a matrix factorization model with a neural network structure. Instead of its original implementations to perform the recommendation ranking task, the authors adjusted its loss to square loss for rating prediction to get better performance.

In recent years, researchers in the recommendation field have turned their attention to the area of deep learning. Compared with traditional recommendation algorithms, social recommendation algorithms based on graph neural networks have significant performance advantages and development potential. Monti et al. proposed the first GCN-based recommendation system method [20]. In their approach, GCN is used to aggregate information from two auxiliary users-user and item-item graphs. After each step of aggregation, the potential factors of users and items are updated,

and the model is trained by combining the objective functions of GCN and MF. Later, Berg et al. proposed the GC-MC model [5]. GC-MC directly described the relationship between users and items as interactive bipartite graphs, using two multi-link graph convolutional layers to aggregate user features and item features and estimating the ratings by predicting edge labels. Due to GCN's strong ability to learn high-quality user and item representation, GC-MC has achieved the most advanced performance in several public recommendation benchmarks in 2017.

Later in 2019, Wenqi Fan et al. proposed a graph neural network model GraphRec [6], based on social recommendation, which consists of three parts: the user modeling, the item modeling, and the rating prediction. The first component is user modeling, with the purpose of understanding the latent factors of the user. Since the social recommendation system data includes two different graphs, namely a social graph and a user–item bipartite graph, two aggregation types are used to learn factors from the two graphs. The first aggregation is called item aggregation and utilized to understand the users' latent characteristics in the item space from the user–item bipartite graph. The second type is social aggregation, and used to learn the users' latent characteristics in the social space from the social graph. Then these two factors are combined to form the final user latent factor. And the item modeling do the same aggregation as the first aggregation in the user modeling. Qitian Wu et al. proposed the DANSER model [7], which also believes that users' relationship plays a larger role in the recommendation system and the relationship between recommended items can take effect. It models the user–user and the item–item effects to capture the related information to improve the recommendation system's effectiveness. More and more researchers have been studying recommendations with graph neural networks.

Besides, there are some interesting related recommendation tasks, including Collaborative dynamic sparse topic regression, Collaborative social group influence for event recommendation and so on. In many time-aware recommendation systems, it is essential to model the accurate evolution of both user profiles and the contents of items over time. The work [21] a time-aware item recommendation method call CDUE, which captures the evolution of both user s interests and item s contents information via topic dynamics, in which the item's topics and user's interests and their evolutions are learned collaboratively. The work [22] consider the influence of social groups and propose a new Bayesian latent factor model, which combines social influence and individual preference for event recommendations.

2.2 Graph neural network

In 2005, Marco Gori et al. first proposed the graph neural network concept [23], which improved the ability to process graph data efficiently and could directly apply the problem's learning processing to graph data. Subsequently in 2009, Micheli A et al. [24] and Scarnli F et al. [25] further elaborated on graph neural networks and proposed a method based on supervised learning to train GNN. The early GNNs used iterative methods to learn, which was very expensive. Since 2012, the rapid development of convolutional neural networks has achieved breakthrough results in the field of computer vision. Hence, researchers began to consider how to combine graph data with convolutional neural networks. In 2013, Bruna et al. proposed the frequency domain convolution operation. They offered a graph convolution network model, which introduced the convolution operation into the graph neural network for the first time [26]. In 2016, Kipf et al. [27] simplified the definition of frequency-domain graph convolution, turning frequency domain graph convolution into spatial domain graph convolution, which significantly improved the graph convolution model's computational efficiency. Simultaneously, the graph convolution model has made great achievements in the study of graph data. Subsequently, Hamilton W et al. proposed the GraphSAGE model [28], which changed the original GCN from full graph training to node-centric small batch training by sampling neighbors, significantly reducing the training process's data size and the computational cost. Veličković et al. [29] proposed the graph attention network (GAT), which uses the attention mechanism to adaptively allocate different neighbors' weights to realize neighbor nodes' aggregation operation. However, overlaying multiple GNN layers will result in excessive smoothing and overfitting. To address this problem, some models have been proposed such as DeepGCNs [30] and DropEdge [31], which improve the performance in some datasets to a certain extent. Ming Chen et al. proposed a latest model called GCNII [32], providing theoretical and empirical evidence so that the two techniques effectively relieves the problem of oversmoothing. It has proved to be the state-of-art so far.

Various GNNs have been developed, greatly enhancing the adaptability of the model to graph data. The combination of deep neural networks and graph data has now become more and more popular. The data related to the recommendation system consists of a bipartite graph and a social graph, which is a typical heterogeneous information network and suitable to be modelled and processed with graph neural networks. The research of a recommendation system based on graph neural networks has been a trend.

3 The proposed method

We propose a hybrid high-order semantic graph neural network for recommendations (HHRec), regarded as an improved version of GraphRec. The overall model framework is shown in Fig. 2. It includes four parts:

- 1) Input layer: Sample the social subgraph and bipartite subgraph and then put the sampling nodes into the embedding layers, which include user embedding, item embedding, opinion embedding, and category embedding
- 2) User encoder: Model the user from two perspectives, and separately summarize the neighbor item feature and friend feature centered on the user to obtain two sub-embeddings, and then put them into a total user representation through MLP form.
- 3) Item encoder: Model the item from a single perspective, summarize neighbor user characteristics centered on the item. Then the encoder directly obtains the item representation.
- 4) Rating prediction layer: The model's output layer is used to predict the rating prediction r_{ij} of the user u_i on the item v_j .

3.1 Embedding layer

In deep learning, especially with the Internet field with the recommendation, advertising, and search as the core, embedding technology is widely used and regarded as the essential operation of deep learning. Formally speaking, embedding uses a low-dimensional dense vector to represent a user or an item. The low-dimensional dense vector can express the corresponding object's specific characteristics, and the distance between the vectors reflects the similarity between the objects. Since the birth of Word2vec [33], the embedding idea spreads from the field of natural language processing to almost all areas of machine learning quickly, including social recommendation systems. For users and items in the social recommendation system, there is also a corresponding embedding method. During the recommendation system training process, the embedding layer generally needs to work with other deep learning networks to complete the entire recommendation training process.

Embedding technology completes the conversion from high-dimensional sparse feature vectors to low-dimensional dense feature vectors. After connecting with other feature vectors, the pre-trained embedding feature vector is input into the training's neural network. Ours proposed model's input layer includes four input layers. We use an embedding vector $e_i \in \mathbb{R}^d$ ($e_j \in \mathbb{R}^d$) to describe a user or an item, where d represents the embedding dimension. Similarly, we also

Fig. 2 An overview of HHRec. The input information includes user-centric second-order subgraph, item-centric second-order subgraph, User-centric social-item subgraph

adopt another two embedding layers to represent the user's opinion score on the item and the item's category. That is to use a low-dimensional vector $e_r \in \mathbb{R}^d$ ($e_c \in \mathbb{R}^d$) to describe the edge weight of the user-item bipartite graph or the category in the item space. Therefore, assuming that the original data has N users, M items including L categories, and K evaluation levels, an embedding space based on the embedding matrix can be formed as:

$$\begin{aligned} E &= [W_u, W_i, W_r, W_c] \\ &= [e_{u1}, \dots, e_{uN}, e_{i1}, \dots, e_{iM}, \\ &\quad e_{r1}, \dots, e_{rK}, e_{c1}, \dots, e_{cL}] \end{aligned} \quad (1)$$

It is worth noting that when defining the model, we can initialize the embedding space randomly and then follow the entire model for end-to-end training. It is also considered that a pre-training step can be added to obtain the pre-trained embedding through item2vec and other methods, and then end-to-end training can be performed to get a more effective embedding.

3.2 High-order multi-view user and item encoder

The user encoder aims to learn a latent representation vector h_i of the user u_i , which is used to model the user and his friends' interests and hobbies. For a heterogeneous graph G composed of a bipartite graph and a social network, two types of nodes, item and user, are associated with the user node u_i . So we need to model users from two aspects. The first is to model the target user's interest and hobbies characteristics from the user-item interactions and learn the user representation in the item space $h_i^I \in \mathbb{R}^d$. The second is to model the friend characteristics of the target user from the user-user social interactions and learn the user representation $h_i^S \in \mathbb{R}^d$ in the social space. Finally, it combines the two representations to get the final user latent representation vector h_i .

The purpose of the item encoder is to learn a latent representation vector h_j of the item, which is used to model the item's category characteristics and audience level. It only needs to model items from one perspective. Model the category characteristics and audience level of the target item from the user-item diagram, and learn the potential representation vector h_j of the item in the userspace.

Here, we propose a multi-layer and high-order aggregation method. For user modeling, the sampling data path is user-item-user and social-item, and for item modeling, the sampling data path is item-user-item. We define two types of single-layer aggregation for a single-layer connection like the user-item and the item-user, called the item-view aggregation and the user-view aggregation. The item-view denotes user-centric single-layer aggregation of neighbor items, while the user-view denotes the item-centric single-layer aggregation of neighbor users. These two operations are introduced in detail respectively below (Fig. 3).

Item-view aggregation: In the user-item bipartite graph, the user and the item have a weighted connection relationship. The user's rating or opinion on the item represents the user-item connection weight. To deal with the weighted user-item bipartite graphs, we combine user-item interactions and their opinions to achieve a single-layer aggregation. The item-view

Fig. 3 Single-layer item-view aggregation (left) and user-view aggregation with an attention mechanism

aggregation is user-centered with items as neighbor nodes, and all neighbor node characteristics are aggregated to get the first-order potential representation of the center user. To describe this aggregation process mathematically, we use the following function:

$$h_i^k = \sigma(W \cdot \text{Aggre}(h_a^{k-1}, \forall a \in C(i)) + b) \quad (2)$$

Where k denotes the number of layers of aggregation, $C(i)$ denotes the set of all items interacting with user u_i , and h_a^{k-1} represents the potential representation of items related to user u_i in the $(k-1)$ th layer. W is the linear transformation weight, b is the weight bias, and σ is the nonlinear activation function (such as Relu, sigmoid, etc.). In particular, we define the potential representation h_i^0 of the central user u_i at the 0-th layer as the user's initial embedding e_i . And the latent representation h_j^0 of the neighbor item v_j at the 0-th layer is defined as a connection of the item's initial embedding e_j and its opinion embedding by the related users u_i . Introducing the user's opinion r on the item is conducive to model the user's interest and preference. So mathematically, we can define the potential representation h_j^0 of the item v_j at the 0-th layer as

$$h_j^0 = f_v([e_j \oplus e_r]) \quad (3)$$

where $\oplus$ represents the concat operation between two vectors, f_v represents a Multi Layer Perceptron (MLP), and the output is the user's potential representation h_j^0 in the 0-th under the user-view. In order to fully consider the influence of different neighbors, we add an attention mechanism [34] to perform weighted aggregation on the neighbors of the center user node. The attention weight can adjust by itself adaptively with model training. In particular, for the input h_a^{k-1} , $\forall a \in C(i)$, the attention-focused aggregation operation becomes

$$\text{Aggre}(\{h_a^{k-1}, \forall a \in C(i)\}) = \sum_{a \in C(i)} \alpha_{ia}^k h_a^{k-1} \quad (4)$$

where α_{ia} represents the attention weight between the center user u_i and the neighbor item v_a , which is used to measure the importance of each neighbor node to the center node u_i in the process of feature aggregation, to model the center user more effectively. To obtain practical attention values, we use a two-layer neural network to model attention generation. Making the potential representation of the center user h_i^{k-1} in the $(k-1)$ th layer and the potential representation of the neighbor item h_j^{k-1} in the $(k-1)$ th layer as input, the attention value can be defined as:

$$\alpha_{ia}^k = \sigma(W_2 \cdot \sigma(W_1 \cdot [h_i^{k-1} \oplus h_a^{k-1}] + b_1) + b_2) \quad (5)$$

Finally, we can get a normalized attention through a softmax function:

$$\alpha_{ia}^k = \frac{\exp(\alpha_{ia}^k)}{\sum_{b \in C(i)} \exp(\alpha_{ib}^k)} \quad (6)$$

In this way, we have completed a single-layer aggregation with users as the center nodes and items as the neighbor nodes.

User-view aggregation: Similarly, based on user-view aggregation, the user-item bipartite graph with opinion weight is also considered. The item is the center node, and the user is the neighbor node. Aggregating all the neighbor node features can get the center user's first-order latent representation. To describe this aggregation process mathematically, we use the following function:

$$h_j^k = \sigma(W \cdot \text{Aggre}(h_d^{k-1}, \forall d \in D(j)) + b) \quad (7)$$

where k represents the number of layers of aggregation, $D(i)$ represents the set of all users interacting with item v_j , h_d^{k-1} represents the potential representation of users associated with item v_j in the $(k-1)$ th layer, and W is the linear transformation weight, b is the weight bias, and σ is the nonlinear activation function (such as Relu, sigmoid, etc.). In particular, we define the potential representation h_j^0 of the center item v_j in the 0-th layer as the initial embedding e_j of the item. And the potential representation h_i^0 of the neighbor user u_i at the 0-th layer is defined as a connection of the initial embedding e_i of the user u_i and its opinion embedding on the related items v_j . Similarly, Introducing the user's opinion

r on the item is conducive to model the item's characteristics and audience level. Based on the user-view, mathematically, we define the potential representation of user u_i at the 0-th layer as

$$h_i^0 = f_v([e_i \oplus e_r]) \quad (8)$$

where $\oplus$ denotes the concat operation between two vectors, f_v represents a Multi Layer Perceptron (MLP), and the output h_i^0 is the potential representation of the user at the 0-th layer under the user-view. Like the item-view aggregation, we use the attention mechanism to perform adaptive weighted aggregation of the neighbors of the center item node. In particular, for the input, the attention-focused aggregation operation becomes

$$\text{Aggre}(\{h_d^{k-1}, \forall d \in D(j)\}) = \sum_{d \in D(j)} \beta_{jd}^k h_d^{k-1} \quad (9)$$

$$\beta_{jd}^k = \sigma(W_2 \cdot \sigma(W_1 \cdot [h_j^{k-1} \oplus h_d^{k-1}] + b_1) + b_2) \quad (10)$$

$$\beta_{jd}^k = \frac{\exp(\beta_{jd}^k)}{\sum_{b \in D(i)} \exp(\beta_{jb}^k)} \quad (11)$$

where β_{jd}^k denotes the normalized attention weight of the center item v_j and neighbor user u_d , which is used to measure the importance of each neighbor node to the center item node v_j in the process of feature aggregation, so as to make it more effectively model the center item.

The item-view aggregation and the user-view aggregation are based on the same idea, just with different center nodes.

We can perform high-order information aggregation and modeling processing on users and items after defining the item-view aggregation operation and the user-view aggregation operation. As Fig. 4 shows, taking a user as the center, we sample user-item-user subgraphs. Similarly, we sample the item-user-item subgraph, taking an item as the center. Besides, we need to consider social connections among users for user modeling. we sample social-item subgraphs for the social-view aggregation in the user modeling, which is shown in Fig. 5.

Fig. 4 User-centric second-order subgraph sampling (left), item-centric second-order subgraph sampling (right)

Fig. 5 User-centric social-item subgraph sampling, including three-order item information

Modeling the user: For high-order semantic information, we propose a high-order aggregation method as follows.

As Fig. 6 shows, 0-order nodes refer to the source nodes, which correspond to the layer-0 in the Fig. 4. To prevent semantic distortion, we implement a multi-layer skip connection, splicing the target node's features at each layer, and then get the output h_i^l through an MLP as follows.

Fig. 6 A multi-aggregator

$$h_i^l = MLP(h_i^0 \oplus h_i^1, \dots, \oplus h_i^n) \quad (12)$$

While n is 2, it turns to be

$$h_i^l = MLP(h_i^0 \oplus h_i^1 \oplus h_i^2) \quad (13)$$

Combined with social-view. For all user nodes in the social connections, as Fig. 5 shows, we use a-layer item-view aggregation to aggregate neighbor items. The information in the social-item subgraph includes the first-order information, the second-order information, and third-order information, corresponding to the layer-0-1, the layer-1-1, and the layer-2-1, respectively.

As is shown in Fig. 7, we implement a-layer item-view aggregation for all users in the social-item graph, which captures the first-order features in the social-item subgraph. Where h_{src-u}^0 denotes users' original embedding and h_{1-i}^0 denotes neighbor items' original embedding. Regarding the output feature as the new feature of all user nodes, we can use the isomorphic graph neural network to learn the nodes' representations for the social graph. Here we use graph attention network (GAT) [29] to get the output h_i^S from the social perspective finally.

Combining the vectors learned from the above two perspectives, the final representation vector of the user node can be obtained as

$$h_i = MLP(h_i^l, h_i^S) \quad (14)$$

Modeling the item: Similar to the user modeling, Fig. 6 is also used to model the item output. Similarly, we achieve multi-layer skip connections, splicing the target node's features at each layer, and then output through an MLP to get h_j as

$$h_j = MLP(h_j^0 \oplus h_j^1 \oplus h_j^2) \quad (15)$$

3.3 Rating prediction

After obtaining the user's representation and the item's representation, we need to realize the rating prediction and output the user's predicted rating value. A common approach is usually to perform dot product operations on the user representation and the item representation directly, which is

$$r_{ij} = h_i^T \cdot h_j \quad (16)$$

Or join the two together, and then enter a fully connected layer, that is

$$r_{ij} = MLP(h_i \oplus h_j) \quad (17)$$

Here we choose the latter one to make the model's rating prediction easier to learn.

Fig. 7 A social aggregator

3.4 The model training

To learn the model parameters, we need to define an objective function. Rating prediction is Essentially a regression task. Thus, we choose the MSE loss as the objective function, which is formulated as

$$\mathcal{L} = \frac{1}{2|O|} \sum_{i,j \in O} (r'_{ij} - r_{ij})^2 + \lambda \|\Theta\|_2^2 \quad (18)$$

It assumes that the observed interactions ought to be assigned higher prediction values than the unobserved ones. Where $|O|$ is the number of observed ratings in a mini-bench. r_{ij} is the ground truth rating assigned by the user u_i on the item v_j , and r'_{ij} is rating prediction output. λ denotes decay weight, and Θ denotes the parameters of the model. We use the RMSprop [35] optimizer in the training process, which randomly selects a training instance and updates each model parameter towards its negative gradient direction. At the beginning of the training, we initialize the model parameters with Xavier Initialization [36] instead of random Initialization. The basic idea of Xavier initialization is to keep the variance of input and output consistent to avoid all output values tending to zero and help for better convergence.

Overfitting is a widespread problem in deep learning, although it has a strong processing ability. To alleviate this problem, researchers have taken some measures into account. We applied Dropout [37] in our model. Dropout is an effective way to prevent the model from overfitting, with the thought of dropping some neurons during the training process randomly. When learning parameters, only part of them will update. Besides, we adopt a regularization term shown in Eq. 18, which can alleviate overfitting to a certain extent.

4 Experiment

We evaluate the proposed HHRec architecture on the two standard datasets. We need to answer the following four questions about our model:

- RQ1:* How does the HHRec for recommendations perform while comparing with the current advanced methods for recommendations?
- RQ2:* How do different hyper-parameter settings affect HHRec for recommendations?
- RQ3:* How do the latent representations reflect the user's interest preferences and the item's category characteristics and audience level?
- RQ4:* Are the critical components in HHRec, such as the attention mechanism and social information, necessary for better performance?

4.1 Datasets

To evaluate our proposed method, we perform multi-group experiments on the two standard product review datasets: Epinions and Ciao, which are publicly accessible for traditional research about trust in the physical world. An essential characteristic of product review sites such as Epinions is that there exist trust networks among users. Epinions and Ciao datasets are crawled by Jiliang Tang et al. [38] from two popular product review sites, Epinions(www.epinions.com) and Ciao (<http://www.ciao.co.uk>), and preprocessed into available form. The statistics of the two datasets are shown in Table 1.

Table 1 Statistics of the datasets

Dataset	Ciao	Epinions
# of Users	7375	22,166
# of Items	106,797	296,277
# of Ratings	283,434	917,619
# of Density (Ratings)	0.0360%	0.0140%
# of Trusts	111,781	355,813
# of Density (Trusts)	0.2055%	0.0724%

Ciao: In this dataset, each user has his profile, ratings, and trust relations. Each rating has the product name and category, the rating score, the time point when the rating is created. It also contains 28 classes. The rating score assigned by the user on the item is in $\{1,2,3,4,5\}$.

Epinions: This dataset contains all the same information as Ciao. But the numbers of users, items, and ratings are larger. It includes 27 classes. The rating score on the item is also in $\{1,2,3,4,5\}$.

We re-preprocess the two datasets to ensure that each user should have at least one neighbor item in his history buying list or at least one friend in the social connections. Each item should have at least one neighbor buyer in its history buyer list. Besides, we achieve high-order sampling to get the user–item–user subgraph, the item–user–item subgraph, and the social-item subgraph for our proposed model.

4.2 Evaluation

We adopted the root mean squared error (RMSE) and mean absolute error (MAE) to evaluate the performance of our proposed model. With the ground truth rating r_{ij} , the predicted rating r'_{ij} and the set of ratings O , the RMSE and the MAE can be formulated as:

$$RMSE = \sqrt{\frac{\sum_{i,j \in O} (r'_{ij} - r_{ij})^2}{|O|}} \quad (19)$$

$$MAE = \frac{\sum_{i,j \in O} |r'_{ij} - r_{ij}|}{|O|} \quad (20)$$

The smaller RMSE and MAE show better performance with higher accuracy. Just a slight improvement in the two metrics can affect the quality of the top-few recommendations.

Experiment settings. We conduct our experiments based on Pytorch v-1.5.0 in Ubuntu16.04 LTS with Intel(R) Xeon(R) E5-2680 v4 @ 2.40GHz CPU and a GTX2080ti GPU. In our experiment, we have two different splits for each dataset. One is to make 80% of the ratings as a training set, 10% as a valid set, and the last 10% a test set. The other is to make 60% of the ratings as a training set, 20% as a valid set, and the last 20% as a test set. We set the embedding size d as 64 for better performance. For the learning rate and the batch size, they are respectively set as that in $\{0.0001, 0.0005, 0.001, 0.0015, 0.002\}$ and that in $\{64, 128, 256, 512, 1024\}$. For the activation function, we choose the ReLU function. Besides, we adopted the early stopping strategy where the training process can stop if the metric on validation gets worse in the constant five epochs.

Neighbor sampling. The model training with mini-batch gradient descent is not critical since each user node or item node has many neighbors, which leads to the increasing computational cost. To address this problem, we adopt the neighbor sampling technique, which is pretty useful and commonly used in graph neural networks on large-scale data [39].

We sample m neighbors randomly for each center user and each center item in the item-view and the user-view aggregation to conduct this task. We sample p 1-order user neighbors randomly and then sample q 2-order user neighbors randomly for each 1-order node in the social-view aggregation. A subgraph that centers on u_i with $(p + pq + 1)$ user nodes can be obtained, which can reduce the computational cost to a certain degree and perform well on the datasets in the meantime. Here we set $m = 30$, $p = 10$, and $q = 5$, and the three values can be adjusted according to the model changing.

4.3 Comparison among methods: RQ1

Baselines: To demonstrate the performance of the proposed model HHRec, we compare it with the following methods:

SoRec [13]: This method performs co-factorization on the user–item rating matrix and the user–user social matrix

SocialMF [15]: It takes social information and propagates social data into the matrix factorization model for recommender systems.

TrustMF [18]: This method factorize trust networks to represented users as two low-dimensional vectors, according to the trust's directional property.

Table 2 Performance comparison of different recommender systems

Training	Metrics	Algorithms						
		SoRec	SocialMF	TrustMF	DeepSoR	GCMC+SN	GraphRec	HHRec
Ciao (60%)	MAE	0.8489	0.8353	0.7681	0.7813	0.7697	0.7540	0.7501
	RMSE	1.0738	1.0592	1.0543	1.0437	1.0221	1.0093	0.9884
Ciao (80%)	MAE	0.8410	0.8270	0.7690	0.7739	0.7526	0.7387	0.7273
	RMSE	1.0652	1.0501	1.0479	1.0316	0.9931	0.9794	0.9702
Epinions (60%)	MAE	0.9086	0.8965	0.8550	0.8520	0.8602	0.8441	0.8169
	RMSE	1.1563	1.1410	1.1505	1.1135	1.1004	1.0878	1.0662
Epinions (80%)	MAE	0.8961	0.8837	0.8410	0.8383	0.8590	0.8168	0.8006
	RMSE	1.1437	1.1328	1.1395	1.0972	1.0711	1.0631	1.0523

Table 3 Comparison among GCMC, GraphRec and HHRec

	Neighbor sampling	Higher-order semantics	Social network	Skip connection
GCMC	No	No	No	No
GraphRec	No	No	Yes	No
HHRec	Yes	Yes	Yes	Yes

DeepSoR [40]: This model adopts a deep neural network to learn representations of each user from social connections, which are used for probabilistic matrix factorization.

GCMC+SN [5]: This model is an advanced recommender system with graph neural network. To integrate social information into GCMC, Node2vec [41] is utilized to obtain user embedding as user social information.

GraphRec [6]: This model is a state-of-the-art social recommendation system tested on the Ciao dataset and Epinions dataset. It similarly considers the social connections between users and designed embedding layers for users, items, and ratings. We experiment with the setting of 60% training set and 80% training set, respectively, comparing all the above methods' performance. Table 2 shows the RMSE and MAE of rating prediction among the methods. In particular, the above baselines' experimental results have been achieved in [6]. We have some analysis as follows.

SoRec, SocialMF, and TrustMF are all based on matrix factorization with social information. Among these methods, SoRec captures social features by considering the user–user social matrix factorization and performs the worst. Unlike SoRec, SocialMF uses social regularization while TrustMF is combined with the propagation of trust information. Both of them make full use of social information in a specific way, which shows that trust information among users contributes to the social recommendation systems. Compared with matrix factorization-based methods, deep-based methods offer the powerful ability to learn the latent factor and give more accurate predictions. In deep-based algorithms, ones with graph neural networks, including GCMC-SN, GraphRec, and our proposed model HHRec, perform the best with considerable improvement. GCMC in the original paper doesn't consider the social connections but turns to be GCMC-SN with trust information using node2vec [41]. In this case, it performs more inferior than the other two methods based on graph neural networks. Among the baselines, GraphRec shows the most powerful performance on the two datasets, which indicates that graph neural networks are beneficial to the representation learning heterogeneous graph data. It can deal with both node information and structure information that all contribute to the performance.

Unlike GCMC and GraphRec, our proposed model considers multi-layer neighbors, sampling strategy, and high-order semantics, achieving the best performance among all the algorithms by modeling multi-view and high-order semantic information. Compared with GraphRec, our proposed HHRec executes multi-layer aggregation and high-order semantics processing, verifying the importance of high-order semantic information, including high-order social information and heterogeneous

Fig. 8 Comparison of different dimensions

Fig. 9 Comparison of different batch size

interactions. Besides, the improvement over GCMC-SN and GraphRec is more significant than that of DeepSoR over Trust-MF, which shows our proposed model's Effectiveness (Table 3).

In a word, the analysis can be summarized as:

- 1) trust information among users is critical to social recommendation;
- 2) graph neural network is a quite a lot powerful tool for graph data processing;
- 3) high-order semantic information and multi-layer aggregation is helpful for the user modeling and the item modeling.
- 4) our proposed method HHRc outperforms the representative baselines.

4.4 Hyperparameter sensitivity: RQ2

In this subsection, we try to study how our proposed model's hyperparameter affects its performance on the two datasets. Hyperparameters that we analyze include embedding size, batch size and learning rate. We take the setting of 80% training set, 10% valid set, and 10% test set in the following experiments.

Effect of embedding size. Our proposed model contains four embedding layers, including user embedding, item embedding, category embedding and opinion embedding, which can learn dynamically in the model training process. If the embedding size is too small, the embedding cannot contain enough feature information. Simultaneously, if it is too large, the embedding layers will become so redundant that it is hard to learn the useful feature information. In our proposed model, we keep the embedding size of the four embedded layers the same. We change the embedding size in {16, 32, 64, 128, 256} and study how it affects the performance of HHRc for recommendations. As is shown in Fig. 8, we found that with the embedding size 64, HHRc can get the most powerful performance on both two datasets. Generally, Its performance will get better and then become worse with the increasing embedding size. Thus, it is critical to choose a suitable embedding size according to the practical model to void a lack of information and redundancy and ensure the best performance.

Fig. 10 Comparison of different learning rate

Effect of batch size and learning rate. We need to set the proper batch size and learning rate in the training process, affecting the model performance greatly. The batch size changes in {64, 128, 256, 512} and the learning rate changes in {0.0001, 0.0005, 0.001, 0.0015, 0.002}. We conduct two groups of experiments. When exploring the batch size, for the specific batch size, we change the learning rate in {0.0005, 0.001, 0.0015} and choose the best result. When exploring the learning rate, we do the same with a change of the batch size in {128, 256}. Fig. 9 demonstrates that the batch size 128 and 156 have a similar effect on model training. But the too small or too larger batch size is not satisfying. When exploring the learning rate, in Fig. 10, when it's set as 0.0001, the learning process is too slow and hard to converge. $Lr = 0.001$ is the best choice, in which we got the most satisfying performance of our model except that $Lr = 0.0015$ set on the Epinions. When it's set larger like $Lr = 0.0015$ or $Lr = 0.002$, we can get smaller MAE but larger RMSE on the two datasets. The larger learning rate is a not bad choice if the metric MAE is more important than RMSE. Indeed, RMSE is greatly affected by abnormal users, and just a small number can lead to poorer RMSE. One feasible future work is to combine rating prediction in recommendation with anomaly detection to get smaller RMSE.

Fig. 11 Visualization of 256 users and 256 items in the test set

4.5 Visualization analysis: RQ3

This subsection aims to understand how the user's and item's latent representations h_i and h_j facilitate the predicted rating in the learning process. We randomly choose a batch of the test set on the Ciao dataset, including 256 users, 256 items, and their ratings, and use t-SNE to visualize the representations' visualization.

Figure 11 shows the visualization of the representations in a two-dimensional space. There are two observations as follows:

- 1) The connections between users and items in the test set are well reflected in a two-dimensional space. All users tend to a line distribution, and the items are divided into five parts.
- 2) Nearly most users will pay attention to the items in part 3 ~ part 5, which shows that these items may belong to mass products. For items in part 3 and part 4, nearly all the users will give them the highest rating, indicating that these items have relatively high quality and audience level. For items in part 5, most users give a rating below 4 to them, from which we can conclude that these items are of a little more inferior quality. For items in part 2, we can see that just a small number of users will buy them, and they are always unsatisfactory. As for items in part 1, we found users in part 1 like them while users in part 2 are not very satisfied with them, which implies that this kind of item only suits certain users.

From the above, HHRec can learn an effective user's latent representation h_i , which the information about the user's hobbies and behavior characteristics, and an effective item's latent representation h_j about its quality and category characteristics.

4.6 Variations of HHRec: RQ4

Exploring attention. HHRec adopts an attention mechanism to weigh the contribution of different neighbor nodes. In our model, we use three attention layers, which are user–item attention, item–user attention, and user–user attention. To analyze how the attention mechanism affects the model's ability, we design two variations HHRec- α and HHRec- β by setting different attentions for the original model.

HHRec- α considers user–item attention and item–user attention, respectively, in the item-view aggregation and the user-view aggregation. For the social-view aggregation, we take two-layer GCNs instead of two-layer GATs to shield the attention mechanism. HHRec- β considers the user–user attention in the social-view aggregation. We use the mean aggregation operator instead of the attention layer in the item-view aggregation and the user-view aggregation. In Fig. 12, we found HHRec- α outperforms HHRec- β in MAE and RMSE on the Ciao while HHRec- β outperforms HHRec- α in MAE and RMSE on the Epinions. With three attention layers in item-view aggregation, user-view aggregation, and social-view aggregation, HHRec transcends the two variations on the two datasets MAE and RMSE, proving the effectiveness of an attention mechanism for heterogeneous graph data in the social recommendation.

Effect of social information and category embedding. We consider the social-view aggregation for user modeling from multi-views. To test the impact of social information, we remove the social-view aggregation and use h_i^l as the last talent representation of the user u_i , called HHRec-SN. What's more, in the baselines, neither of the other two methods with graph

Fig. 12 Comparison of different attention settings

Fig. 13 Comparison of different variations

neural network is combined with category information of items. In general, the model can adaptively learn category feature information of an item hidden in the talent representation. However, in the recommendation's original data, each item has its category label to indicate its kind of commodity. We directly use the category label and add a category embedding layer for the model, making it easier to learn the item's talent category feature information. We remove the category embedding layer and design HHRec-CN to test how the category embedding affects our proposed model's performance. Fig. 13 shows a comparison among HHRec-SN, HHRec-CN and HHRec.

There are two key findings from the result: (1) The connectivities of users play a critical role in user modeling for social recommendations. Compared with HHRec, HHRec-SN has a farther more unsatisfactory performance, verifying social network information effectiveness. If not considering social data, a large amount of semantic information will lose, which leads to the worse implementation of the recommendations. (2) We now focus on the effectiveness of category embedding. We can find that the performance of HHRec-CN is a little worse than HHRec. For example, on average, MAE on Ciao dataset is 0.38% larger than that in HHRec while RMSE is 0.52% larger, and MAE on Epinions is 0.81% larger than that in HHRec while RMSE is 0.79% larger. It implies that category embedding makes it easier to learn category feature information, improve performance and verify the category's effectiveness.

To sum up, HHRec can capture rich high-order semantic information from multiple perspectives, including item-view aggregation, user-view aggregation, and social-view aggregation. It realizes multi-layer aggregations and skip connections to conduct rating prediction for recommendations with powerful performance.

5 Conclusion

This paper proposed a high-order hybrid semantic graph neural network for extending graph neural networks to process heterogeneous graphs for recommendations. HHRec mainly consists of two parts, which respectively realize high-order aggregations for a user and an item. We deal with the heterogeneous graph information network containing all the relevant information about the graph's topological structure. In terms of the aggregation methods, if we only aggregate the first-order neighbors of a specific user or an item, a lot of semantic information will lose. Therefore, information aggregation based on multi-order neighbors helps extract high-order semantic information from the network, achieving better rating predictions. We have tested our proposed architecture on two standard social recommendation data sets and compared the existing more advanced methods. The experimental results show that our proposed model HHRec has better performance than the current state-of-the-art methods. For our model, here we propose several improvements in the future:

- 1) Introduce the knowledge graph among the items and add an information network structure
- 2) The rating data is unevenly distributed. Enhancing the data or negative sampling can be considered.
- 3) Introduce the concept of a dynamic network to make the simulation of data more accurate.

In general, the proposed HHRec model exploits the advances in heterogeneous graph processing to present novel constructions of deep learning, which can handle high-order semantics data represented as the heterogeneous graph signals.

Acknowledgements This work was supported in part by the National Natural Science Foundation of China under Grant 61771322 and Grant 61871186 and in part by the Fundamental Research Foundation of Shenzhen under Grant JCYJ20190808160815125.

Data availability The datasets analysed during this study are included in the published article [38] (and its supplementary information files) and can be found in <https://www.cse.msu.edu/~tangjili/datasetcode/truststudy.htm>

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

References

- Ramlatchan A, Yang M, Liu Q, Li M, Wang J, Li Y. A survey of matrix completion methods for recommendation systems. *Big Data Mining Anal.* 2018;1(4):308–23.
- Xiaoxiao M, Wu J, Shan X, Jian Y, Sheng Quan Z, Hui X, editors. A comprehensive survey on graph anomaly detection with deep learning. 2021. arXiv preprint [arXiv:2106.07178](https://arxiv.org/abs/2106.07178).
- Su X, Shan X, Fanzhen L, Wu J, Jian Y, Chuan Z, Hu W, Cecile P, Surya N, Di J, et al. In: A comprehensive survey on community detection with deep learning. arXiv preprint [arXiv:2105.12584](https://arxiv.org/abs/2105.12584), 2021
- Liu F, Xue S, Wu J, Zhou C, Hu W, Paris C, Nepal S, Yang J, Yu PS. Deep learning for community detection: progress, challenges and opportunities. arXiv preprint [arXiv:2005.08225](https://arxiv.org/abs/2005.08225). 2020.
- van den Berg R, Kipf TN, Welling M. Graph convolutional matrix completion. arXiv preprint [arXiv:1706.02263](https://arxiv.org/abs/1706.02263). 2017.
- Fan W, Ma Y, Li Q, He Y, Zhao E, Tang J, Yin D. Graph neural networks for social recommendation. In: *The world wide web conference*; 2019. pp 417–426.
- Wu Q, Zhang H, Gao X, He P, Weng P, Gao H, Chen G. Dual graph attention networks for deep latent representation of multifaceted social effects in recommender systems. In: *The world wide web conference*; 2019. pp 2091–2102.
- Roshan B, Deepak G. Recommending top n movies using content-based filtering and collaborative filtering with hadoop and hive framework. In: *Recent developments in machine learning and data analytics*. Springer; 2019. p. 109–18.
- Chen R, Hua Q, Chang Y-S, Wang B, Zhang L, Kong X. A survey of collaborative filtering-based recommender systems: From traditional methods to hybrid methods based on social networks. *IEEE Access.* 2018;6:64301–20.
- Koren Y, Bell R, Volinsky C. Matrix factorization techniques for recommender systems. *Computer.* 2009;42(8):30–7.
- Yehuda K. 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. p. 426–34.
- Mnih A, Salakhutdinov RR. Probabilistic matrix factorization. In: *Advances in neural information processing systems*; 2008. pp 1257–1264.
- Hao M, Haixuan Y, Michael LR, Irwin K. Sorec: social recommendation using probabilistic matrix factorization. In: *Proceedings of the 17th ACM conference on information and knowledge management*; 2008. p. 931–40.
- Hao M, Dengyong Z, Chao L, Michael LR, Irwin K. Recommender systems with social regularization. In: *Proceedings of the fourth ACM international conference on Web search and data mining*; 2011. p. 287–96.
- Mohsen J, Martin E. A matrix factorization technique with trust propagation for recommendation in social networks. In: *Proceedings of the fourth ACM conference on Recommender systems*; 2010. p. 135–42.
- Liu BYYLD, Liu J. Social collaborative filtering by trust. In: *Proceedings of the twenty-third international joint conference on artificial intelligence, IEEE*; 2013. Citeseer.
- Guo G, Zhang J, Yorke-Smith N. Trustsvd: collaborative filtering with both the explicit and implicit influence of user trust and of item ratings. *AAAI.* 2015;15:123–5.
- Bo Yang Yu, Lei JL, Li W. Social collaborative filtering by trust. *IEEE Trans Pattern Anal Mach Intell.* 2016;39(8):1633–47.
- He X, Liao L, Zhang H, Nie L, Hu X, Chua T-S. Neural collaborative filtering. In: *Proceedings of the 26th international conference on world wide web*; 2017. pages 173–182.
- Monti F, Bronstein M, Bresson X. Geometric matrix completion with recurrent multi-graph neural networks. In: *Advances in neural information processing systems*; 2017. pp 3697–3707.
- Gao L, Wu J, Zhou C, Hu Y. Collaborative dynamic sparse topic regression with user profile evolution for item recommendation. In: *Proceedings of the AAAI conference on artificial intelligence*; 2017. volume 31.
- Li G, Wu J, Zhi Q, Chuan Z, Hong Y, Hu Y. Collaborative social group influence for event recommendation. In: *Proceedings of the 25th ACM international conference on information and knowledge management*; 2016. p. 1941–4.
- Gori M, Monfardini G, Scarselli F. A new model for learning in graph domains. In: *Proceedings 2005 IEEE international joint conference on neural networks, 2005. IEEE*; 2005. volume 2, pp 729–734.
- Micheli A. Neural network for graphs: a contextual constructive approach. *IEEE Trans Neural Netw.* 2009;20(3):498–511.
- Scarselli F, Gori M, Tsoi AC, Hagenbuchner M, Monfardini G. The graph neural network model. *IEEE Trans Neural Netw.* 2008;20(1):61–80.
- Bruna J, Zaremba W, Szlam A, LeCun Y. Spectral networks and locally connected networks on graphs. arXiv preprint [arXiv:1312.6203](https://arxiv.org/abs/1312.6203), 2013.
- Kipf TN, Welling M. Semi-supervised classification with graph convolutional networks. arXiv preprint [arXiv:1609.02907](https://arxiv.org/abs/1609.02907), 2016.
- Hamilton W, Ying Z, Leskovec J. Inductive representation learning on large graphs. In: *Advances in neural information processing systems*; 2017. pp 1024–1034.
- Veličković P, Cucurull G, Casanova A, Romero A, Lio P, Bengio Y. Graph attention networks. arXiv preprint [arXiv:1710.10903](https://arxiv.org/abs/1710.10903), 2017.

30. Guohao L, Matthias M, Ali T, Bernard G. Deepgcns: can gcns go as deep as cnns? In: Proceedings of the IEEE International Conference on Computer Vision; 2019. p. 9267–76.
31. Yu R, Wenbing H, Xu T, Junzhou H. Droppedge: towards deep graph convolutional networks on node classification. In: International conference on learning representations; 2019.
32. Chen M, Wei Z, Huang Z, Ding B, Li Y. Simple and deep graph convolutional networks. arXiv preprint [arXiv:2007.02133](https://arxiv.org/abs/2007.02133), 2020.
33. Mikolov T, Chen K, Corrado G, Dean J. Efficient estimation of word representations in vector space. arXiv preprint [arXiv:1301.3781](https://arxiv.org/abs/1301.3781), 2013.
34. Bahdanau D, Cho K, Bengio Y. Neural machine translation by jointly learning to align and translate. arXiv preprint [arXiv:1409.0473](https://arxiv.org/abs/1409.0473), 2014.
35. Fangyu Z, Li S, Jie Z, Zhang W, Liu W. A sufficient condition for convergences of adam and rmsprop. In: Proceedings of the IEEE conference on computer vision and pattern recognition; 2019. p. 11127–35.
36. Kaiming H, Xiangyu Z, Shaoqing R, Jian S. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In: Proceedings of the IEEE international conference on computer vision; 2015. p. 1026–34.
37. Srivastava N, Hinton G, Krizhevsky A, Sutskever I, Salakhutdinov R. Dropout: a simple way to prevent neural networks from overfitting. *J Mach Learn Res.* 2014;15(1):1929–58.
38. Tang J, Gao H, Liu H. mtrust: discerning multi-faceted trust in a connected world. In: Proceedings of the fifth ACM international conference on Web search and data mining; 2012. pp. 93–102.
39. Jiezhong Q, Jian T, Hao M, Yuxiao D, Kuansan W, Jie T. Deepinf: social influence prediction with deep learning. In: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining; 2018. p. 2110–9.
40. Wenqi F, Qing L, Min C. Deep modeling of social relations for recommendation. Thirty-second AAAI conference on artificial intelligence (AAAI-18). AAAI press; 2018. p. 8075–6.
41. Aditya G, Jure L. node2vec: Scalable feature learning for networks. In: Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining; 2016. p. 855–64.

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.