# Heterogeneous Hypergraph Neural Network for Social Recommendation Using Attention Network

BILAL KHAN, JIA WU, JIAN YANG, and XIAOXIAO MA, School of Computing, Macquarie University, Australia

Graph neural networks (GNNs) have been used extensively as a backbone for social recommendation. However, their application to a diverse range of situations is still rather limited. This is because graph structures only leverage pairwise user relationships. They cannot capture the higher-order relationships so common in the real world, and ignoring the interest friends and strangers might have in similar items is severely hampering the expressiveness of the current graph-based recommendation models. Hence, in this article, we outline a heterogeneous hypergraph neural network for social recommendation, called Heterogeneous Hypergraph neural network for Social Recommendation using an Attention Network (HHGSA), that incorporates an attention network to address these issues. The hypergraph is able to represent higher-order relationships through five motifs: friend and stranger item appeal, item similarity, user similarity based on interactions with items, and social relations. Two modules, the attentive vertex aggregation module and the attentive hyperedge aggregation module, capture user and item attention. In addition, it has been discovered that similar items have identical appeal when displayed to users. A GNN aggregates the user embedding data, including information about the friend and stranger and item embeddings. Finally, information about users and items is aggregated for social recommendations. Extensive experiments on four datasets demonstrate that the HHGSA model outperforms a wide range of baselines and can significantly improve the accuracy of recommendations.

CCS Concepts: • **Information systems** → **Collaborative filtering;**

Additional Key Words and Phrases: Graph, graph neural networks, node embedding, hypergraph, hyperedge, attention network, recommender system

**ACM Reference format:**
Bilal Khan, Jia Wu, Jian Yang, and Xiaoxiao Ma. 2025. Heterogeneous Hypergraph Neural Network for Social Recommendation Using Attention Network. *ACM Trans. Recomm. Syst.* 3, 3, Article 30 (March 2025), 22 pages.
https://doi.org/10.1145/3613964

## 1 INTRODUCTION

With the exponential growth of data over the past several decades, information overload has become a severe problem for internet consumers. Our best solution to date has been recommender systems, which filter information and suggest information of likely interest to online users. A good recommender system will both filter data and deliver accurate and trustworthy information to its users, assisting them to discover vital information hidden inside the ocean of data available on

This work was supported by the Australian Research Council (ARC) Projects DP230100899, DE200100964 and LP210301259.
Authors' address: B. Khan, J. Wu, J. Yang, and X. Ma, School of Computing, Macquarie University, Sydney, NSW, Australia, 2109; emails: {bilal.khan, jia.wu, jian.yang}@mq.edu.au, xiaoxiao.ma2@students.mq.edu.au.

2770-6699/2025/03-ART30 $15.00
https://doi.org/10.1145/3613964

the internet. Recommendation systems are used widely in the business sector. For example, most people will be familiar with the product recommendation systems on Amazon and eBay, the movie recommender systems on Netflix, and the music recommenders on iTunes.

Since the early 1990s, recommender systems have been an active field of study, gaining substantial attention across a diverse range of disciplines [17, 18, 57], including mathematics, psychology, physics, and computer science. There are many different types of recommender systems. For example, there are **collaborative filtering– (CF)** based approaches [2], content-based approaches [41], hybrid methods [34], and systems built on latent factors [9].

The content-based approaches offer similar items to users based on their past preferences [41]. The collaborative filtering approaches predict user preferences by identifying complicated patterns from the historical activities of users. They then recommend items liked by other users with comparable preferences. Initially, users and items of relevance to the target query are differentiated by analyzing past interactions. Then, depending on the analysis result, the model may recommend an item. However, there are two main approaches to CF: memory based and model based [36]. The memory-based methods can be either user or item based. The user-based techniques provide recommendations based on user–user correlations that measure the closeness of the two users' interests in a specific item. Similarly, the item-based techniques yield recommendations based on item–item correlations. The model-based algorithms use machine learning techniques to build a model that can make recommendations based on unknown ratings [40]. It shows the latent factors that describe the observed ratings. Some of the more well-known model-based techniques include the Bayesian belief net CF model [33], Random Walk [25], Neural CF [22], and models based on matrix factorization [5]. In fact, matrix factorization has been extensively used to implement recommender systems. With matrix factorization, the items and users exist in a shared space [5, 38].

In addition, with the rise in popularity of social networking platforms like Facebook, Twitter, Epinions, and so on, social relations are now being incorporated into the preferences that inform recommender systems. Social relations provide user interaction information so as to more accurately model user preferences. This is often coupled with other perspectives that expose users to items [16]. Supported by homophily [32] and social influence [8], these social connections can substantially influence user preferences. Homophily refers to the notion that users with similar interests are more likely to be linked, while social influence demonstrates that related users tend to make comparable decisions [48]. So because users are strongly influenced by the items their friends buy and share, these rich data have vast potential to enhance recommendation performance [28], and recommendations based on social relations are highly likely to become future interests [30].

Incorporating social information into recommender systems has been demonstrated to result in substantial performance improvements in recommender systems under ideal conditions [13, 14, 44, 46]. Additionally, this approach can alleviate the problem of data sparsity [19]. For example, suppose a user has limited involvement with an item. In such cases, a social recommender system will look to the user's friends for engagement with the items and will generate an improved recommendation by determining whether the user's friends like the product. Numerous social recommendation models have been proposed based on this paradigm [52] with many outperforming traditional recommendation models.

With the emergence of deep learning technologies, the **graph neural networks (GNNs)** are becoming more and more common as the backbone of artificial intelligence frameworks. Among other functions, GNNs are being used to aggregate feature data and learn data representations [10]. Once aggregated, node information is propagated across the graph. Thus, GNNs capture information about both nodes and topological structures, not only making them a highly effective tool for learning representations [13, 20] but also offering a concept that fundamentally conforms with

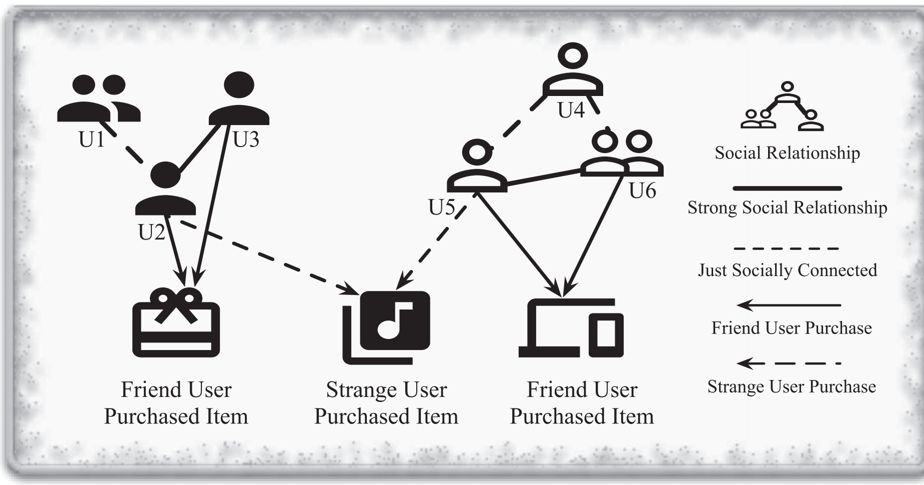


Fig. 1. A user–item graph representing strong, weak, and stranger relationships.

social recommendation. In fact, several recent social recommendation models built on GNNs have shown remarkable performance [29, 39, 45]. For example, with GraphRec [13], Fan et al. proposed the first social recommendation technique based on a GNN, where graphs of social relations and users–items are used to learn item and user embeddings. MGNN [49] aggregates social relations and user–item graphs to create mutual social (user–user) embedding layers, while Wu et al. [47] devised a framework to learn both user-specific and dynamic attention weight representations using a GNN. However, traditional GNN-based models often involve many propagation steps for each paired interaction to transmit their message, which is both computationally costly and likely to introduce excessive noise. Furthermore, a significant limitation of the various GNN-based social recommendation algorithms is their reliance on pairwise user interactions while ignoring the higher-order, more complex relations. For instance, if two friends purchase the same item, then their relationship can be thought of as being more substantial than if they were merely just socially connected. Additionally, details of purchasing the same product even when the user is a stranger is often neglected. Figure 1 demonstrates that $(U_2, U_3)$, and $(U_5, U_6)$ have strong relationships, shown by a solid line, since they are socially connected and also buy the same products, whereas $(U_1, U_2)$, $(U_4, U_5)$, and $(U_5, U_6)$ are just socially connected shown by the dotted line. In contrast, the dashed line shows the $(U_2, U_5)$, strangers who have no social relation and buy but have purchased the same item.

Although social recommendation approaches have been studied for a long time, they still have several serious shortcomings. For example,

- *Inadequate modeling of high-order correlations*: Higher-order connections between users and items are essential for realistic modelling. However, the current graph structure methods can only depict a pairwise relationships and their ability to represent higher-order correlations is limited [26].
- *Item–item similarity, friend, and stranger information is hard to extract*: Social recommendation systems manage both user–item bipartite graphs and social network graphs. Integrating information effectively from both graphs is still an active area of research. In addition, existing research techniques do not consider item–item similarities and stranger interactions with items.

In this research, we propose a novel framework based on heterogeneous hypergraphs to overcome these shortcomings. Our framework can represent higher-order complex relations, such as user–user relations, all users purchased an item, all items purchased by a user, strangers purchased an item, friend users bought an item, and so on. A hypergraph [43] is a non-pairwise data structure that can represent high-order complex relations, such as items purchased by strangers or the diversity of social impacts, and so on. In other words, it allows hyperedges, which connect more than two nodes. A few researchers have been motivated to explore how GNNs and hypergraphs might be integrated to help improve representation learning for social recommendation but other than this, hypergraphs have not received much attention in this field. This is unfortunate, because hypergraphs have significant benefits to user modeling over traditional graphs. With this research article, our aim is to build a heterogeneous hypergraph-based framework for social recommendation. Called **Heterogeneous Hypergraph neural network for Social Recommendation using an Attention Network (HHGSA)**, the framework incorporates an attention mechanism and other novel techniques to overcome some of the limitations of the traditional GNNs.

Using five distinct motifs, HHGSA constructs a heterogeneous hypergraph to learn higher-order relations, such as friend and stranger attraction to items. To improve representation learning for social recommendation, the GNN is combined with a hypergraph. Two modules, the attentive vertex aggregation module and the attentive hyperedge aggregation module capture the varying levels of influence among users. Additionally, HHGSA computes user–user and item–item similarities. Further, since we know that similar items will always have comparable attraction when displayed to users, a convolutional graph network aggregates information on the user embeddings (both friend and stranger) and the item embeddings. Finally, the user and item aggregations are combined to generate recommendations. In summary, this research article makes the following contributions to the literature:

- This article outlines a novel framework for social recommendation, called HHGSA, that is based on a GNN and heterogeneous hypergraph modeling. The framework captures high-order complex relations, such as user–item relations, item–user relations, and interaction with similar items by both friends and strangers.
- HHGSA maintains the specific properties and representations of the users and items while integrating multiple types of hypergraphs, including social relations, items purchased by a user, the set of users who bought an item, friends who purchased the same item(s), and strangers who purchased the same item(s).
- An extensive set of experiments with social recommendation tasks were conducted on four benchmark datasets. The results demonstrate that HHGSA provided higher-quality recommendations and more precise results than the most recent models in this field.

The rest of this article is structured as follows. The problem formulation is covered in Section 2. In Section 3, hypergraph convolution is explained. Section 4 introduces the proposed framework and methodology. The experimental findings are presented in Section 5. In Section 6, the literature on social recommendation is reviewed. The article then concludes with a summary of the material covered, the limitations of this study, and our intentions for future work.

## 2 PROBLEM FORMULATION

### 2.1 Notation Definition

Let $U = \{u_1, u_2, u_3 \ldots, u_n\}$ and $\mathcal{T} = \{t_1, t_2, t_3, \ldots t_n\}$ represent the set of $n$ users and $m$ items, respectively. We assume the purchased matrix $P_{ij} \in \mathbb{R}^{n \times m}$. This is also known as the user–item graph. $P = 1$ if the user $u_i$ purchased an item $t_j$ and 0 otherwise. Similarly, the social relations matrix of users $U \in \mathbb{R}^{nxn}$ is also known as a user–user symmetric graph or social graph. $U_{ij} = 1$

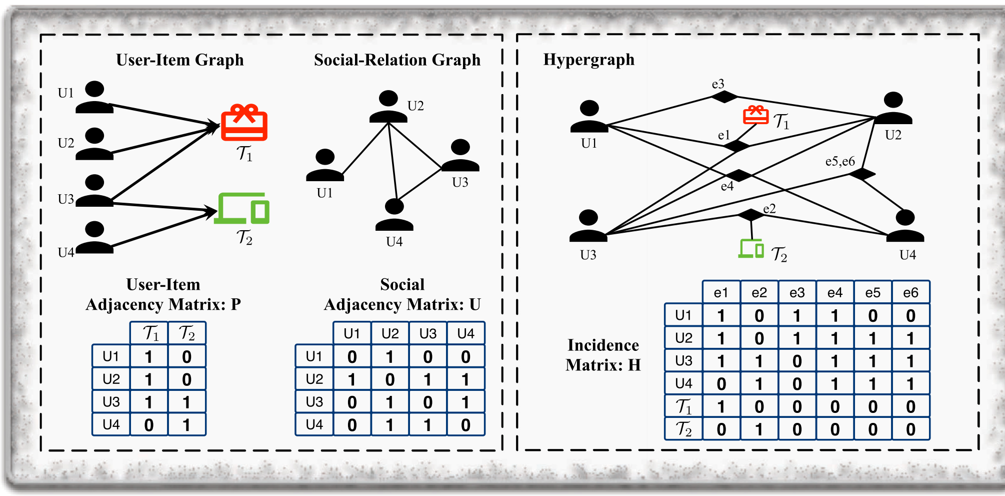


Fig. 2. A comparison between graph and hypergraph structures. An example of a graph and a hypergraph. (a) A bipartite graph (left) and social relation graph (right). The bipartite graph displays the relationship between users and items. Likewise, the social relation graph indicates user–user links. Both graphs are represented by the corresponding adjacency matrix. (b) A hypergraph of the same data. The hypergraph shows six hyperedges ($e_1$–$e_6$), four users ($U_1$–$U_4$), and two objects ($\mathcal{T}_1$, $\mathcal{T}_2$). The hyperedge $e_1$ indicates that $U_1$, $U_2$, and $U_3$ acquire item $\mathcal{T}_1$. The hyperedge $e_2$ indicates that users $U_3$ and $U_4$ bought item $\mathcal{T}_2$. The hyperedge $e_3$ demonstrates that user $U_1$ is $U_2$'s buddy. The hyperedge $e_4$ demonstrates $U_2$'s friendship with $U_1$, $U_3$, and $U_4$. The hyperedge $e_5$ indicates that user $U_3$ is a friend of user $U_2$, and $U_4$, while the hyperedges $e_6$ indicates that user $U_4$ is a friend of both $U_2$ and $U_3$.

if user $u_i$ is connected with user $u_j$ and 0 otherwise. This research aims to predict the unknown interactions between the user and the item with the appropriate encoded representation. In the neural network, $x_i \in \mathbb{R}^d$ and $y_j \in \mathbb{R}^d$ denote the embedding vector representation of $u_i$ user and $t_j$ item, respectively. The length of the embedding vector is $d$.

## 2.2 Hypergraph Definition

A hypergraph is denoted as $H(V, E)$, where $V = \{v_1, v_2, \ldots v_N\}$ and $E = \{e_1, e_2, \ldots e_M\}$. Here $V$ represents the set of nodes in the hypergraph, and $E$ represents the set of edges also known as hyperedges. The hyperedges are nonempty subsets of the nodes, which may connect several nodes, unlike the graph. The incidence matrix $H \in \mathbb{R}^{NXM}$ represents the hypergraph, where $N$ and $M$ denote the set of nodes and hyperedges, respectively. $H_i = 1$ if a node exists in hyperedge and 0 otherwise.

Figure 2 depicts the graph structure and hypergraph used to represent the social recommendation network. In this example Figure 2(a), there are two basic graphs, one for the social relationships and the other for the users–items. The bipartite user–item graph indicates that user $u_i$ has purchased item $t_i$. The social relationship graph illustrates the social connections between users. The adjacency matrices $P$ and $U$ depict the basic graphs, the user–item and social relationship graphs, respectively. $P_{ij} = 1$ if $u_i$ purchased the item $t_j$ and 0 otherwise. Similarly, $U_{ij} = 1$ if $u_i$ and $u_j$ are socially linked and 0 otherwise.

Figure 2(b) shows the incidence matrix $H$, which depicts the hypergraph of the links between user–user and user–item. If node $v_i$ exists in hyperedge $e_j$, then the input of the incidence matrix is 1 and 0 otherwise. For simplicity, Figure 2(b) uses just two kinds of hyperedges. The hyperedges

$e_1$ and $e_2$ represent the items $\mathcal{T}_1$ and $\mathcal{T}_2$, respectively, which were purchased by a set of users. The hyperedges $e_3$, $e_4$, $e_5$, and $e_6$ illustrate the users' $u_1$, $u_2$, $u_3$, and $u_4$ respectively social connections with friends. Because the hyperedge connects several nodes in the hypergraph, which improves the higher-order relations. However, the simple graph cannot depict the higher-order complex relations between users and items.

## 3 HYPERGRAPH CONVOLUTION

In a hypergraph, a convolutional network operator estimates the probability of a transition between vertices so that the embedding of each vertex may be transmitted across a GNN. This hypergraph convolutional network operator [1] is defined as follows:

$$(\mathbf{X})^{(l+1)} = \sigma\left(\mathbf{D}^{\frac{-1}{2}}\mathbf{H}\mathbf{L}^{-1}\mathbf{H}^T\mathbf{D}^{\frac{-1}{2}}\mathbf{X}^{(l)}\mathbf{P}\right). \tag{1}$$

In Equation (1), $\sigma(.)$ is a nonlinear activation function, $H$ is the hypergraph incidence matrix, and $(X)^l$ is the embedding of the vertex feature of the $l$th layer. $\mathbf{P} \in \mathbb{R}^{F^{(l)} \times F^{(l+1)}}$, where $P$ trainable parameters are considered as a matrix (weight) between the $l$th and the $l+1$ layer. $\mathbf{X}^{(l)} \in \mathbb{R}^{N \times F^{(l)}}$ is the input vertex feature at the $l$th layer, and $\mathbf{X}^{(l+1)} \in \mathbb{R}^{NxF^{(l+1)}}$ is the input vertex feature at the $(l+1)$th layer. $(\mathbf{X})^{(l+1)}$ is the output of the $l$th layer, $D$ is the vertex degree matrix, and $L$ represents the hyperedge degree matrix in the hypergraph. Both $D$ and $L$ are diagonal matrices.

$H$ defines the message passing path from the hyperedge to the vertices. In the incidence matrix, the hyperedges are considered to be columns, and the vertices are rows. $H^T$ specifies the message passing path from the vertices to the hyperedges where the vertices are the columns and the hyperedges are the rows.

First, the vertex features are collected through $H^T$ to construct the hyperedge feature. Then the enhanced vertex features are obtained through $H$ by aggregating any associated hyperedge features. In the last step, the model $P$ is trained with the nonlinear activation function $\sigma(\cdot)$.

Hence, the outlined hypergraph convolutional network operator can capture high-order correlations.

## 4 METHODOLOGY

This section presents the detail of our framework HHGSA. A description of how the heterogeneous hypergraph is constructed is presented in Section 4.1. Section 4.2 explains the relationship between the attention mechanism and the hyperedge and outlines the two attentive hyperedge aggregation and attentive vertex aggregation modules. Then Section 4.3 presents the user and item aggregation processes, which merge the embeddings associated with the friends and strangers into one user aggregation. The item and user aggregations are then concatenated. Section 4.4 describes the loss function and how the is model optimized.

### 4.1 Construction of Hypergraph

The first step in constructing the hypergraphs is to unify the user–item bipartite graph and the social network. This is done to ensure that the higher-order relationships between users and items are depicted accurately. Then the heterogeneous hypergraphs are constructed. A hypergraph includes item nodes, user nodes, and hyperedges. The hyperedges represent the higher-order relations between different types of nodes. Five motifs guide the hypergraph creation framework to define the appropriate structures for the user–item and user–user networks. Figure 3 shows the motifs in the hypergraph.

The first form of motif is the "Set of items purchased by a user," known as the user motif. This identifies strong user connections and assists in locating user interest communities. The second is

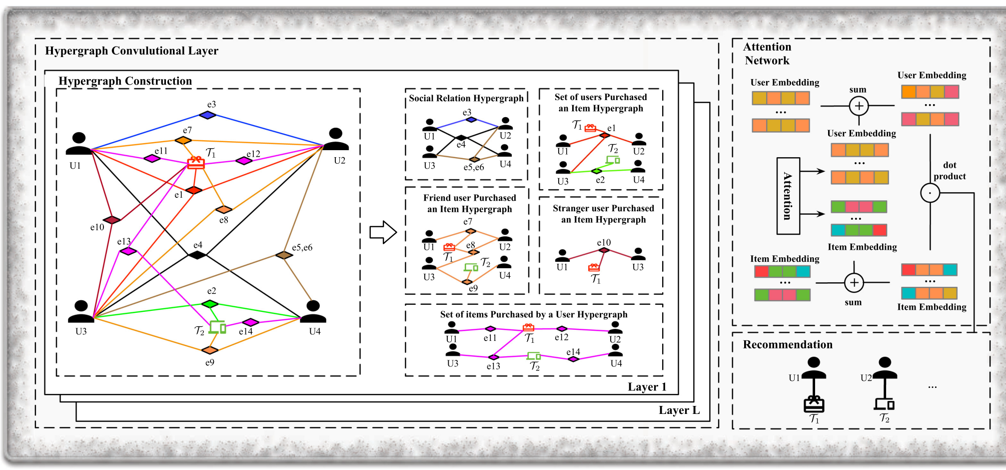


Fig. 3. The HHGSA framework.

the "Set of users who purchased an item," known as the item motif. This type of motif identifies item–item similarities based on user purchases. The third and fourth motifs are "Friends who bought an item" (the friend motif) and "Strangers who purchased an item" (the stranger motif), respectively. The last category is "social relations" or the social motif. This motif identifies the user's friend list.

Prior studies typically define $2^{M+N} - M - N - 1$ hyperedges. However, we define $2M + N + f + s$ hyperedges, where $N$ and $M$ represent the number of users and items, respectively. This considerably reduces the time complexity of our model compared to existing hypergraph frameworks. Here $f$ and $s$ represent the friend and stranger groups, respectively.

Moreover, it helps to identify the best user–user and item–item samples for the attention mechanism based on the definitions of five hyperedge motifs before attempting to predict future behaviors.

In this step, the weight of the $i$th hyperedge is calculated. Equation (2) computes the weight of the $i$th hyperedge based on its degree. The degree of a hyperedge is the number of nodes included in the hyperedge,

$$\mathbf{w}_i = \frac{deg(e_i) - min(deg(e))}{max(deg(e)) - min(deg(e))}, \tag{2}$$

where $deg(e_i)$ defines the degree of $i$th hyperedge and $min(deg(e))$ and $max(deg(e))$ are the respective minimum and maximum degrees of all the hyperedges.

The user and item representations are learned through a divide-and-conquer strategy. A hypergraph built from the five motif categories can be partitioned into the same number of hypergraphs to illustrate higher-order relations, as shown in Figure 3. These hypergraphs are represented by incidence matrices $H_{Users}$, $H_{Items}$, $H_{Friends}$, $H_{Str}$, and $H_{Social}$. The hypergraph incidence matrix $H_{Users}$ contains information related to the "set of items purchased by a user." In this hypergraph, the user is the hyperedge, and all the items are nodes. The hypergraphs incidence matrices $H_{Items}$ include data related to the "Set of users who purchased an item." This hypergraph represents items as hyperedges and users as nodes. The hypergraph incidence matrix $H_{Friend}$ contains information related to the "Friends who bought an item." In this hypergraph, the hyperedges represent the items, while the nodes indicate socially linked users. The hypergraph incidence matrix $H_{str}$ contains information related to the "Users that did not have a social connection yet purchased the same item," while the incidence matrix of hypergraphs $H_{social}$ provides information on "socially linked users," which gives a list of common acquaintances.

Inspired by recent developments in spectral clustering [4], a **multilayer perceptron (MLP)** with a softmax calculates the hyperedge assignments for the users and items. Each user $e_{uk}$ or item $e_{ik}$ representation corresponds to the $k$th row of its hypergraph-incidence matrix $H$,

$$\mathbf{H}_{\mathbf{user}} = Softmax(ReLU(E_{user}W_{user,1})W_{user,2}), \tag{3}$$

$$\mathbf{H}_{\mathbf{item}} = Softmax(ReLU(E_{item}W_{item,1})W_{item,2}), \tag{4}$$

where $W$ is a trainable weight matrix that learns $K$ hyperedges. The $Softmax$ function allocates a node to many hyperedges probabilistically. All five types of hypergraphs use the same dynamic hypergraph learning. Once all the hypergraphs have been learned, a spectral hypergraph convolution process follows, as outlined in Equation (1). The modified equations Equations (5) to (9) correspond to each particular hypergraph, defining its convolution:

$$(\mathbf{X}_{\mathbf{user}})^{(l+1)} = \mathbf{D}_{\mathbf{user}}{}^{-1}\mathbf{H}_{\mathbf{user}}\mathbf{L}_{\mathbf{user}}{}^{-1}\mathbf{H}_{\mathbf{user}}{}^{T}\mathbf{X}_{\mathbf{user}}{}^{(l)}, \tag{5}$$

$$(\mathbf{X}_{\mathbf{item}})^{(l+1)} = \mathbf{D}_{\mathbf{item}}{}^{-1}\mathbf{H}_{\mathbf{item}}\mathbf{L}_{\mathbf{item}}{}^{-1}\mathbf{H}_{\mathbf{item}}{}^{T}\mathbf{X}_{\mathbf{item}}{}^{(l)}, \tag{6}$$

$$(\mathbf{X}_{\mathbf{friend}})^{(l+1)} = \mathbf{D}_{\mathbf{friend}}{}^{-1}\mathbf{H}_{\mathbf{friend}}\mathbf{L}_{\mathbf{friend}}{}^{-1}\mathbf{H}_{\mathbf{friend}}{}^{T}\mathbf{X}_{\mathbf{friend}}{}^{(l)}, \tag{7}$$

$$(\mathbf{X}_{\mathbf{str}})^{(l+1)} = \mathbf{D}_{\mathbf{str}}{}^{-1}\mathbf{H}_{\mathbf{str}}\mathbf{L}_{\mathbf{str}}{}^{-1}\mathbf{H}_{\mathbf{str}}{}^{T}\mathbf{X}_{\mathbf{str}}{}^{(l)}, \tag{8}$$

$$(\mathbf{X}_{\mathbf{social}})^{(l+1)} = \mathbf{D}_{\mathbf{social}}{}^{-1}\mathbf{H}_{\mathbf{social}}\mathbf{L}_{\mathbf{social}}{}^{-1}\mathbf{H}_{\mathbf{social}}{}^{T}\mathbf{X}_{\mathbf{social}}{}^{(l)}, \tag{9}$$

Equations (5) to (9) produce specific hypergraph embeddings. To learn the user representations that encode the higher-order information in each specific hypergraph $H_{user}$ in Equation (5).

The matrix multiplication $H_{user}{}^{T}\mathbf{X}_{\mathbf{user}}{}^{(l)}$ then propagates the messages from the nodes to the hyperedges, followed by a premultiplication step on $\mathbf{X}_{\mathbf{user}}$ to aggregate the hyperedge information and update the nodes.

Multiple user representations $\{X^{(0)}_{user}, X^{(1)}_{user} \dots X^{(l)}_{user}\}$ and item representations $\{X^{(0)}_{item}, X^{(1)}_{item}, \dots X^{(l)}_{item}\}$ are generated from the $l$th layers of the hypergraph convolution. Equations (10) and (11) formulate the final representations by concatenating the representations from each layer,

$$\mathbf{X}_{\mathbf{user}} = \left\{X^{(0)}_{user}||X^{(1)}_{user}|| \cdots ||X^{(l)}_{user}\right\}, \tag{10}$$

$$\mathbf{X}_{\mathbf{item}} = \left\{X^{(0)}_{item}||X^{(1)}_{item}|| \cdots ||X^{(l)}_{item}\right\}, \tag{11}$$

where || represents a concatenation operation. The friend, stranger, and social hypergraph convolutions follow the same procedure. The final representations are then combined by concatenating the representations from each layer. Friends, strangers, and social relations are the final representations, indicated by $X_{freind}$, $X_{str}$, and $X_{social}$, respectively. The user features $X_u$ are created by combining the $X_{user}$ and $X_{social}$ representation results. Similarly, the item aggregator $X_i$ combines the results of the $X_{item}$, $X_{str}$, and $X_{friend}$ to provide latent item features.

In this section, we learned how hypergraph convolution helps structure the hypergraph and captures higher-order information, such as item–item similarity, friend, and stranger and the various ways users or items clustered together along a hyperedge.

### 4.2 Hypergraph Attention Relation

The HHGSA framework incorporates the hypergraph attention network HGAT [7]. HGAT comprises two primary components: an **attentive hyperedge aggregation (AHA)** module and an **attentive vertex aggregation (AVA)** module. AHA aggregates information about the hyperedges, whereas AVA aggregates information about the nodes. The attention network receives the node embedding matrix and the hyperedge embedding matrix of all five sub-hypergraphs as its input.

AVA generates the hyperedge embeddings by aggregating the information on connected nodes. The trainable weight matrix $W$ calculated in Equation (2) transforms the nodes and hyperedge features. Then the coefficient matrix between the node $i$ $(v_i)$ and the hyperedge $j$ $(e_j)$ can be computed using Equation (12),

$$\mathbf{coe_{ij}} = \frac{exp(W^T(v_i, e_j))}{\sum_{e_k \in E_i} exp(W^T(v_i, e_k))}, \tag{12}$$

where $v_i$ and $e_j$ represent the concatenated feature vectors of the node $i$ and the hyperedge $j$. $E_i$ is a set of hyperedges connected to a node $i$, $W$ is a linear transformation weight matrix, and $e_j$ is the feature vector of hyperedge $j$. A weight vector $W$, the concatenated feature vectors of the node $v_i$, and the hyperedge $e_j$ are used to calculate $coe_{ij}$. Through AVA, the node feature vector $v_i = \sum_{j \in E_i} coe_{ij} e_j$ is the weighted sum of the feature vectors of the hyperedges incident to node $i$.

AHA combines the information from different hyperedges by assigning an attention weight to each hyperedge. The coefficient matrix generated in Equation (12) by AVA is identical to that computed for AHA, except it applies to $N_j$, which is the set of nodes incident to the hyperedge $j$. The hyperedge feature vector for the hyperedge $e_j = \sum_{i \in N_j} coe_{ij} x_i$, where $x_i$ is the feature vector of node $i$.

The hypergraph $H_{item}$, which contains information about the "set of users who purchased an item," is used to learn a precise item embedding, with the node $v_i$ representing users and the hyperedge $e_j$ representing items. The hypergraph $H_{user}$, which contains information about the "Set of items purchased by a user," is used to learn the embedding vector for the users. Information about the node $v_i$ for the next layer of users is learned from the hyperedge $e_j$ information. The attention coefficient matrix COE in Equation (12) is obtained through a nonlinear ReLU and a softmax function, where the $coe_{ij} \in [0, 1]$. The coefficient matrix is calculated by Equation (13),

$$\mathbf{COE} = H \cdot softmax(ReLU(((VW)(EW)^T))), \tag{13}$$

where $H$ is the hyperedge incidence matrix (node-hyperedge relation matrix), $V \in R^{N \times d}$ is the node feature matrix, and $E \in R^{M \times d}$ is the hyperedge feature matrix. $N$ and $M$ represent the number of nodes and hyperedges, respectively. In the incidence matrix $H_{user}$, conversely, the items are nodes and the users are hyperedges. Finally, a weighted sum of the connected nodes is computed in Equation (14) using the coefficient matrix COE defined in Equation (13),

$$\mathbf{E_{features}} = \sigma[(COE)^T V], \tag{14}$$

where $\sigma$ is the activation function. Thus, the hyperedge features $E_features$ can be computed using Equation (14). For example, in the hypergraphs $H_{user}$ and $H_{item}$ , the features are the users $E_u$ and items $E_i$, respectively. The same process is used to compute the node features for the AHA module. The hyperedge-vertex attention coefficient matrix is computed with Equation (15),

$$\mathbf{COE} = H \cdot softmax(ReLU(((EW)(VW)^T))). \tag{15}$$

The COE may also be used with Equation (16) to produce the node features,

$$\mathbf{E_{features}} = \sigma[(COE)^T E]. \tag{16}$$

HGAT generates hyperedge features by aggregating the features of connected nodes. These nodes are also updated by considering the hyperedge features. This node-hyperedge-node approach of transforming the graph efficiently defines higher-order relationships in the data.

The same method is used to obtain the node and hyperedge features for all five types of split hypergraphs (as discussed in the Section 4.1).

The primary goal of the HGAT is to formulate an incidence matrix that reveals the relationships between hyperedges. While people do pay attention to a certain number of social relations, most do not have endless social links. From a social recommendation standpoint, a fundamental analogue of attention is where two users purchase similar products. If this list is relatively long, then there is a high level of attention between them; otherwise, there is low attention. Equation (17) calculates the attention score between the hyperedges $e_i$ and $e_j$,

$$\mathbf{Atten_{e_i,e_j}} = \left[\frac{sim(e_i, e_j)}{|e_i|}\right], \tag{17}$$

where $sim(\cdot)$ computes the similarities between the hyperedges $e_i$ and $e_j$ based on common nodes and $|e_i|$ represents the number of vertices in hyperedge. Attention relation focuses on a subset of users, which also improves the efficiency of the suggested framework.

The attention will equal 0 if the two hyperedges share no common node. In addition, $Atten_{e_i,e_j}$ and $Atten_{e_j,e_i}$ do not receive the same attention. For instance, say user $u_1$ is interested in the list of products that user $u_2$ also purchased, but user $u_2$ need not share this interest. Thus, "attention" identifies the user interest group and estimates the item–item and user–user similarities. More specifically, it calculates the user–user, item–item, user–item, and friend and stranger attention. The attention coefficient of the node is calculated by Equation (18),

$$\mathbf{v_j} = \mathbf{w}^{\mathbf{T}}[x_i||y_j], \tag{18}$$

where $w$ is defined in Equation (2), $x_i$ and $y_j$ are two vertices, while $||$ is a vector concatenation operator. By using hyperedges, it is possible to acquire hypergraph information through attention to the users and the items. Equation (1) can be used with the attention matrix to learn a layer-by-layer embedding. With $X(l)$ and $P$ used in Equation (1), the hypergraph's attention matrix propagates gradients to the incidence matrix.

Overall, in this article, we focus on higher-order user relations (user–user, stranger, and friend) and user–item interactions to learn user and item embedding vectors. Consequently, the attention operator aggregates information about the hyperedge nodes no matter whether the entity is a user (i.e., a stranger or friend) or an item. Thus, vectors of the potential features of users and items are produced through the $l$th layer [11].

### 4.3 User and Item Aggregation

One of the framework's strengths is that users and items are aggregated identically but separately. The results of the convolutional and attention networks of the hypergraphs are combined to identify the latent user and item features. User and item representations in the hypergraph convolutional network are denoted as $X_u$ and $X_i$, whereas the final user and item representations in the attention network are denoted as $E_u$ and $E_i$. In Equation (19), $O_u$ are the final user features, while in Equation (20), $O_i$ are the final item features,

$$\mathbf{O_u} = X_u \oplus E_u, \tag{19}$$

$$\mathbf{O_i} = X_i \oplus E_i. \tag{20}$$

**ALGORITHM 1:** HHGSA

**Input**:
$\mathcal{U}$: A user–user graph or social graph
$\mathcal{P}$: User-Item Graph
**Parameter** : embedding size $d$, number of layers $l$, learning rate $\alpha$
**Output**:
$\mathcal{R}$:Rating Prediction

Construction of hypergraph **H** based on five types of motif
**for** $e_i$ *in* **H do**
  //Calculate the hyperedges weight
  Compute hyperedge weights by using Equation (2)
**foreach** *user* $\in$ **H do**
  MLP with $Softmax(.)$ to calculate the hyperedge assignment of each user using Equation (3)
**foreach** *item* $\in$ **H do**
  MLP with $Softmax(.)$ to calculate the hyperedge assignment of each item using Equation (4)
**foreach** $MT$ *in* [“$Socialmotif''$, “$Usermotif''$, “$Itemmotif''$, “$Friendmotif''$, “$Strangermotif''$] **do**
  **if** *MT is* ***“Social motif”*** **then**
    Compute the social motif convolution $X_{social}$ using Equation (9)
  **if** *MT is* ***“User motif”*** **then**
    Compute the user motif convolution $X_{user}$ using Equation (5)
  **if** *MT is* ***“Item motif”*** **then**
    Compute the item motif convolution $X_{item}$ using Equation (6)
  **if** *MT is* ***“Friend motif”*** **then**
    Compute the friend motif convolution $X_{friend}$ using Equation (7)
  **if** *MT is* ***“Stranger motif”*** **then**
    Compute the strnger motif convolution $X_{str}$ using Equation (8)
  Concatenate the $X_{user}$ users representations $\{X_{user}^{(0)}, X_{user}^{(1)} \ldots X_{user}^{(l)}\}$ using Equation (10)
  Concatenate the $X_{item}$ items representations $\{X_{item}^{(0)}, X_{item}^{(1)} \ldots X_{item}^{(l)}\}$ using Equation (11)
  Use the same procedure for the $X_{friend}$, $X_{str}$, and $X_{social}$ representations.
  Create user features $X_u$ by combining $X_{user}$ and $X_{social}$ representations.
  Create item features $X_i$ by combining $X_{friend}$, $X_{str}$, and $X_{item}$ representations.
Attention network to compute the coefficient matrix using Equation (12).
Compute a weighted sum of the connected nodes in using Equation (14).
Compute the hyperedge-vertex attention coefficient matrix COE using Equation (15).
Apply COE in Equation (16) to produce users $E_u$ or item $E_i$ features.
Final user representations $O_u$ using Equation (19).
Final item representations $O_i$ using Equation (20).
Use inner product to evaluate user–item preference $\mathcal{R}$ using Equation (21).
Return $\mathcal{R}$

Finally, in Equation (21), the inner product is used to evaluate how much the user $u_j$ likes the intended item $i_k$,

$$\mathbf{r_{jk}} = O_{u_j}^T O_{i_k}, \tag{21}$$

where in $O_{u_j}$ the $u_j$ corresponds to row to $O_u$ and similarly the $i_k$ in $O_{i_k}$ corresponds to the row in $O_i$.

Table 1. Statistics of Datasets

| Dataset | Douban | Yelp | Epinions | Filmtrust |
|---|---|---|---|---|
| # of Users | 32,314 | 17,237 | 28,442 | 1,476 |
| # of Items | 14,109 | 38,342 | 128,240 | 1,844 |
| # of Interaction | 3,493,821 | 204,448 | 546,325 | 26,622 |
| # of Density (Rating) | 0.766% | 0.0309% | 0.01498% | 0.9781 |
| # of Social Relations | 331,315 | 143,765 | 334,305 | 1,609 |
| # of Density (Social Relations) | 0.0317% | 0.0484 % | 0.0413% | 0.0739% |

### 4.4 Model Optimization

Pairwise logistic optimization is used to optimize this model. It assumes that a particular user $u$ prefers an observed item $i$ over an unobserved item $j$. Therefore, the observed item $i$ should be ranked higher. Further, pairwise **Bayesian personalized ranking (BPR)** is used as the loss function [37],

$$\mathcal{L} = \sum_{(u,i,j)\in\tau} - \ln \sigma(\hat{r}_{u,i} - \hat{r}_{u,j}) + \lambda \|\Theta\|_2^2, \tag{22}$$

where $\tau$ represents the pairwise training data, $r_{u,i}$ is the predicted score of user $u$ for item $i$, $\Theta$ represents the model's parameters, and $\sigma(\cdot)$ represents the sigmoid function. A regularization parameter $\lambda$ helps to avoid overfitting.

## 5 EXPERIMENTS

In this section, we present the experiments conducted on four benchmark datasets in the field of recommender systems to assess HHGSA's effectiveness. We also conducted an ablation study to explore which components of the model contribute most to the performance improvements.

### 5.1 Datasets

The four datasets used in the experiments Yelp, Douban, Epinions, and Filmtrust. The statistics of the datasets are shown in Table 1.

Douban is a website where users can review books, music, and movies.

Yelp is an online social network where users rate and review local businesses. It serves personal, academic, and educational functions. In addition to expressing their experiences through ratings and reviews, users can also form friendships. The rating scale ranges between 0 and 5.

Epinions is a social network where users provide ratings and trust assertions for items. This network provides a massive amount of social and rating information on a scale of 0–5. It also includes timestamp information.

FilmTrust is another network for rating and sharing movies. The rating system for movies on this network spans 1 to 5. In addition, it provides social interactions via trustors and trustees.

### 5.2 Baselines

We compared HHGSA to three types of baselines. These types included (a) **classic methods of social recommendation (CMR)**, (b) **GNN-based Social Recommendation (GNNR)**, and (c) **hypergraph-based network recommendation systems (HGNR)**. The specific baselines compared in each category follow.

**The Classic Method of Social Recommendation**

- BPR [37]: Bayesian personalized ranking is a well-known ranking model that involves matrix factorization.
- SocialMF [23] is also based on matrix factorization, and trust information is propagated. This approach incorporates social impact among users.

**Graph Neural Network-based Social Recommendation**

- GraphRec [13] integrates social relations and considers user–user and user–item interactions.
- LightGCN [21] uses a graph convolution network to aggregate neighboring nodes. During message passing, nonlinear projections are removed, and sum-based pooling is used to generate the user representations.

**Hypergraph-Based Social Recommendation**

- DHCF [24] incorporates jumping hypergraph convolution for message passing. The divide and conquer strategy with two channels is used for collaborative filtering.
- MHCN [53] a method that uses self-supervised learning to enhance GNN-based recommendations. It maximizes graph-level and node-mutual information embedding.

## 5.3 Metrics and Parameter Settings

For a comprehensive comparison of HHGSA and the baselines, we used *NDCG*, *Recall*, and *Precision* as our evaluation metrics. The top-$k$ items were taken as the final recommendations for all models, where $k$ was set to both $K = 10$ and $K = 20$ across two sets of experiments. Since the models compute a prediction score for all items, ranked highest to lowest, we selected the top $k$ items for each baseline and applied *recall@K* and *precision@K* as the two relevance metrics and *NDCG@K* as the ranking metric. Precision@K shows how much each user likes the top $K$ recommended items, while Recall@K pertains to the rate of similar items in each user's top $K$ recommendations. In terms of the datasets, all records were used. So these three metrics measured performance with the entire dataset, not some sample data. Each experiment was conducted five times with the average results reported to ensure an unbiased assessment process. The regularization coefficient $\lambda$ was set to 0.001, while the user and item embedding dimensions $d$ were both set to $\{8, 16, 32, 64, 128, 256\}$. We searched for the optimal learning rate $\alpha$ in the range of $\{0.0005, 0.001, 0.005, 0.01, 0.05, 0.1\}$. The model was trained on an NVIDIA GeForce RTX 4050 GPU and built using the Pytorch 1.9 framework.

## 5.4 Experimental Results

The experimental results are provided in Table 2, with the best baseline values reported in bold. Overall, we made the following observations of the results.

- The evaluation metrics indicate that the GNN-based social recommendation models outperform the conventional classic models, which indicates that the skip-connection in Equations (5) to (9) are justified in that they support the explicit high-order relations required for collecting collaborative signals from a bipartite user–item graph.
- The hypergraph-based methods and the GNN-based methods showed similar performance. In certain instances, the GNN-based models would outperform the hypergraph-based models, while in other cases, the reverse was true. We attribute the poor performance of these hypergraph-based models to an improper construction approach.
- HHGSA performed better than the other baselines on all four datasets with increases in Precision of 11.81% to 31.23%, Recall improvements ranging from 1.82% to 28.62%, and NDCG improvements from 10.07% to 35.12%.

In short, HHGSA is highly robust and produces significantly better recommendations than its comparators due to the strength of its heterogeneous hypergraph neural network structure and its attention network.

Table 2. Comparison between the Models

| | | CMR | | GNNR | | HGNR | | **HHGSA** | |
|---|---|---|---|---|---|---|---|---|---|
| Dataset | Matric | BPR | SocialMF | GraphRec | LightGCN | DHCF | MHCN | **HHGSA** | **%Impr** |
| **Douban** | Precision@10 | 0.038 | 0.0261 | 0.0171 | 0.06840 | **0.5581** | 0.106 | **0.7324** | **31.23** |
| | Recall@10 | 0.0483 | 0.0168 | 0.05916 | 0.08294 | 0.3311 | **0.6848** | **0.7931** | **15.81** |
| | NDCG@10 | 0.0537 | 0.0931 | 0.0191 | 0.0990 | 0.2204 | **0.5954** | **0.7872** | **32.21** |
| | Precision@20 | 0.0423 | 0.31 | 0.0346 | 0.1887 | **0.5872** | 0.231 | **0.7433** | **26.58** |
| | Recall@20 | 0.0784 | 0.0565 | 0.1262 | **0.5684** | 0.394 | 0.4845 | **0.7311** | **28.62** |
| | NDCG@20 | 0.0616 | 0.0385 | 0.245 | **0.6874** | 0.3321 | 0.3621 | **0.7713** | **12.21** |
| **Yelp** | Precision@10 | 0.0323 | 0.055 | 0.023 | **0.6039** | 0.4637 | 0.5836 | **0.7165** | **18.65** |
| | Recall@10 | 0.0339 | 0.204 | 0.0607 | 0.4589 | 0.3563 | **0.6473** | **0.7311** | **12.95** |
| | NDCG@10 | 0.0397 | 0.133 | 0.04653 | **0.6742** | 0.5161 | 0.6572 | **0.7784** | **15.46** |
| | Precision@20 | 0.02689 | 0.0247 | 0.0564 | **0.7139** | 0.4541 | 0.612 | **0.7982** | **11.81** |
| | Recall@20 | 0.0587 | 0.0233 | 0.0845 | 0.538 | 0.3214 | **0.5431** | **0.6631** | **22.09** |
| | NDCG@20 | 0.0482 | 0.187 | 0.0583 | **0.6942** | 0.5717 | 0.683 | **0.7862** | **13.25** |
| **Epinions** | Precision@10 | 0.0067 | 0.00405 | 0.0331 | 0.01498 | 0.3037 | **0.5836** | **0.7465** | **27.91** |
| | Recall@10 | 0.0155 | 0.0401 | 0.0607 | 0.03956 | 0.3563 | **0.5673** | **0.6811** | **20.06** |
| | NDCG@10 | 0.0119 | 0.01036 | 0.02532 | 0.03066 | **0.5761** | 0.4572 | **0.7784** | **35.12** |
| | Precision@20 | 0.0053 | 0.0037 | 0.0604 | 0.1392 | 0.543 | **0.612** | **0.7582** | **23.89** |
| | Recall@20 | 0.0249 | 0.0212 | 0.0543 | 0.4387 | 0.3762 | **0.5771** | **0.6932** | **20.12** |
| | NDCG@20 | 0.0148 | 0.0197 | 0.0773 | 0.4932 | 0.5726 | **0.6435** | **0.7673** | **19.24** |
| **FilmTrust** | Precision@10 | 0.2580 | 0.2453 | 0.2322 | 0.2671 | 0.2436 | **0.5657** | **0.6665** | **17.82** |
| | Recall@10 | 0.4798 | 0.3104 | 0.04317 | **0.5379** | 0.3773 | 0.5141 | **0.6836** | **27.09** |
| | NDCG@10 | 0.04021 | 0.0125 | 0.05053 | 0.5722 | 0.4225 | **0.6069** | **0.7627** | **25.67** |
| | Precision@20 | 0.1953 | 0.1007 | 0.04865 | 0.1887 | 0.1788 | **0.5903** | **0.6982** | **18.28** |
| | Recall@20 | 0.713 | 0.03211 | 0.06543 | 0.7184 | 0.5716 | **0.7302** | **0.7435** | **1.82** |
| | NDCG@20 | 0.6739 | 0.1947 | 0.04783 | 0.6875 | 0.5136 | **0.7143** | **0.7862** | **10.07** |

The colors red and blue represent the best and second-best results, respectively.

### 5.5 Ablation Study

The components of the model tested in this Ablation Study include the hypergraph motifs, the dimension of the embeddings, and two variants of our model—one without the hypergraph construction and the other without the attention mechanism.

*5.5.1 Hypergraph Motifs Components.* Figure 4 shows the impact of each hypergraph motif to illustrate the model's interpretability. In this experiment, we disabled one hypergraph motif at a time and used *Precision*, *Recall*, and *NDCG* to determine the effect of each motif on the final result. For example "User Motif" means the user motif hypergraph is disabled. Recall that, the user motif relates to a set of items purchased by a user; the item motif related to users who purchased an item. The friend motif relates to friends purchasing an item, while the stranger motif covers strangers buying an item. Last, the social motif represents social relations.

The contribution of each hypergraph motif to the final outcome varies amongst datasets. However, the results indicate that the most prevalent hypergraph motif is item followed by the social motif and then the user motif. The attention network calculates item similarity using the item motif. The friend and stranger motifs have the least influence on the evaluation metrics, but they may improve model performance and contribute to the model's final output.

*5.5.2 Variants of Model:* We also conducted experiments to explore the influence of different parts of the model. The two variants tested were *HHGSA-GNN*, which does not include the hypergraph construction step or the earlier graph convolution, and *HHGSA-attention*, which does not include the attention network.

The results of these experiments are provided in Table 3. As shown, HHGSA performs better than the other variants. Without the hypergraph construction, *HHGSA-GNN* did not perform as well, because this model does not have the same capacity to extract high-order relations. Notably, however, HHGSA-GNN's performance was comparable to all the other baselines. From this result, we conclude that incorporating a hypergraph into the framework to represent the higher-order relationship is essential to maintaining the robustness of the model. Removing the attention network in *HHGSA-attention* delivered the worst results in the ablation. A probable explanation is that the model no longer computes the similarity between items. Consequently, we find that an attention mechanism is crucial to high-quality recommendations.

### 5.6 Parameter Sensitivity Analysis

This section reports the results of the following assessments: the sensitivity of the embedding size, the impact of the learning rate, and the effect that differing numbers of layers have on the result.

*5.6.1 Effect of Embedding Size:* Figure 5 shows the sensitivity of the embedding length with all four datasets at varying lengths. The results demonstrate that, as the embedding size rises from 8 to 128, performance improves and then declines for embedding sizes of 256 and more. These results indicate that a large embedding size may produce a trustworthy representation, but it will also increase the model's complexity.

*5.6.2 Influence of the Learning Rate:* The learning rate also influences the sensitivity of the HHGSA model. We searched for the optimal learning rate from 0.0005 to 0.1. Figure 6 illustrates the influence of different learning rates on performance. Generally, as the learning rate increased, performance increased, up to a point where it reached its maximum at a learning rate of 0.01, after which it began to decline gradually. Notably, this exact trend applied to all four datasets. Based on these results, one can conclude that even a small value for the learning rate can positively affect the recommendation task. By contrast, larger values may produce misleading outcomes.

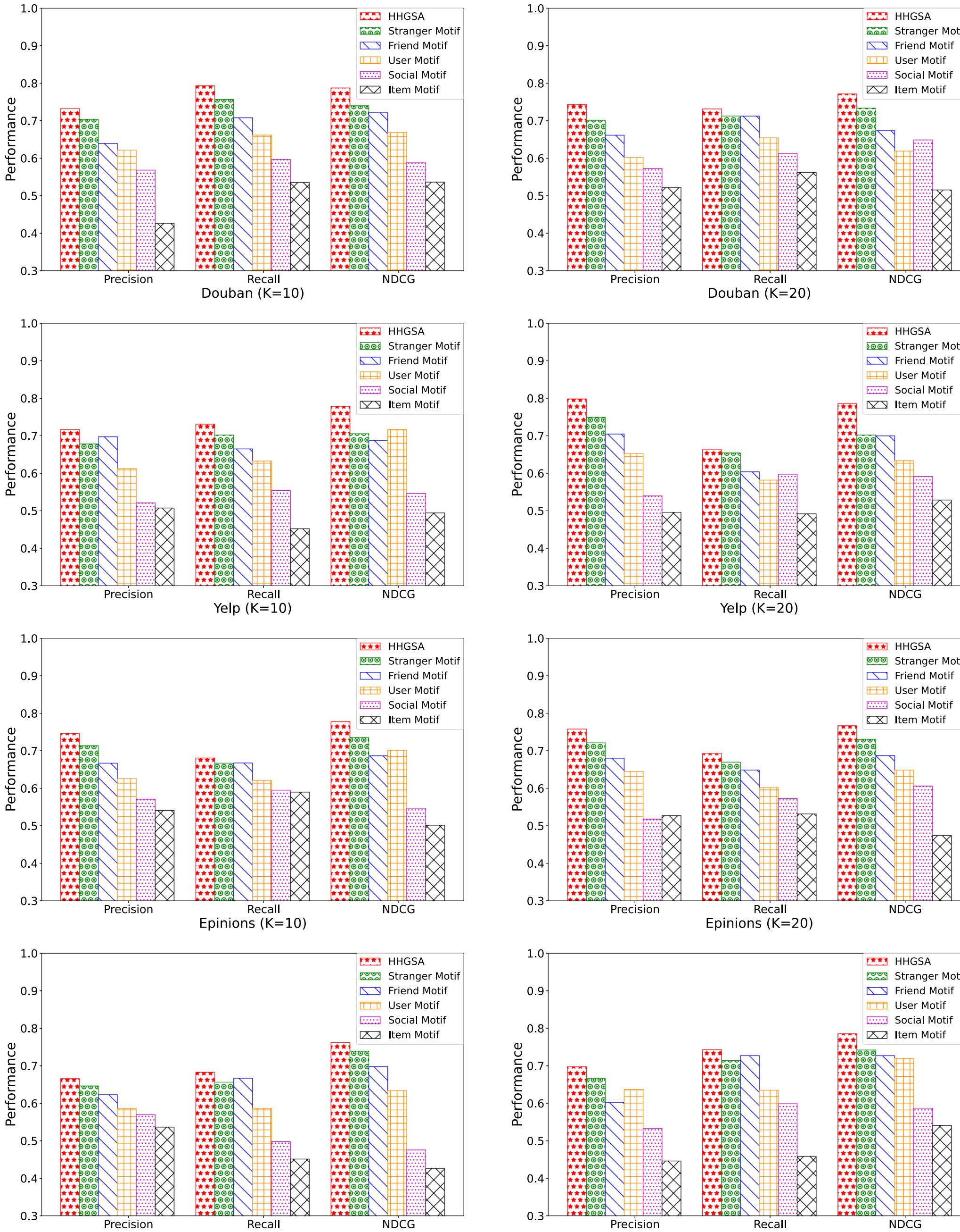


Fig. 4. Hypergraph motif components.

*5.6.3 Influence of the Number of Hypergraph Layers:* The next test was to investigate the influence of the number of hypergraph neural network layers from 1 to 5. Figure 7 shows the results. Here the best performance occurs with two layers, after which efficiency degrades. Again, this trend applies to all datasets.

Table 3. Experimental Comparison between Variants of HHGSA Model

| Dateset | Matric | HHGSA-GNN | HHGSA-attention | HHGSA |
|---|---|---|---|---|
| **Douban** | Precision@10 | 0.6381 | 0.6786 | 0.73241 |
| | Recall@10 | 0.6097 | 0.6421 | 0.7931 |
| | NDCG@10 | 0.5372 | 0.6143 | 0.7872 |
| | Precision@20 | 0.5923 | 0.631 | 0.7433 |
| | Recall@20 | 0.5784 | 0.6565 | 0.7311 |
| | NDCG@20 | 0.4616 | 0.5385 | 0.7713 |
| **Yelp** | Precision@10 | 0.6138 | 0.6610 | 0.7165 |
| | Recall@10 | 0.4783 | 0.5168 | 0.7311 |
| | NDCG@10 | 0.4537 | 0.5931 | 0.7784 |
| | Precision@20 | 0.5423 | 0.5431 | 0.7982 |
| | Recall@20 | 0.4784 | 0.5565 | 0.6631 |
| | NDCG@20 | 0.5898 | 0.6385 | 0.7862 |
| **Epinion** | Precision@10 | 0.6348 | 0.4217 | 0.7465 |
| | Recall@10 | 0.5483 | 0.6168 | 0.6811 |
| | NDCG@10 | 0.4537 | 0.5931 | 0.7784 |
| | Precision@20 | 0.6203 | 0.6315 | 0.7582 |
| | Recall@20 | 0.4784 | 0.5445 | 0.6932 |
| | NDCG@20 | 0.5616 | 0.6935 | 0.7673 |
| **FilmTrust** | Precision@10 | 0.4038 | 0.6279 | 0.6665 |
| | Recall@10 | 0.5411 | 0.5968 | 0.6836 |
| | NDCG@10 | 0.5373 | 0.5931 | 0.7627 |
| | Precision@20 | 0.4403 | 0.5991 | 0.6982 |
| | Recall@20 | 0.5784 | 0.5465 | 0.7435 |
| | NDCG@20 | 0.5616 | 0.6185 | 0.7862 |

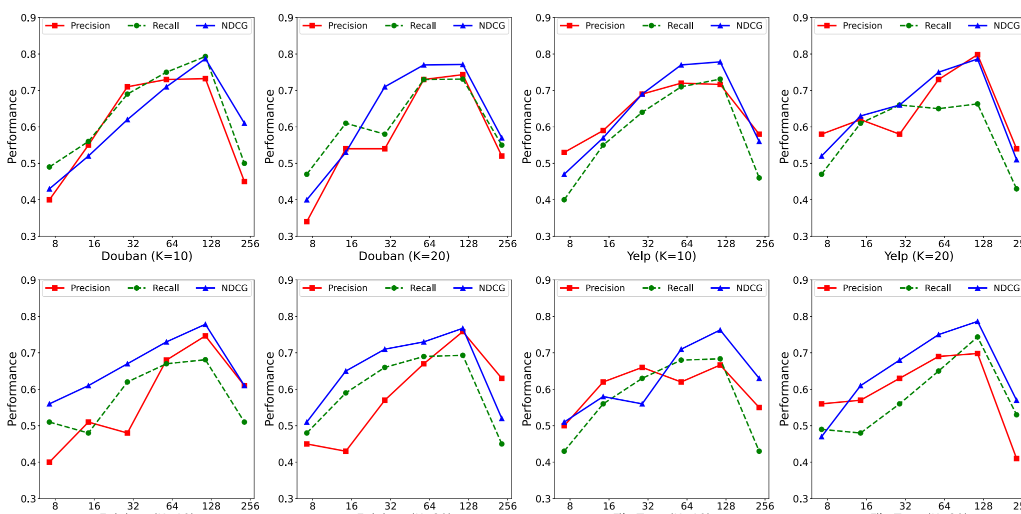


Fig. 5. Effect of embedding size.

## 6 RELATED WORK

This literature review is divided into the main categories of approaches to social recommendation: traditional approaches to social recommendation, those based on GNNs, and those based on hypergraphs.

### 6.1 Traditional Approaches for Social Recommendation

Social recommendation has been investigated since 1997 [27], and with the advent of social networks, it has gained more attention. The initial models were recommender systems based on

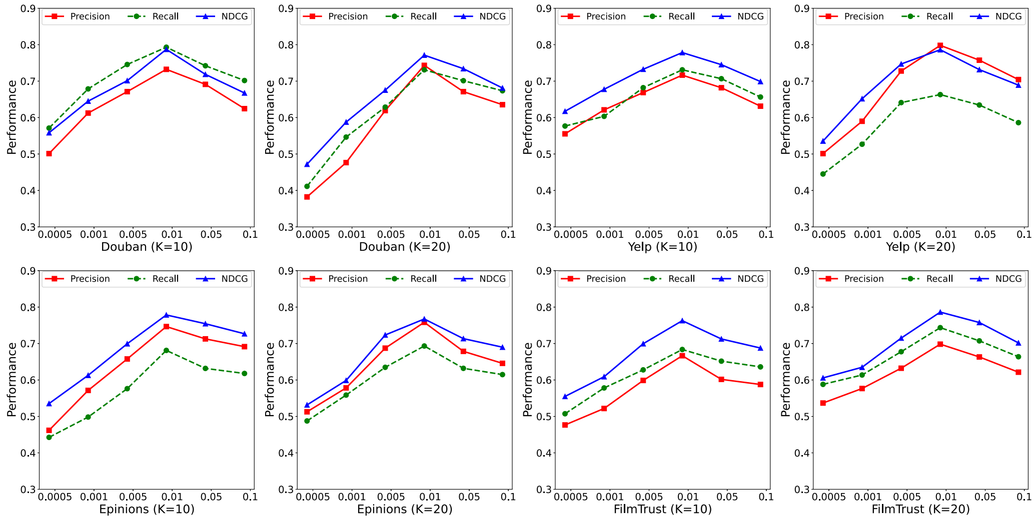


Fig. 6. Influence of learning rates.

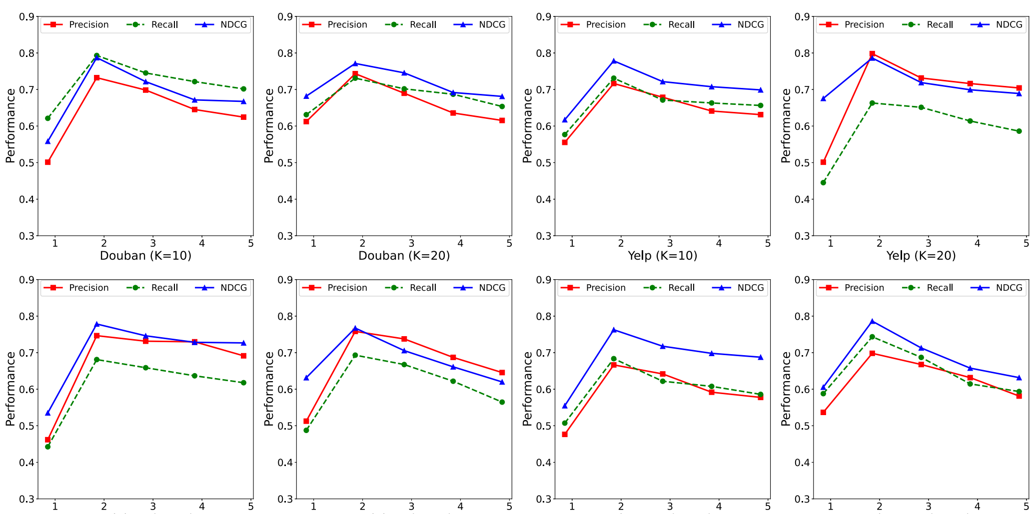


Fig. 7. Influence of the number of layers.

matrix factorization. According to social correlation theories, a user's preferences are influenced by the preferences of neighboring or socially connected users. Hence, with SoRec [31], a co-factorization approach was introduced that a social relation matrix and an item rating matrix by sharing latent user features. In TrustMF [50], the researchers factorize social trust networks and map the users as trusters and trustees in two low-dimensional spaces. Thus, they model the mutual influence between users. Focusing on cross-domain recommender systems, Zhang et al. [55] use item similarity features . Using these features helps to resolve the issues with cold starts and data sparsity. Meanwhile, Pavin et al. [35] use a trust network to improve social recommendation performance by assessing how the trust information influences the final verdict. ContextMF [42] merges the social network with the social context using a matrix factorization framework.

However, although these findings and advancements are promising, there are other areas in need of improvement.

### 6.2 Graph Neural Network–based Social Recommendation

With the advancement of deep learning technologies [54], GNN-based techniques have demonstrated ground-breaking performance in learning representations of graph data. Most commonly, GNNs are used to aggregate data and to learn the data representations [10]. GNNs may also encode the collaboration signals of user–item interactions to maximize the user–item representations through propagation. The state of each node is continuously updated through neighborhood information exchange until the node reaches stability, after which the network generates a prediction.

Fan et al. [12] integrated a neural network into a probabilistic matrix factorization. They used the pre-trained node embedding approach to represent users and $K$ nearest neighbors to link the neural network with the user embedding features. Berg et al. [3] presented a graph autoencoder approach that uses convolutional layers to generate the latent user and item features. However, these layers have an identical impact on all neighboring nodes, so the model may not accurately depict the network connections between the user and the item. In addition, if a node has many edges, then this situation could worsen.

With the GraphRec method [13], the authors aggregate data from three sources of information using an attention mechanism. The main components of this approach are item modeling, user modelling, and rating prediction. The main difference between our approach and the other GNN-based approaches is that all the other well-known approaches consider the item to be a user feature and not an independent node. Therefore, the items merely include users as nodes and have no link to one another. We may not be able to calculate the item–item similarity. Consequently, the neural network has one fewer dimension for data collection. By contrast, in our approach, a heterogeneous hypergraph is constructed and the relationship between the user, the item, and their social relations is considered.

### 6.3 Hypergraph-based Social Recommendation

Hypergraphs have been widely used to model complex higher-order relations and, with the advent of deep learning, researchers have looked to combine GNNs with hypergraphs to improve representation learning. HGNN [15] was the first study to propose spectral hypergraph convolution for learning hidden layer representations. Then Bu et al. [6] applied hypergraph learning to music recommender systems, treating the task as a hypergraph ranking problem. Their framework constructs user and item hypergraphs heuristically based on user–item interactions, followed by hypergraph convolution. Ji et al. [24] proposed a jump hypergraph convolution method, constructing user and item hypergraphs to help propagate high-order correlations.

Further, the benefits of hypergraph neural networks have been leveraged in recent studies such as HMF [56] and MHCN [53]. HMF [56] is a hybrid recommendation model that integrates a social network with a user–item interaction hypergraph. Note, though, that this model does not fully leverage higher-order relations. LBSN2Vec [51] constructs hyperedges through a random walk. It concentrates on linking various entities rather than using a higher-order social network. MHCN [27] uses a collection of motifs to pick out triangle structures from social networks. In fact, most of these methods based on hypergraphs focus on the triangular structures of the hypergraph or linking various entities rather than exploiting the higher-order relations. Further, the well-known approaches consider items to be user features instead of independent nodes. Therefore, the items merely include users as nodes and are not linked to one another. By contrast, in this article, we explored higher-order relations using hypergraph construction, calculating item–item and user–user similarity using an attention network.

## 7 CONCLUSION

This article outlines the HHGSA framework, which is designed to produce social recommendations using a heterogeneous hypergraph neural network. The hypergraph captures higher-order relationships through five hypergraph motifs—friend and stranger item attraction, item similarity, user similarity, and social relations while a hypergraph attention network assigns weights to the user and items within the embeddings. The proposed framework overcame the problems with higher-order relations and also identified the similarity between items and between users. When displayed to users, it has been observed that similar items always generate comparable levels of interest. In a set of experiments with social recommendation tasks, HHGSA outperformed six baselines across three categories of model. As with all studies, this implementation has its limitations. In this study the trust score is not considered when making social recommendations, which could reduce the accuracy and reliability of the recommendations. To overcome this issue, we plan to incorporate trust scores into the system in future work.

## REFERENCES


[1] Song Bai, Feihu Zhang, and Philip H. S. Torr. 2021. Hypergraph convolution and hypergraph attention. *Pattern Recogn.* 110 (2021), 107637.

[2] Saumya Bansal and Niyati Baliyan. 2019. A study of recent recommender system techniques. *Int. J. Knowl. Syst. Sci.* 10, 2 (2019), 13–41.

[3] Rianne van den Berg, Thomas N. Kipf, and Max Welling. 2017. Graph convolutional matrix completion. arXiv:1706.02263. Retrieved from https://arxiv.org/abs/1706.02263

[4] Filippo Maria Bianchi, Daniele Grattarola, and Cesare Alippi. 2020. Spectral clustering with graph neural networks for graph pooling. In *International Conference on Machine Learning*. PMLR, 874–883.

[5] Dheeraj Bokde, Sheetal Girase, and Debajyoti Mukhopadhyay. 2015. Matrix factorization model in collaborative filtering algorithms: A survey. *Proc. Comput. Sci.* 49 (2015), 136–146.

[6] Jiajun Bu, Shulong Tan, Chun Chen, Can Wang, Hao Wu, Lijun Zhang, and Xiaofei He. 2010. Music recommendation by unified hypergraph: combining social media information and music content. In *Proceedings of the 18th ACM International Conference on Multimedia*. 391–400.

[7] Chaofan Chen, Zelei Cheng, Zuotian Li, and Manyi Wang. 2020. Hypergraph attention networks. In *Proceedings of the IEEE 19th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom'20).* IEEE, 1560–1565.

[8] Robert B. Cialdini and Noah J. Goldstein. 2004. Social influence: Compliance and conformity. *Annu. Rev. Psychol.* 55, 1 (2004), 591–621.

[9] Lin Cui, Jia Wu, Dechang Pi, Peng Zhang, and Paul Kennedy. 2020. Dual implicit mining-based latent friend recommendation. *IEEE Trans. Syst. Man Cybernet. Syst.* 50, 5 (2020), 1663–1678.

[10] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. 2016. Convolutional neural networks on graphs with fast localized spectral filtering. *Adv. Neural Inf. Process. Syst.* 29 (2016).

[11] Kaize Ding, Jianling Wang, Jundong Li, Dingcheng Li, and Huan Liu. 2020. Be more with less: Hypergraph attention networks for inductive text classification. *2020 Conference on Empirical Methods in Natural Language Processing, EMNLP'20*, Association for Computational Linguistics (ACL), 4927–4936.

[12] Wenqi Fan, Qing Li, and Min Cheng. 2018. Deep modeling of social relations for recommendation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 32.

[13] Wenqi Fan, Yao Ma, Qing Li, Yuan He, Eric Zhao, Jiliang Tang, and Dawei Yin. 2019. Graph neural networks for social recommendation. In *Proceedings of the World Wide Web Conference*. 417–426.

[14] Xiaodong Feng, Zhen Liu, Wenbing Wu, and Wenbo Zuo. 2022. Social recommendation via deep neural network-based multi-task learning. *Expert Syst. Appl.* 206 (2022), 117755.

[15] Yifan Feng, Haoxuan You, Zizhao Zhang, Rongrong Ji, and Yue Gao. 2019. Hypergraph neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33. 3558–3565.

[16] Bairan Fu, Wenming Zhang, Guangneng Hu, Xinyu Dai, Shujian Huang, and Jiajun Chen. 2021. Dual side deep context-aware modulation for social recommendation. In *Proceedings of the Web Conference 2021.* 2524–2534.

[17] Li Gao, Jia Wu, Chuan Zhou, and Yue Hu. 2017. Collaborative dynamic sparse topic regression with user profile evolution for item recommendation. In *Proceedings of the 31st AAAI Conference on Artificial Intelligence.* 1316–1322.

[18] Li Gao, Hong Yang, Jia Wu, Chuan Zhou, Weixue Lu, and Yue Hu. 2018. Recommendation with multi-source heterogeneous information. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*. 3378–3384.

[19] Lei Guo, Hongzhi Yin, Qinyong Wang, Tong Chen, Alexander Zhou, and Nguyen Quoc Viet Hung. 2019. Streaming session-based recommendation. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 1569–1577.

[20] Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. *Adv. Neural Inf. Process. Syst.* 30 (2017).

[21] Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, Yongdong Zhang, and Meng Wang. 2020. Lightgcn: Simplifying and powering graph convolution network for recommendation. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 639–648.

[22] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural collaborative filtering. In *Proceedings of the 26th International Conference on World Wide Web*. 173–182.

[23] Mohsen Jamali and Martin Ester. 2010. A matrix factorization technique with trust propagation for recommendation in social networks. In *Proceedings of the 4th ACM Conference on Recommender Systems*. 135–142.

[24] Shuyi Ji, Yifan Feng, Rongrong Ji, Xibin Zhao, Wanwan Tang, and Yue Gao. 2020. Dual channel hypergraph collaborative filtering. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2020–2029.

[25] Zhengshen Jiang, Hongzhi Liu, Bin Fu, Zhonghai Wu, and Tao Zhang. 2018. Recommendation in heterogeneous information networks based on generalized random walk model and bayesian personalized ranking. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*. 288–296.

[26] Xiaojun Kang, Xinchuan Li, Hong Yao, Dan Li, Bo Jiang, Xiaoyue Peng, Tiejun Wu, Shihua Qi, and Lijun Dong. 2022. Dynamic hypergraph neural networks based on key hyperedges. *Inf. Sci.* 616 (2022), 37–51.

[27] Henry Kautz, Bart Selman, and Mehul Shah. 1997. Referral web: Combining social networks and collaborative filtering. *Commun. ACM* 40, 3 (1997), 63–65.

[28] Chun-Yi Liu, Chuan Zhou, Jia Wu, Yue Hu, and Li Guo. 2018. Social recommendation with an essential preference space. In *Proceedings of the AAAI Conference on Artificial Intelligence*. 346–353.

[29] Siwei Liu, Iadh Ounis, Craig Macdonald, and Zaiqiao Meng. 2020. A heterogeneous graph neural model for cold-start recommendation. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 2029–2032.

[30] Yuanfu Lu, Ruobing Xie, Chuan Shi, Yuan Fang, Wei Wang, Xu Zhang, and Leyu Lin. 2020. Social influence attentive neural network for friend-enhanced recommendation. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 3–18.

[31] Hao Ma, Haixuan Yang, Michael R. Lyu, and Irwin King. 2008. Sorec: Social recommendation using probabilistic matrix factorization. In *Proceedings of the 17th ACM Conference on Information and Knowledge Management*. 931–940.

[32] Miller McPherson, Lynn Smith-Lovin, and James M. Cook. 2001. Birds of a feather: Homophily in social networks. *Annu. Rev. Sociol.* (2001), 415–444.

[33] Koji Miyahara and Michael J. Pazzani. 2000. Collaborative filtering with the simple Bayesian classifier. In *Pacific Rim International Conference on Artificial Intelligence*. Springer, 679–689.

[34] Tulasi K. Paradarami, Nathaniel D. Bastian, and Jennifer L Wightman. 2017. A hybrid recommender system using artificial neural networks. *Expert Syst. Appl.* 83 (2017), 300–313.

[35] Hashem Parvin, Parham Moradi, Shahrokh Esmaeili, and Nooruldeen Nasih Qader. 2019. A scalable and robust trust-based nonnegative matrix factorization recommender using the alternating direction method. *Knowl.-Bas. Syst.* 166 (2019), 92–107.

[36] K. RaviKanth, K. ChandraShekar, K. Sreekanth, and P. Santhosh Kumar. 2019. Recommendation system for e-commerce by memory based and model based collaborative filtering. In *International Conference on Soft Computing and Pattern Recognition*. Springer, 123–129.

[37] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*, 452–461.

[38] Steffen Rendle, Walid Krichene, Li Zhang, and John Anderson. 2020. Neural collaborative filtering vs. matrix factorization revisited. In *Proceedings of the 14th ACM Conference on Recommender Systems*. 240–248.

[39] Changhao Song, Bo Wang, Qinxue Jiang, Yehua Zhang, Ruifang He, and Yuexian Hou. 2021. Social recommendation with implicit social influence. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 1788–1792.

[40] Jiliang Tang, Xia Hu, and Huan Liu. 2013. Social recommendation: A review. *Soc. Netw. Anal. Min.* 3, 4 (2013), 1113–1133.

[41] Donghui Wang, Yanchun Liang, Dong Xu, Xiaoyue Feng, and Renchu Guan. 2018. A content-based recommender system for computer science publications. *Knowl.-Bas. Syst.* 157 (2018), 1–9.

[42] Junfei Wang, Darshan Bagul, and Sargur Srihari. 2018. ContextMF: A fast and context-aware embedding learning method for recommendation systems.

[43] L. Wang, E.K. Egorova, and A.V. Mokryakov. 2018. Development of hypergraph theory. *J. Comput. Syst. Sci. Int.* 57, 1 (2018), 109–114.

[44] Youquan Wang, Zhiwen Dai, Jie Cao, Jia Wu, Haicheng Tao, and Guixiang Zhu. 2022. Intra- and inter-association attention network-enhanced policy learning for social group recommendation. In *Proceedings of the International World Wide Web Conference (WWW'22)*.

[45] Le Wu, Junwei Li, Peijie Sun, Richang Hong, Yong Ge, and Meng Wang. 2020. Diffnet++: A neural influence and interest diffusion network for social recommendation. *IEEE Trans. Knowl. Data Eng.* (2020).

[46] Le Wu, Peijie Sun, Yanjie Fu, Richang Hong, Xiting Wang, and Meng Wang. 2019. A neural influence diffusion model for social recommendation. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 235–244.

[47] Qitian Wu, Hengrui Zhang, Xiaofeng Gao, Peng He, Paul Weng, Han Gao, and Guihai Chen. 2019. Dual graph attention networks for deep latent representation of multifaceted social effects in recommender systems. In *Proceedings of the International World Wide Web Conference*. 2091–2102.

[48] Zhongxiu Xia, Weiyu Zhang, and Ziqiang Weng. 2021. Social recommendation system based on hypergraph attention network. *Comput. Intell. Neurosci.* (2021).

[49] Yang Xiao, Lina Yao, Qingqi Pei, Xianzhi Wang, Jian Yang, and Quan Z. Sheng. 2020. MGNN: Mutualistic graph neural network for joint friend and item recommendation. *IEEE Intell. Syst.* 35, 5 (2020), 7–17.

[50] Bo Yang, Yu Lei, Jiming Liu, and Wenjie Li. 2016. Social collaborative filtering by trust. *IEEE Trans. Pattern Anal. Mach. Intell.* 39, 8 (2016), 1633–1647.

[51] Dingqi Yang, Bingqing Qu, Jie Yang, and Philippe Cudre-Mauroux. 2019. Revisiting user mobility and social relationships in lbsns: A hypergraph embedding approach. In *Proceedings of the International World Wide Web Conference*. 2147–2157.

[52] Junliang Yu, Min Gao, Hongzhi Yin, Jundong Li, Chongming Gao, and Qinyong Wang. 2019. Generating reliable friends via adversarial training to improve social recommendation. In *Proceedings of the IEEE International Conference on Data Mining (ICDM'19)*. IEEE, 768–777.

[53] Junliang Yu, Hongzhi Yin, Jundong Li, Qinyong Wang, Nguyen Quoc Viet Hung, and Xiangliang Zhang. 2021. Self-supervised multi-channel hypergraph convolutional network for social recommendation. In *Proceedings of the World Wide Web Conference*. 413–424.

[54] Chenyan Zhang, Jing Li, Jia Wu, Donghua Liu, Jun Chang, and Rong Gao. 2022. Deep recommendation with adversarial training. *IEEE Trans. Emerg. Top. Comput.* 10, 4 (2022), 1966–1978.

[55] Qian Zhang, Jie Lu, Dianshuang Wu, and Guangquan Zhang. 2018. A cross-domain recommender system with kernel-induced knowledge transfer for overlapping entities. *IEEE Trans. Neural Netw. Learn. Syst.* 30, 7 (2018), 1998–2012.

[56] Xiaoyao Zheng, Yonglong Luo, Liping Sun, Xintao Ding, and Ji Zhang. 2018. A novel social network hybrid recommender system based on hypergraph topologic structure. *World Wide Web* 21, 4 (2018), 985–1013.

[57] Qiannan Zhu, Xiaofei Zhou, Jia Wu, Jianlong Tan, and Li Guo. 2020. A knowledge-aware attentional reasoning network for recommendation. In *Proceedings of the AAAI Conference on Artificial Intelligence*. 6999–7006.


Received 26 January 2023; revised 28 May 2023; accepted 24 July 2023