Research on Social Recommendation Algorithm Based on PSO_KFCM Clustering and CBAM Attention Mechanism of Graph Neural Networks

Yue Teng， Kai Yang

Abstract—In today's society, people increasingly need information acquisition due to the rapid development of science and technology and the consequent increase in available data. However, finding the information users need from this vast data has become challenging. To tackle this problem, recommending preferred information to users is becoming increasingly important. However, accurately recommending information by analyzing existing models such as GraphRec is still a challenging problem. A method called PSO_KFCM is proposed in this paper to solve this problem better. The technique combines Particle Swarm Optimization (PSO) with hybrid optimization and the kernel fuzzy C-means clustering technique to cluster similar recommendation data into one class. This way, the complexity and randomness of the recommendation data are reduced. It improves the speed and accuracy of the model prediction, which lays a solid foundation for the subsequent recommendation. Various factors will impact the recommendation process, and channel and spatial characteristics are essential. CBAM attention is added to the original attention mechanism to fully utilize these features in the recommendation data to enhance its performance. Furthermore, this paper proposes a social recommendation prediction method that combines CBAM attention and PSO_KFCM clustering and introduces a new social model called TTYGNN. The TTYGNN model optimizes the recommendation effect while maintaining the original advantages, enabling users to obtain the required information more quickly and accurately. To verify the effectiveness and practicality of the proposed model, extensive experimental comparisons were conducted on two widely used datasets. The results show that the TTYGNN model outperforms similar methods in all indicators, proving its superiority in information recommendation.

***Index Terms*—CBAM Attention Mechanism, Fuzzy clustering, Graph neural network, Recommendation system**

I. Introduction

In recent years, the extensive use of social networks in recommender systems has garnered significant attention. A standard view in these models is that users' preferences may be influenced by those around them (e.g., nearest neighbors), a view that can be well justified by social correlation theory. SoRec [1] provides this method of joint factor decomposition, an approach that uses ratings and social relationships to decompose a matrix of potential common user characteristics. TrustMF is a recommendation algorithm grounded in matrix factorization, integrating user ratings and social network information to enhance recommendation accuracy. The model improves the accuracy of recommendations by considering the relationship between users and items from the perspective of trusting and trusted users through their trust in the social network. SoDimRec [2] is a social dimension-based recommendation model designed to augment the precision of recommendations. The model forecasts users' ratings of goods by extensively calculating the similarity among users. In summary, SoDimRec portrays user relationships as a graph, where nodes symbolize users and edges denote their connections. The model then employs matrix decomposition techniques to learn the representations of nodes and edges, utilizing these learned representations to predict user ratings for items.

A Graph Neural Network (GNN) model called GraphRec was chosen and optimized to improve the existing social recommendation system. This model includes existing and new CBAM attentional mechanisms (Channel Attention Mechanism and Spatial Attention Mechanism) [3]. Particle Swarm Optimization (PSO) hybrid optimization of the Kernel Fuzzy C-mean Clustering Method (PSO_KFCM) was incorporated to enhance the original clustering. This new model aims to make more accurate predictions and recommend users' favorite content.

As shown in Figure 1, this paper proposes the TTYGNN model, which consists of three main components. Firstly, the user's social data is transformed into graph-structured data. This data is then fed into a graph neural network to perform complex transition pattern calculations and identify changes in user interests. The obtained information is then sent to the prediction layer, which generates accurate recommendation data. The optimal mean square error and average absolute error are calculated for various items, and suitable content for users is recommended based on the magnitude of these errors.

In our paper, we introduce a novel approach to address the obstacles faced in social recommendation systems. We propose a social recommendation model called TTYGNN, based on a combination of PSO_KFCM clustering of Graph Neural Network (GNN) and CBAM attention [3].

Manuscript received December 5, 2023; revised May 22, 2024.

Yue Teng is a postgraduate student at the School of Computer Science and Software Engineering, University of Science and Technology Liaoning, Anshan, 114051, China (e-mail:1127743361@qq.com).

Kai Yang is a Professor at the School of Computer Science and Software Engineering, University of Science and Technology Liaoning, Anshan, 114051, China (corresponding author, phone:86-15124122757; e-mail: asyangkai@126.com).

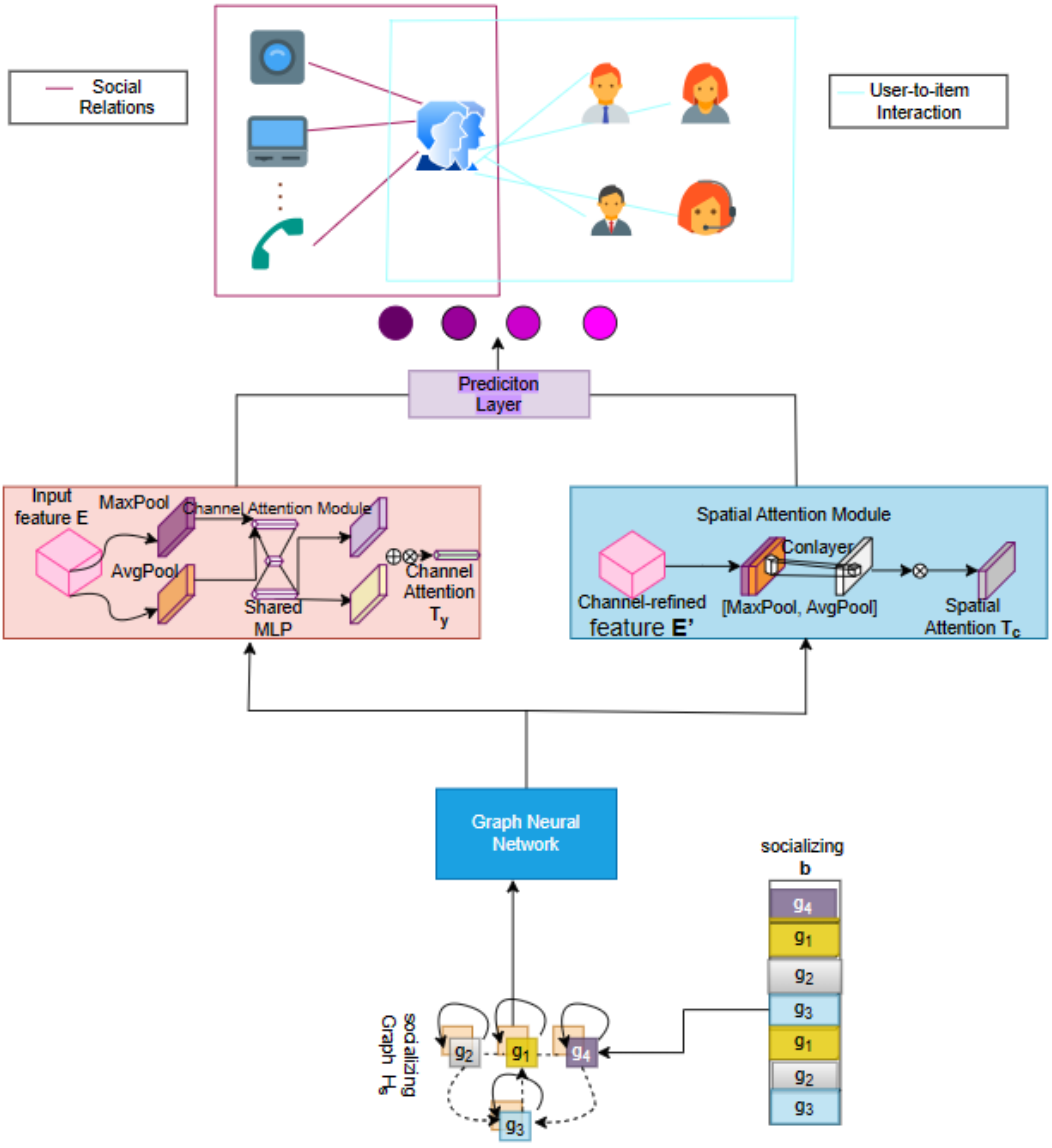

Fig. 1 Graph data in social recommendations. From bottom to top, the eight graphs are user-social data graphs, graph neural network graphs, spatial attention learning graphs, channel attention learning graphs, prediction layer graphs, user-item graphs, and user-user social graphs.

The TTYGNN model is proposed to improve prediction accuracy and speed while minimizing errors. The CBAM attention mechanism is integrated to derive the globally averaged feature vector across the channel dimension, with the channel attention module executing global average pooling on the feature map. The inclusion of a spatial attention mechanism allows the model to adaptively. Determine the attention weights of different regions, capturing crucial spatial information. The PSO_KFCM clustering is also integrated, reducing the data's complexity and randomness, leading to a substantial acceleration in prediction speed, enhanced accuracy, and minimized mean square error and average absolute error. The feasibility and efficacy of the TTYGNN model were tested on public datasets from two actual users, Ciao and Epinions. The experimental results show that the TTYGNN model significantly outperforms comparable models, confirming its validity and practicality.

Overall, this paper is innovative in the following respects:

1) A new graph neural network model, TTYGNN, is proposed to increase prediction accuracy while reducing prediction time.

2) The CBAM attention mechanisms are added to the original model to enhance significant features and reduce irrelevant ones.

3) Integrating PSO_KFCM clustering into the original social and user clustering models provides efficient search capabilities and ease of implementation. It also inherits KFCM's superior performance in managing complex data.

4) Extensive experiments have been conducted to confirm the feasibility and effectiveness of the proposed TTYGNN model.

In the following sections, we will discuss the proposed framework in detail. Firstly, in Section II, we will present the framework in detail. After that, in Section III, we will conduct extensive experiments on two authentic datasets to demonstrate our approach's practical validity and feasibility. Section IV will provide an overview of other frameworks similar to our proposed one. Finally, in Section V, we will summarize the findings of this work and articulate future research directions and goals, particularly within the realm of social recommender systems.

II. Proposed Framework

In this section, we will begin by providing definitions and terminology. We will then elaborate on the modeling process and its constituent elements. Finally, we will present an in-depth discussion of methods for learning model parameters.

TABLE I
NOTATION

Symbol	Definitions and Descriptions
E	Indicates the eigenvalue.
Θ	denotes element-by-element multiplication.
$E^{''}$	Indicates final refined output.
λ	denotes the sigmoid function.
$E_{\text{avg}}^{\text{c}}$	denotes the average pooling characteristics.
$E_{\max}^{\text{c}}$	denotes the maximum pooling characteristic.
$\text{p}^{7\times7}$	denotes a convolution operation with a filter size of 7 × 7.
V	Denote the affiliation matrix.
A	Indicates a data point.
C	An array indicating the number of classes into which the number is to be divided.
V_{ij}	denotes the degree of affiliation of sample A_{i} to class X_{j}.
a_1a_2	denotes the learning factor.
n_1n_2	denoted as a uniform random number in the range.
k	Indicates a weighted index.
μ	denotes a nonlinear mapping.
g	Represents a collection of items for socialization.
d_{cw}	User x_{c} rating value for item v_{w}.
$\text{d}_{cw}^{'}$	User x_{c} predicted score value for item v_{w}.
g_i	Indicates items that users have clicked on in the social s.
$p^{7\times7}$	denotes the convolution operation with a filter size of 7×7.
q	Indicates the current iteration step.

A. Constructing social graphs

Let $G=\{\text{g}_1,\text{g}_2,\cdots,\text{g}_m\}$ denote the set encompassing all unique items associated with a social user, termed the itemset. Indicate the number of items with m. The unknown user social sequence is represented by $\text{s}=\{g_1,g_2,\cdots,g_n\}$, where $\text{g}_i\in G$ signifies the item clicked by the user during the social interaction. For instance, the social sequence $\text{s}=\{g_1,g_2,\cdots,g_n\}$ is structured as a weighted directed social graph $H_s=(G_s,J_s)$, within which $H_s\in H$, H constitutes the ensemble of all social graphs. In social graph H_S, the set of nodes G_s encompasses all the item nodes in this network, each embodying an item $g_i\in G_s$. The set of edges J_s denotes the set of all outgoing edges. Each directed edge $(g_i,g_{i+1})\in J_s$ indicates that the user clicked on g_i an item within the social network s and subsequently clicked on another item g_{i+1}.

Following the processing above, the social graph is fed into the model to extract the latent information. Predict the item g_{n+1} that the user is likely to click on, based on the user's social sequence s. Input social sequence is used for iterative computation of optimal mean square error and average absolute error in the learning and prediction layers.

B. Learning the embedding vectors of social graphs

In this section, we incorporate the social graph into the graph neural network to extract embedding information of the global items. We can capture users' long-term interests by analyzing the intrinsic sequential patterns within user conversations. To derive the global item embedding formula, we input the social graph into the graph neural network. We have chosen a gated graph neural network [4] to learn the embedding vectors of the nodes within the graph. This constitutes a classical GRU-based spatial message-passing model that utilizes inter-nodal communication for iterative information updating. For the social graph H_S, the update function of the node vectors can be articulated as follows:

$$d_i^t=D_{I:}[l_1^{t-1},\ldots,l_n^{t-1}]^T\text{L}+\text{b} \tag{1}$$

$$w_i^t=\gamma(F_z d_i^t+M_z l_i^{t-1}) \tag{2}$$

$$e_i^t=\gamma(F_r d_i^t+M_r l_i^{t-1}) \tag{3}$$

$$l_l^t=\left(\text{F}_\text{O}\text{d}_\text{i}^\text{t}+\text{M}_\text{o}(\text{e}_\text{i}^\text{t}\otimes l_i^{t-1})\right) \tag{4}$$

$$l_i^t=(1-w_i^t)\otimes l_i^{t-1}+w_i^t\otimes l_l^t \tag{5}$$

Where d_i^t represents the outcome of the bi-directional transfer of node information interaction, L signifies the control weights, w_i^t denotes the update gate, $\gamma(\bullet)$ refers to the Sigmoid function, and $F_zF_rF_o$ indicates the matrix of learnable parameters. $M_zM_rM_o$ denotes a matrix that is learnable, e_i^t signifies a reset gate, $\otimes$ represents a dot product operator, and l_i is equivalent to g_i, the initial vector.

Consider the socialization sequence $s=[g_{4,}g_1,g_2,g_3,g_1,g_2,g_3]$ as an example. Equations (2) to (5) mirror the GRU computation process, in which information is updated through the generation and forgetting

of control information. Ultimately, the final vector for each node is obtained through multiple computations.

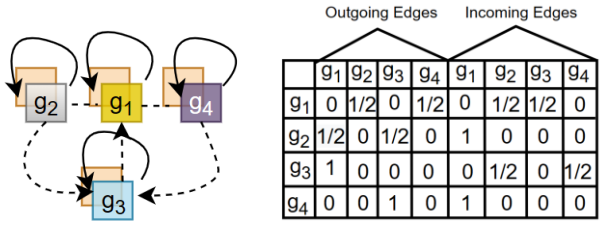

Fig. 2 Social graph and adjacency matrix.

C. CBAM attention-based mechanism

The CBAM attention mechanism gained popularity in 2018 thanks to the work of Jeonghee Choo et al. This mechanism features the introduction of both channel attention and spatial attention modules. Unlike conventional attention mechanisms, the channel attention module is applied first, followed by the spatial attention module. This sequential design enables the model to identify the target object more effectively and suppress irrelevant noise information.

It is necessary to consider channel and spatial factors to improve the accuracy of recommendation data forecast, Which is where the CBAM attention mechanism comes in. Incorporating this mechanism enhances the ability to capture long-term dependencies, resulting in better prediction accuracy.

With the given intermediate feature map $E \in \mathbb{R}^{C\times H\times W}$ as input, CBAM sequentially generates a 1D channel attention map $T_y \in \mathbb{R}^{C\times 1\times 1}$ and a 2D spatial attention map $T_c \in \mathbb{R}^{1\times H\times W}$. The computational formula is as follows:

$$E^{'} = T_y(E)\Theta E \tag{6}$$

$$E^{''} = T_c(E^{'})\Theta E^{'} \tag{7}$$

During the multiplication process, attention values are broadcasted as needed, and channel attention values propagate across the spatial dimension. Figure 4 illustrates the computation process.

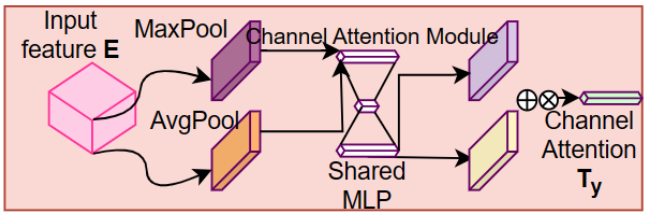

Fig. 3 Channel attention mechanisms mechanisms map.

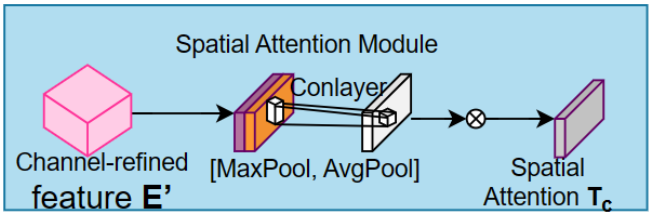

Fig. 4 Spatial attention mechanisms map.

Initially, the spatial information of the feature map is aggregated, and two distinct spatial context descriptors are produced by applying average pooling and maximum pooling operations: E^{y}_{avg} and $E^{y}_{\max}$, representing the average-pooled features and the maximum-pooled features, respectively. The two descriptors are fed into a shared network to generate channel attention, as shown in Figure $T_y \in \mathbb{R}^{C\times 1\times 1}$. The input feature maps' spatial dimensions are compressed, and average pooling is used to consolidate spatial information in a shared network consisting of a single-hidden-layer multilayer perceptron (MLP), to minimize parameter overhead. The size of the hidden activation is set to $\mathbb{R}^{C/r\times 1\times 1}$, where r represents the reduction ratio. The channel attention is computed as follows:

$$\begin{aligned} T_y(E) &= \lambda(MLP(AvgPool(E)) + MLP(MaxPool(E))) \\ &= \lambda(V_1(V_0(E^{y}_{avg})) + V_1(V_0(E^{y}_{\max}))) \end{aligned} \tag{8}$$

Where, λ denotes the $sigmoid$ function, $V_0 \in \mathbb{R}^{C/r\times C}$, $V_1 \in \mathbb{R}^{C\times C/r}$. Note that MLP weights MLP and V_1 are shared for both inputs.

The channel information of the aggregated feature map is processed using two types of pooling to yield two 2D maps: $E^{c}_{\text{avg}} \in \mathbb{R}^{1\times H\times W}$ and $E^{c}_{\max} \in \mathbb{R}^{1\times H\times W}$. These are the average and maximum pooled features in the channel, respectively, followed by the generated 2D spatial attention graph, which is computed as:

$$\begin{aligned} T_c(E) &= \lambda(p^{7\times 7}([AvgPool(E); MaxPool(E)])) \\ &= \lambda(p^{7\times 7}([E^{c}_{avg}; E^{c}_{\max}])) \end{aligned} \tag{9}$$

Where, $T_c(E) \in R^{H\times W}$, λ denotes the $sigmoid$ function, and $p^{7\times 7}$ denotes the convolution operation with a filter size of 7×7.

D. Fuzzy c-mean clustering fcm

The FCM algorithm is a clustering technique that optimizes an objective function [5]. It uses the concept of geometric proximity to assign data points to different clusters and calculate the distances between them. The algorithm follows these specific steps:

1) To initialize the affiliation matrix Y, random values between 0 and 1 are required, adhering to the constraints specified in Eq.

$$\sum_{j=1}^{s} V_j(A_i) = 1, \quad \text{i=1,2,}\cdots\text{,n} \tag{10}$$

Where the sample in the data is assumed to contain $A = \{a_1, a_2, a_3, \cdots, a_n\}$ data points.

$C(2 \le c \le n)$ indicates the number of classes into which the data is to be divided.

2) Calculate C cluster centers, $C_i(i=1,2,\cdots,C)$.

$$C_{\mathrm{i}}=\frac{\sum_{j=1}^{n}V_{ij}^{\ m}\cdot A_j}{\sum_{j=1}^{n}V_{ij}^{\ m}} \tag{11}$$

3) Determine the value function and determine if the algorithm should stop based on its value and the degree of change.

$$\mathrm{J}_{\mathrm{b}}(V,h)=\sum_{i=1}^{n}\sum_{k=1}^{c}(V_{ik})^{b}(\mathrm{d}_{ik})^{2} \tag{12}$$

Where v is the affiliation matrix comprised of data points of similar types, with the clustering centers of the distinct kinds denoted by h, and the individual class clustering centers encompass $\{h_1,h_2,h_3,\cdots,h_c\}$.

4) Calculate the new affiliation matrix, i.e., calculate the degree of affiliation of the sample A_i to the class X_i using the following formula V_{ij} . Pass the calculation result to step 2 to compute the new clustering center.

$$V_{ij}=\frac{1}{\sum_{k=1}^{c}(\frac{d_{ij}}{d_{kj}})^{\frac{2}{m-1}}} \tag{13}$$

The fuzzy clustering process involves refining the clustering center and degree of affiliation through steps 2 and 4. This facilitates the classification of data points into distinct categories. Once the algorithm converges, it signifies the successful completion of fuzzy clustering for all data points.

E. Particle swarm optimization

In 1995, Kennedy and Eberhart [6] introduced the Particle Swarm Optimization (PSO) algorithm, a global optimization technique. This algorithm is inspired by the collective behaviors observed in flocks of birds and schools of fish. In the PSO algorithm, each particle mimics an individual's behavior within a flock of birds or a school of fish, possessing its direction and velocity. The direction of movement reflects the particle's current position, and it's essential to consider the interaction between individual particles and the rest of the particle swarm. The Particle Swarm Optimization algorithm identifies individual extrema and the global optimum [7] to modify the velocity and position.

Consider a colony of M particles in a Y-dimensional target search space. Each particle, denoted as l , can be represented by a Y-dimensional vector, symbolized as $F_i=(f_{l1},f_{l2},\ldots,f_{1y}),\quad i=1,2,\cdots M$. Additionally, the velocity of the particle l can be defined as $V_l=(v_{l1},v_{l2},\ldots,v_{1y}),\quad l=1,2,\cdots M$. The optimal position found by the particle l up to this point is termed the individual extremum, denoted as $B_{best}=(b_{l1},b_{l2},...,b_{1y}),\quad l=1,2,\cdots M$. The globally optimal solution explored by the entire swarm of particles is denoted as $S_{best}=(b_{s1},b_{s2},\ldots,b_{sy}),\quad l=1,2,\cdots M$. The particle refines its position by tracking two "poles," (B_{best},S_{best}) , and updates its velocity and location using a particular formula:

$$\begin{aligned}v_{ly}(t+1)&=v_{ly}(t)+a_1n_1(t)[b_{ly}(t)-x_{ly}(t)]\\&+a_2n_2(t)[p_{sj}(t)-x_{ly}(t)]\end{aligned} \tag{14}$$

$$k_{ly}(t+1)=k_{ly}(t)+v_{ly}(t+1) \tag{15}$$

Among these, $l=1,2,\cdots M$ represents the total number of particles in the swarm, a_1 and a_2 are learning factors, also collectively termed acceleration constants; n_1 and n_2 are uniform random numbers within the range [0,1], which augment the randomness of particle movement; v_{lj} denotes the particle velocity, while $v_{lj\in}[-v_{\max},v_{\max}]$ and $v_{\max}$ are user-defined constants that serve to constrain the particle's velocity. The specific calculation process of PSO is as follows:

1) Initialize the parameters of the particle swarm algorithm, including the overall size M , the velocity v_l and position f_l of each particle, the acceleration constant, the maximum number of iterations, and the minimum allowable error.

2) Determine the initial fitness value for each particle $flt_{[l]}$. The fitness function is utilized to appraise the quality of a particle within a given solution space, with calculations derived from the particle's initial velocity and position values.

3) For each particle within the swarm, its fitness value, $flt_{[l]}$, is compared to the current individual extreme value, b_{ly}. If $flt_{[l]}>b_{ly}$ is true, then replace b_{ly} with $flt_{[l]}$.

4) For each particle in the swarm, compare the particle's fitness value, $flt_{[l]}$, with the global extreme value, b_{sy}. If $flt_{[l]}>b_{sy}$ is true, then replace b_{sy} with $flt_{[l]}$.

5) Update the velocity, v_l, and position, f_l, of the particle iteratively based on the two equations above.

6) Boundary condition processing confines particle positions within a feasible search space to prevent positions or velocities from exceeding predetermined values.

7) Check if the maximum number of iterations or convergence conditions is met. If yes, terminate the algorithm and present optimized particle results. If not, go back to step 2 for more iterations.

F. Kernel fuzzy c-mean clustering

The traditional FCM [8] algorithm uses Euclidean distance as the distance function, which may only sometimes be effective for clustering. While FCM has good performance in various applications, it has its issues. Due to the random initialization of cluster centers, the clustering centers and affiliation matrices tend to converge towards local optima. Moreover, FCM is sensitive to noise and outliers, which can result in incorrect allocation of these data points to specific clustering centers, thereby affecting the accuracy of the clustering outcomes. The kernel function is integrated into the FCM algorithm to overcome these challenges, resulting in the kernel fuzzy C-means clustering algorithm (KFCM). This method uses the kernel function to project finite-dimensional

data into a higher-dimensional space. By incorporating the kernel function, the computational load associated with inner product operations in the high-dimensional feature space is substantially reduced, thus enhancing computational efficiency. KFCM is an unsupervised algorithm designed to address traditional FCM's limitations. It can effectively manage nonlinear data, improve clustering performance, and uncover hidden data structures by leveraging data features without external intervention.

KFCM maps the data to a high-dimensional feature space by transforming the distance function in the FCM algorithm via a nonlinear mapping, $\Omega: A \to Q: \phi(a) = y$. Partitioning the dataset with a linear function in the high-dimensional feature space results in more precise sample clustering than the conventional FCM algorithm. The algorithmic computations are as follows:

1) Given the original feature dataset, A, the number of clusters, h, and the convergence accuracy, ζ, the clustering centers are initialized using the FCM algorithm, N_0.

2) Definition $U_i (i = 1, 2, \cdots, n)$ represents the affiliation function of class i for sample j, and the objective function in KFCM is expressed by the subsequent equation:

$$\begin{aligned} J_{KFCM} &= \sum_{i=1}^{k}\sum_{j=1}^{n} v_{ij}^{m} \| \Theta(A_j) - \Theta(U_i) \|_2^2 \\ &= \sum_{i=1}^{k}\sum_{j=1}^{n} v_{ij}^{m} \left(2 - 2K(A_j - U_i)\right) \\ &= 2\sum_{i=1}^{k}\sum_{i=1}^{n} v_{ij}^{m} \left(1 - K(A_j - U_i)\right) \end{aligned} \tag{16}$$

Where Θ denotes the nonlinear mapping, and m is the weighted exponent. $\Theta(A_j)$ and $\Theta(U_i)$ represent the mapping of high-dimensional sample data and the images of clustering centers, respectively, from the original input space to the high-dimensional feature space. $K(A_j - U_i)$ is the Gaussian radial basis function, which can be formulated by the equation below:

$$K(A_j - U_i) = e^{-\frac{\| A_j - U_i \|^2}{2\omega^2}} \tag{17}$$

Where ω denotes the kernel radius and is the scaling parameter.

3) The values for the degree of affiliation, $v_{ij}^{(q+1)}$, and the clustering center, $U_i^{(q+1)}$, can be derived from the Lagrangian extreme value method, as articulated by the following formula:

$$v_{ij}^{(q+1)} = \frac{(1 - K(A_j, U_i))^{\frac{-1}{m-1}}}{\sum_{i=1}^{k} (1 - K(A_j, U_i))^{\frac{-1}{m-1}}} \tag{18}$$

$$U_i^{(q+1)} = \frac{\sum_{j=1}^{n} v_{ij}^{m} K(A_j, U_i) A_j}{\sum_{j=1}^{n} v_{ij}^{m} K(A_j, U_i)} \tag{19}$$

where q denotes the current iteration step.

4) The values of affiliation, $v_{ij}^{(q+1)}$, and clustering center, $U_i^{(q+1)}$, are computed iteratively until the convergence condition $\| J_{KFCM}{}^{(q)} - J_{KFCM}{}^{(q-1)} \| < \zeta$ is met, or the maximum number of iterations, N, is reached. Otherwise, the iteration advances to step $q = q + 1$.

G. PCA-PSO_KFCM clustering

This paper discusses the challenges faced by KFCM, a clustering algorithm that addresses subpar performance in FCM due to noise and outliers. Despite its effectiveness, KFCM still needs help with issues such as local optima and reduced computational efficiency when dealing with high-dimensional data. To overcome these challenges, the paper proposes a new method integrating Particle Swarm Optimization (PSO) with the KFCM algorithm. The PSO algorithm is used to identify the optimal subset of features, avoiding the pitfall of getting trapped in local optima and seeking the global optimal solution. Additionally, PSO can automatically tune clustering parameters, thus improving the accuracy and robustness of the clustering outcomes. The detailed computational workflow of the proposed method is depicted in Figure 5.

The PSO-KFCM clustering process is outlined below:

1) The raw data is preprocessed to remove outliers and jittery adjacent values, ensuring a consistent scale across all features.

2) Initialize the Particle Swarm Optimization (PSO) parameters and establish the configurations for the algorithm. This encompasses the number of particle clusters, N, the learning factors, N and a_2, the maximum iterations for KFCM, the inertia weight, and the parameter boundaries' minimum and maximum values.

3) The swarm's adaptive value is calculated using Equation (20) and compared to the previous generation's value for each particle. If the current fitness value is greater than the previous generation's fitness value, it replaces the last individual generation as the new individual b_{best}, and updates the fitness value to the individual's optimal fitness value. If no modifications are made, then it will remain in its current state without any alterations.

$$y_{flt} = \frac{1}{1 + J_{\mathrm{KFCM}}(V, U)} \tag{20}$$

Where y corresponds to the output data.

4) Compare the individual fitness value against the group's optimal individual fitness value; if the individual's fitness value exceeds that of the group's optimal, replace the global optimal individual s_{best} with this individual and update both the group's optimal and associated fitness values. Compare with the group optimal individual fitness.

5) Update the velocity and position of the particle using Equations (18) and (19).

6) The algorithm terminates When it reaches the maximum number of iterations or a predetermined threshold surpassing the variation in the fitness value of the cluster's best global position—indicating no further enhancement in the population's fitness value. Otherwise, continue with step 3.

7) Upon meeting the termination condition of the Particle Swarm Optimization (PSO) algorithm, the initial cluster center matrix and the initial affiliation matrix are output.

8) Please update the cluster center matrix and affiliation matrix using Equations (18) and (19).

9) When the maximum number of iterations, N, is reached or the condition $\| J_{KFCM}{}^{(q)} - J_{KFCM}{}^{(q-1)} \| < \zeta$ is satisfied, terminate and output the final result. Otherwise, proceed to step 8.

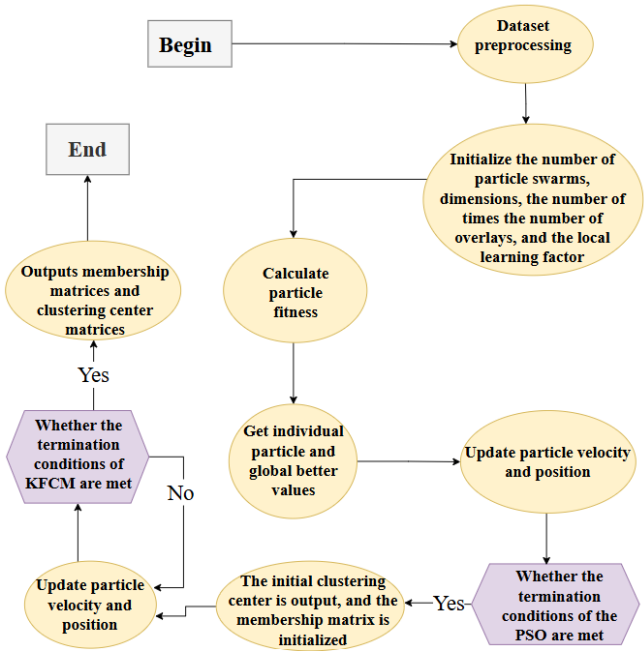

Fig. 5 PSO _KFCM specific flow chart.

H. Evaluation of projections

This section will create recommendation tasks to learn the model parameters. There are different types of recommendation tasks, such as item ranking and rating prediction. For this project, we will use the TTYGNN model for the rating prediction recommendation task. The user and item latent factors will be linked and fed into a multilayer perceptron (MLP) for prediction, which will help us achieve our objective. The calculations for this process are as follows:

$$y = [z_c \Leftrightarrow g_w] \tag{21}$$

$$y_2 = \sigma(F_2 \cdot y_1 + e_2) \tag{22}$$

$$\cdots$$

$$y_{l-1} = \sigma(F_l \cdot y_{l-1} + e_l) \tag{23}$$

$$b_{cw}^{'} = F^T \cdot y_{l-1} \tag{24}$$

where l denotes the index of the hidden layer, and b_{cw} represents the predicted rating from user j_c to item i_w. z_c signifies the user latent factor of the user j_c, derived from the amalgamation of item space z_c^C and social space z_c^s.

z_c^C represents the item space user latent factors derived from the itemset $B(\mathrm{c})$ of user j_c.

z_c^s represents the latent factors of z_c^s social space users, specifically from the social friends j_c of the user a_i. The calculations are as follows:

$$W_{ei} = y_\upsilon([h_a \odot p_r]) \tag{25}$$

Opinion-aware interactions are denoted as W_{ei}, and through the integration of h_a and p_r, which embed opinions, y_υ symbolizes the fusion of interaction information with opinion information.

$$z_c^C = \sigma(F \cdot \left\{ \sum_{i \in B(c)} i_c w_{ei} \right\} + e) \tag{26}$$

The elementary mean of the vectors within $\left\{ w_{ei}, \forall_i \in B(c) \right\}$ serves as a mean-based aggregator, which can be regarded as a linear approximation of the local spectral convolution, as mentioned in [10].

Where i_c is fixed to $\frac{1}{|B(c)|}$, for all items in the mean-based aggregator. This approach assumes that every interaction equally impacts the user's understanding. However, this may only sometimes be accurate as some interactions may be more important than others. Therefore, it is recommended that weights be assigned to each interaction. This allows for different interactions to have varying levels of contributions to the user's underlying factors.

$$z_c^S = \sigma(F \cdot \left\{ \sum_{o \in A(i)} \partial_i z_o^C \right\} + e) \tag{27}$$

The 、 elementary mean of the vector $\left\{ z_o^C, \forall_i \in A(i) \right\}$ is calculated, with $\left\{ z_o^C, \forall_i \in A(i) \right\}$ fixed value $\frac{1}{|A(i)|}$, for all items within the mean-based aggregator. Where z_o^C signifies social attention, F and e denote the attention weights respectively, $b_{cw}^{'}$ represents the prediction ratings from j_c to i_w, and l indicates the index of the hidden layer.

An objective function must be chosen to optimize the parameters of TTYGNN. Since the focus is on rating prediction, a commonly adopted objective function is formulated as follows:

$$Loss = \frac{1}{2|O|} \sum_{c,w \in O} (b_{cw}^{'} - b_{cw})^2 \tag{28}$$

$|O|$ represents the number of observed ratings, while b_{cw} is the actual rating of an item w by the user c.

III. EXPERIMENT

A. Experiment setup

Datasets. For this experiment, we selected two publicly available datasets, Ciao and Epinions, to simulate and evaluate the model's performance on these specific datasets. Ciao is an open-source dataset that gathers social information about its users from various social networking platforms (http://www.ciao.co.uk). Meanwhile, Epinions is a dataset that provides social information on numerous users from publicly accessible websites. It has millions of users who can comment online simultaneously (www.epinions.com). This dataset includes much information about ratings and social data. The rating scale ranges from 1 to 5. First, a random permutation of these five numbers is utilized to initialize the opinion embedding. This means that five different embedding vectors are chosen. Please consult Table II for statistical information regarding both datasets.

TABLE II
DSTASET STATISTICS

Data set	Ciao	Epinions
Number of users	7,317	18,088
Number of projects	10,4975	261, 649
Number of ratings	283,319	764,352
Density (nominal)	0.0368%	0.0161%
Social contact	111,781	355,813
Density(social relations)	0.2087%	0.1087%

Evaluation Metrics. Two commonly used metrics for evaluating recommendation algorithms are Mean Absolute Error (MAE) and Root Mean Square Error (RMSE). These metrics help to measure the accuracy of predictions and ensure the quality of recommendations. A lower value of MAE and RMSE indicates higher prediction precision and accuracy. Even slight improvements, such as a decrease in MAE, can significantly impact the overall quality and effectiveness of the recommendation algorithm.

Baseline. To guarantee the precision and dependability of the assessment outcomes of the TTYGNN model, a comparison was conducted among three different sets of recommender systems: conventional recommender systems, traditional social recommender systems, and those based on deep neural networks [10]. For each category of methods, representative classical baseline models were chosen, and their performance is subsequently explained in detail.

•PMF [11]: The probabilistic matrix decomposition algorithm inputs only a user/item rating matrix. This matrix is then processed through a Gaussian distribution to reveal the underlying characteristics of users and items.

•SoRec [12]: Social recommendation leverages user ratings and social relationships to extract valuable features through joint decomposition.

•SoReg [13]: Social network information is transformed into regularized terms that constrain matrix factorization algorithms.

•SocialMF [14]: The recommender system model incorporates trust information and propagates it through matrix decomposition.

•TrustMF [15]: The trust network is divided into two spaces: the trustor space and the trustee space. This is accomplished using matrix decomposition techniques to project users into two lower-dimensional spaces based on the directionality of trust.

•NeuMF [16]: A neural network-based matrix decomposition model was developed for recommendation ranking tasks. It was later modified to predict ratings using a squared loss function.

•DeepSoR [17]: This study proposes a model that enhances the accuracy of rating predictions by using deep neural networks to extract user features from complex social relationships and integrate them into probabilistic matrix decomposition models.

•GraphRec [18]: The model uses network embedding techniques to effectively diminish dimensionality and streamline the training process.

The PMF and NeuMF models rely solely on collaborative filtering to predict ratings and do not incorporate social network data. On the other hand, the different models fall under the social recommendation category and consider social network information.

We acquired the optimal experimental data after conducting multiple simulated parameter optimization experiments. We used a certain percentage (x%) of the training data for each dataset as the learning parameters. In contrast, the remaining data was divided into two equal parts: one for hyperparameter tuning and the other for final performance comparison. The value of x ranged from 80% to 60%. We explored the embedding dimension d experimentally, using the values [8, 16, 32, 64, 128, 256]. We also tested batch sizes and learning rates within [32, 64, 128, 512] and [0.0005, 0.001, 0.005, 0.01, 0.05, 0.1], respectively.

In deep learning, ablation studies are widely used to evaluate the impact of individual components or features in a model on its performance. Ablation experiments provide a powerful analytical tool that offers valuable guidance for model improvement. We will conduct a detailed comparative analysis, focusing on the ablation experiments of each model. This experimental approach can help us understand the contribution of each component of the model to its performance, leading to a better comprehension of how the model works and how to optimize and improve its structure.

We conducted an ablation study in this experiment to evaluate our model's performance against other models. We progressively removed or replaced specific components or features of the model to assess their impact on its overall performance. This experiment helped us understand the contribution of each part of the model to its overall performance and guided further improvements. To ensure a comprehensive evaluation, we compared our model with other representative models, including traditional machine learning models, deep learning models, and some of the latest models. These models varied in structure, number of parameters, and training methods, providing various perspectives and references.

We perform an ablation study to assess the importance of different components in our model. This involves removing key elements, such as a convolutional or recurrent layer, and training and testing the modified model using the same dataset and evaluation metrics. We then compare the performance changes before and after the modification, which helps us evaluate the significance of that component to the model's performance. We can also try replacing certain components, such as changing the activation function or optimizer, to evaluate the interactions between different

components and their impact on the model's performance. Through these experiments, we can better understand how our model works and make improvements to enhance its performance.

During an ablation study, it is crucial to ensure that we maintain consistency in controlled variables. This means that when making modifications, we should only change one component at a time while keeping others unchanged. This approach allows for a more accurate assessment of the specific impact of each component on the model's performance without interference from other factors.

We conducted an ablation study on our model and compared it to other models, evaluating the influence of different components on the model's performance. After analyzing the data, we found that the error of our model is lower than that of the other models.

TABLE III
PERFORMANCE COMPARISON OF DIFFERENT RECOMMENDER SYSTEMS

Training Ciao(60%）	MAE	RMSE	Training Ciao(80%）	MAE	RMSE
PMF	0.9520	1.1967	PMF	0.9021	1.1238
SoRec	0.8489	1.0738	SoRec	0.8410	1.0652
SoReg	0.8987	1.0947	SoReg	0.8611	1.0848
SocialMF	0.8353	1.0592	SocialMF	0.8270	1.0501
TrustMF	0.7681	1.0543	TrustMF	0.7690	1.0479
NeuMF	0.8251	1.0824	NeuMF	0.8062	1.0617
DeepSoR	0.7813	1.0437	DeepSoR	0.7739	1.0316
GraphRec	0.7540	1.0093	GraphRec	0.7387	0.9794
TTYGNN	**0.7510**	**1.0032**	**TTYGNN**	**0.7247**	**0.9613**

TABLE IV
PERFORMANCE COMPARISON OF DIFFERENT RECOMMENDER SYSTEMS

Training Epinions (60%）	MAE	RMSE	Training Epinions (80%）	MAE	RMSE
PMF	1.0211	1.2739	PMF	0.9952	1.2128
SoRec	0.9086	1.1563	SoRec	0.8961	1.1437
SoReg	0.9412	1.1936	SoReg	0.9119	1.1703
SocialMF	0.8965	1.1410	SocialMF	0.8837	1.1328
TrustMF	0.8550	1.1505	TrustMF	0.8410	1.1395
NeuMF	0.9097	1.1645	NeuMF	0.9072	1.1476
DeepSoR	0.8520	1.1135	DeepSoR	0.8383	1.0972
GraphRec	0.8441	1.0878	GraphRec	0.8168	1.0631
TTYGNN	**0.8308**	**1.0701**	**TTYGNN**	**0.8103**	**1.0537**

B. Recommend a performance comparison of systems

The first step is to compare the performance of all proposed methods. Tables 3 and 4 comprehensively summarize the prediction errors (RMSE and MAE) of the other models on the Ciao and Epinions datasets. The key findings are as follows:

•Matrix decomposition-based methods such as SoRec, SoReg, SocialMF, and TrustMF have been found to outperform PMF in recommender systems. They use rating data and social network information to improve recommendations' accuracy. In particular, incorporating social network information can enhance the performance of recommender systems.

•Neural Collaborative Filtering (NeuMF) outperforms Probabilistic Matrix Factorization (PMF), which relies solely on rating information. It is worth noting that NeuMF employs a neural network architecture, highlighting the superiority of neural network models in recommender systems.

•DeepSoR and GraphRec outperform SoRec, SoReg, SocialMF, and TrustMF by leveraging ratings and social network data. These models use neural network architectures, highlighting the potential of neural networks in recommender systems.

•In initial tests, TTYGNN demonstrated satisfactory performance. This suggests that Graph Neural Networks (GNNs) possess immense potential in learning graph data representation, enabling effective integration of node information and topology.

•The paper thoroughly examines the contributions of each model component and proposes suitable frameworks for integrating ratings data with social network information. TTYGNN outperforms GraphRec and DeepSoR in this regard.

In summary, the comparative results demonstrate that:

1) Improving the accuracy of recommendations by incorporating social network data.

2) Integrating neural network models can significantly enhance the efficiency of recommender systems.

3) The proposed framework outperforms several commonly used baselines.

C. Model analysis

This section thoroughly investigates the impact of model components and hyperparameters on the results.

The text below discusses the examination of the impact of social networks and user opinions. The proposed framework presented in the concluding chapter demonstrates its efficacy. The framework introduces model components integrating comprehensive social network information and incorporating user opinions and preferences. To gain a deeper understanding of how TTYGNN operates, the data is summarized similarly to the GraphRec model and compared against two variants: TTYGNN-SN and TTYGNN-Opinion. The findings of this comparison are presented in Figure 6. The subsequent sections define these two variants:

•TTYGNN-SN: In TTYGNN, user social network information is omitted. This variant uses only the item space latent factor z_c^C to represent the user's latent factor and does not consider the user's latent factor z_c^S in the social space.

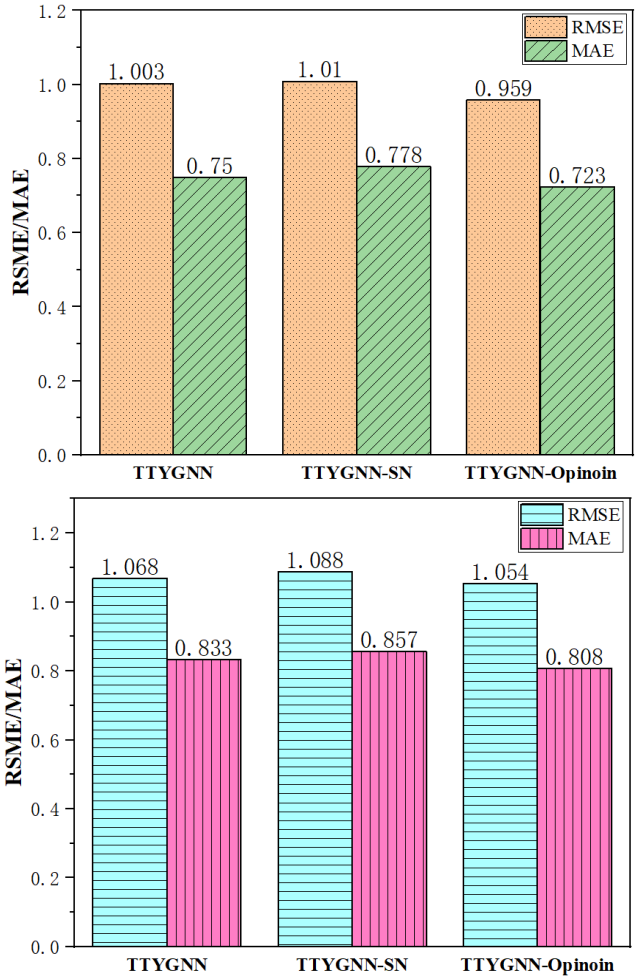

Fig.6 Impact of social networks and user Information on the ciao and epinions datasets.

•TTYGNN-Opinion: During the Study w_{ei} , this variant ignores the user's opinion of the project. The text talks about a specific approach that only considers the potential of users and projects within a project space without considering the user's perspective on the interaction between the user and the project. The third figure shows how TTYGNN and its two variants performed on the Ciao and Epinions datasets. By analyzing the results, we can conclude the following things:

•An investigation is being conducted on how social network information affects recommender systems. The findings suggest that TTYGNN-SN is less effective than TTYGNN, highlighting social network information's significance in enhancing recommendation learning and accuracy.

•During the interaction process, opinions play a crucial role. If there is a lack of opinion information, it can significantly reduce the accuracy of rating predictions. For instance, on the Ciao and Epinions datasets, the average relative errors calculated using RMSE are 3.50% and 2.64%, respectively. On the other hand, those determined using MAE are 5.84% and 5.02% respectively. This highlights the importance of user reviews in uncovering a user's or program's underlying factors, enhancing recommender systems' effectiveness.

The article examines the influence of attention and how it can affect the performance of a model. To improve the model's performance, the CBAM attention mechanism is incorporated and compared with an older version that does not have this feature. The findings of this comparison are presented in Figure 7.

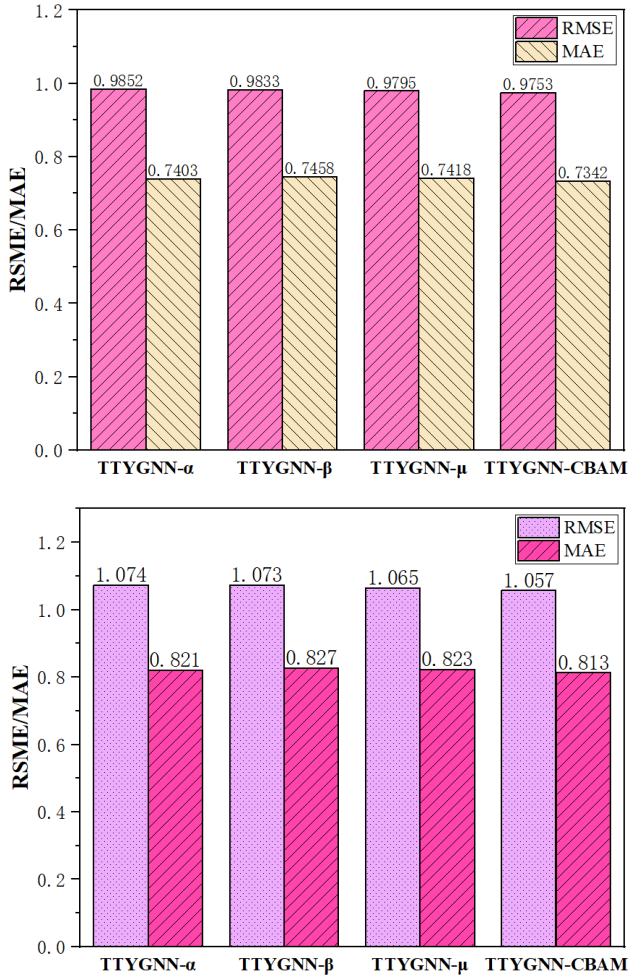

Fig.7 Impact of cbam attention mechanisms on the ciao and epinions datasets.

Four TTYGNN variants were compared: TTYGNN-α, TTYGNN-β, TTYGNN-μ, and TTYGNN-CBAM.

TTYGNN- α :Represented as item attention. This variant is not considered in the concept-aware interactive representation of aggregated items. The mean aggregation function combines items and represents user latent factors within the item space.

TTYGNN- β :Denoted as social attention. In this variant, TTYGNN neglects the social attention α during user neighbor aggregation. The variant uses a mean-based aggregation function to capture social aggregation of user latent factors in social space.

TTYGNN- μ :Denoted as user attention. When users interact with each other, the perceived TTYGNN of intersections within the overall opinion μ is ignored. This variant uses a mean-based aggregation function to model the underlying factors of the project for user aggregation.

TTYGNN-CBAM: The CBAM module enhances intermediate feature maps with channel and spatial attention mechanisms to optimize recommender systems.

•Different users of interactive items, such as those with varying purchase histories, contribute differently to user latent factors within the item space. Similarly, the importance

of understanding item latent factors varies among interactive users, such as buyers. The TTYGNN model was evaluated using two different attentional mechanisms. The experimental results indicate that TTYGNN-α, TTYGNN-β, and TTYGNN-μ performed worse than TTYGNN-CBAM. These findings suggest that the CBAM attentional mechanism outperforms the no-addition-prior attentional mechanism.

•In summary, integrating the CBAM attention mechanism significantly improves recommendation performance for the TTYGNN model.

According to the results displayed in Figure 8, it is evident that KFCM clustering offers significant advantages in accurately identifying data clusters and improving cluster stability compared to the traditional FCM clustering method. When the PSO algorithm is integrated with KFCM clustering, it results in an enhanced PSO_KFCM clustering model. This model exhibits several notable advantages over the original KFCM clustering method. By combining the PSO algorithm, we can optimize the parameter selection of KFCM clustering, thereby improving the clustering effect.

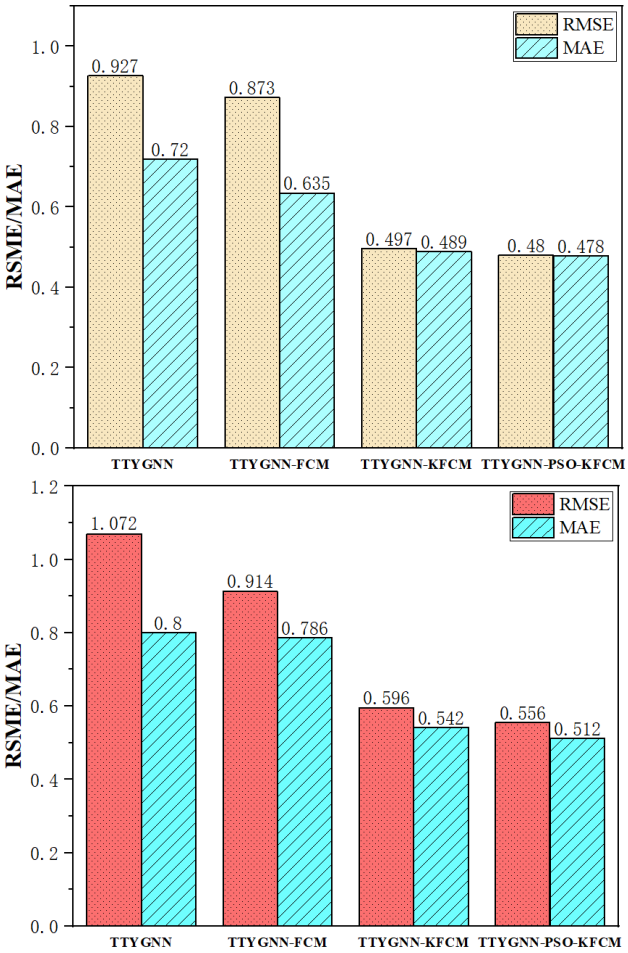

Fig.8 PSO_KFCM clustering on the ciao and epinions datasets.

As a heuristic optimization algorithm, the PSO algorithm can effectively explore the parameter space and find the optimal parameter combinations, thus obtaining more accurate and stable clustering results. Compared with the traditional KFCM clustering method, the PSO_KFCM clustering model can better adapt to different datasets and improve the accuracy and reliability of clustering. In addition, the PSO algorithm helps to avoid the local optimal solution problem in KFCM clustering. In KFCM clustering, the choice of initial parameters and the non-convexity of the objective function may lead to local optimal solutions, while the global optimal solution cannot be realized. In contrast, the PSO algorithm avoids the problem of local optimal solutions by simulating the foraging behavior of bird flocks and effectively searching globally in the solution space. This means that the PSO_KFCM clustering model can better explore the solution space, discover better clustering results, and improve the sta bility and accuracy of clustering.

By combining the PSO algorithm with KFCM clustering, we arrive at a new enhanced PSO_KFCM clustering model. Compared with the original KFCM clustering method, this model has the advantages of optimizing parameter selection and avoiding local optimal solutions, thus improving the stability and accuracy of clustering results. This innovative model combining heuristic optimization algorithm and clustering method provides a new idea and method for solving complex data clustering problems.

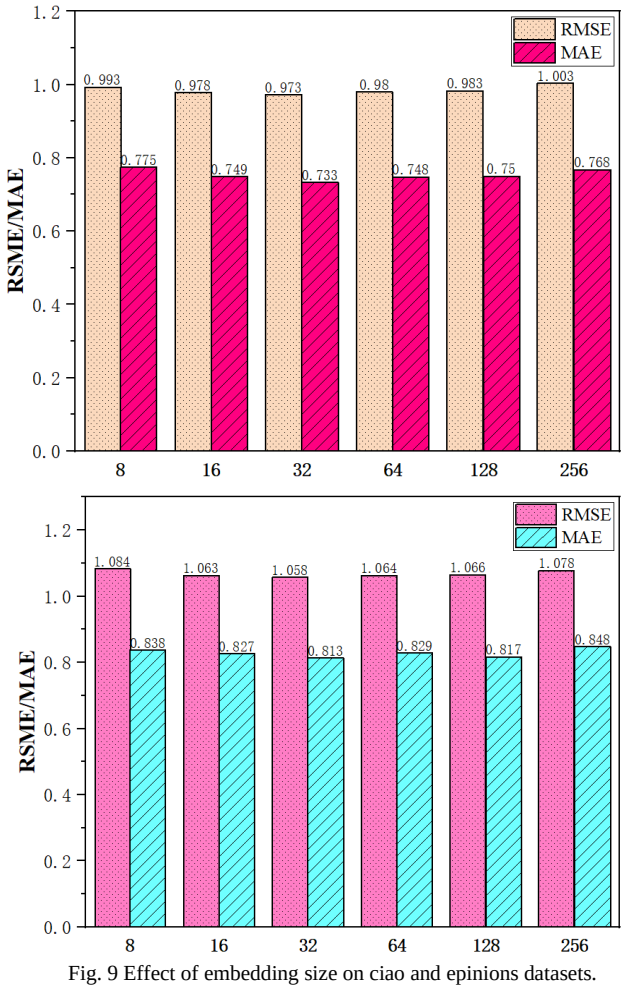

Fig. 9 Effect of embedding size on ciao and epinions datasets.

The figures presented in Figure 9 illustrate the impact of embedding length on the Ciao and Epinions datasets. The experimental findings indicate that enhancing the embedding length leads to a significant improvement in performance when the embedding size falls between 8 and 32. However, when the embedding size is increased to 256, the performance of TTYGNN begins to decline. This indicates that while larger embedding sizes may improve

representativeness, excessively long embedding lengths may increase the complexity of the model. Therefore, finding an optimal embedding length is crucial to achieving maximum experimental performance.

As shown in Figure 9, the performance of the Ciao and Epinions datasets increases with an increase in the embedding length when the embedding size is between 8 and 32. This implies that elevating the embedding length within this range can extract more feature information, thus enhancing the model's performance. However, when the embedding size is further increased to 256, the performance of the TTYGNN model begins to decrease. This suggests that an embedding size that is too large may make the model too complex, negatively affecting its performance. Therefore, determining an appropriate embedding length is essential to achieve the best experimental performance. The embedding length should provide enough feature information while avoiding making the model too complex.

In summary, the figures in Fig. 9 exhibit the impact of embedding length on the Ciao and Epinions datasets. Increasing the embedding length can significantly improve the performance within a specific range. However, an embedding size that is too large may make the model too complex, which negatively affects its performance. Therefore, determining an optimal embedding length is crucial to achieving maximum experimental performance.

IV. Related work

This section briefly overviews our contributions to social recommendation systems, deep neural network techniques, and advanced graph neural networks.

Recently, recommender systems have shifted from traditional algorithms to deep learning-based algorithms. This paper concentrates explicitly on social recommender systems prioritizing frequent interactions among individuals. These systems aim to assist users in filtering out irrelevant information by collecting and analyzing data. As a result, social relationships play a critical role in enhancing the effectiveness of recommender systems.

In the current age of rapid internet expansion, there has been significant progress in deep neural network technology for image data processing. Graph Neural Networks (GNN) are a collection of network architectures that exhibit exceptional performance in image classification tasks. The core concept behind GNN is to integrate feature information from a self-defined graph neighborhood using a neural network. This method effectively learns representations by transferring node information and merging graph topologies. The emergence of deep neural network models has profoundly impacted various learning domains, such as image recognition, natural language processing, and speech recognition. Researchers have also proposed neural collaborative filtering frameworks, such as NeuMF, designed to capture the nonlinear interactions between users and items. This framework can improve the ability of recommender systems to understand user interests and needs, leading to more personalized recommendations.

Integrating deep neural networks with social recommender systems has recently gained popularity. One example is the NSCR, which uses the NeuMF model to suggest items from an information domain to potential users within a social network. This method introduces a neuro-social collaborative ranking framework. The NeuroMF model uses a neural network to represent and align user and item features, resulting in more precise recommendations. Unlike traditional collaborative filtering algorithms, NeuroMF considers the similarities and differences among users and the degree of association between users and items.

SMRMNRL uses a ranking approach that adopts the perspective of learning from a multimodal heterogeneous network to illustrate the evolution of socially conscious movie recommendations within social media. They use recurrent neural networks and convolutional neural networks to understand elements such as the portrayal in movie text descriptions. A stochastic perturbation-based approach was employed to integrate multimodal neural networks [19,20]. Through these initiatives, they have successfully addressed the challenge of cross-domain social recommendation, which is notably different from conventional social recommender systems.

The paper closely relates to two neural network [21] models: DeepSoR and GraphRec. The DeepSoR team has proposed a unique method that combines probabilistic matrix decomposition with a neural network model that considers users' social relationships. This method employs pre-trained node embedding techniques to represent users and c-nearest-neighbor algorithms to create connections between the user's embedded features and the neural network [22-24]. On the other hand, the GraphRec model introduces the concept of "relational embeddings," which capture the vector representations of node-edge relationships within the graph. By acquiring these embeddings, the model can better understand the graph's structure, improving its performance in classifying graph relations.

GraphRec is an advanced recommendation system that uses attention mechanisms to effectively combine user, social, and network information for accurate and personalized recommendations. Since every user has unique interests and preferences, GraphRec employs an attention mechanism [25-26] to integrate user, social, and network information, enabling personalized recommendations to users. This fosters social network interactions and exchanges, leading to a more engaging and personalized social experience. The emergence of this technology presents new opportunities and challenges for the advancement of social networks.

Graph Neural Networks (GNNs) have shown impressive ability in handling graph-structured data, particularly in recommender system tasks where the complex interaction between users and items is considered a prime example of graph data. To tackle the recommendation challenge, researchers have developed GNN-based models such as SRMGCNN [25-26]. This model utilizes Graph Neural Network (GNN) techniques to derive graph embedding representations from the intricate relationships between users and items. The graph auto-coding framework introduced by GCMC generates latent features for users and items by disseminating unique messages throughout the user-item graph. The framework leverages node and edge representations to extract valuable features through message propagation. It can play a crucial role in social network analysis and personalized recommendations. The stochastic wandering graph neural network of PinSage is a promising approach for learning node embedding representations in web-scale graphs. This method holds significant potential

within the domain of graph neural networks. It is expected to provide an effective solution for various graph data analysis and modeling tasks.

V. Conclusions and Future Work

In the internet era, exchanging information and interacting with each other has become increasingly important. For recommender systems that focus on social interactions, accurately understanding user-user interactions is crucial for providing personalized recommendations. However, existing recommender system models often have limitations that can affect the accuracy and efficiency of recommendations. To address these issues, we propose a new model called TTYGNN, which is based on graph neural networks. TTYGNN combines the PSO_KFCM clustering algorithm and the CBAM attention mechanism. Graph neural networks are powerful tools to process complex network-structured data efficiently, while PSO_KFCM is an improved clustering algorithm that combines historical recommendation data points with similar features. CBAM is an attention mechanism that identifies essential features in an image while suppressing irrelevant noise information. By integrating CBAM, TTYGNN can understand and capture the dynamics of user-item interactions more comprehensively, providing more accurate and personalized recommendations. TTYGNN also employs the Particle Swarm Optimization (PSO) algorithm to optimize the KFCM clustering process. PSO is an efficient global optimization algorithm that effectively finds the optimal solution and improves the accuracy and efficiency of clustering. We conducted experiments on two real datasets, and the results show that TTYGNN performs significantly better than other similar recommender system models. This demonstrates the effectiveness of our model in capturing user-item interaction dynamics and improving recommendation accuracy and efficiency. In our future research, we plan to explore and study the factors affecting user-to-user dynamics to understand better and predict users' social behaviors, providing more accurate and personalized recommendation services.

References

[1] H. Ma, H. Yang, M.R. Lyu, and I. King,"Sorec:social recommendation using probabilistic matrix factorization." *In Proceedings of the 17th ACM conference on Information and knowledge management*, pp. 931-940, 2008.

[2] B. Cai, X. Li, W. Kong, J Yuan, and S. Yu, " A reliable and lightweight trust inference model for service recommendation in SIoT," *IEEE Internet of Things Journal*, vol.9, no. 13, pp.10988-11003,2021.

[3] S. Woo, J. Park, J. Y. Lee, and I. S. Kweon, "Cbam: Convolutional block attention module," arXiv:1807.06521,2018.

[4] F. Yu, Y. Zhu, Q. Liu, S. Wu, L. Wang, and T. Tan, "TAGNN: Target attentive graph neural networks for session-based recommendation," *In Proceedings of the 43rd international ACM SIGIR conference on research and development in information retrieval*, pp. 1921-1924, 2020.

[5] A. M. Mehar, K. Matawie, and A. Maeder, "Determining an optimal value of K in K-means clustering," *In 2013 IEEE international conference on bioinformatics and biomedicine*, pp. 51-55, 2013.

[6] R. Eberhart, and J. Kennedy, "A new optimizer using particle swarm theory," *In MHS'95. Proceedings of the sixth international symposium on micro machine and human science*, pp. 39-43, 1995.

[7] X. Zhu, and Z. Han, "Research on LS-SVM wind speed prediction method based on PSO," *Proceedings of the CSEE*, vol.36, no. 23, pp.6337-6342,2016.

[8] S. Das, and H. K. Baruah, "Towards finding a new kernelized fuzzy C-means clustering algorithm," *J. Process Manag*, vol.2, no. 2, pp. 54-66,2014.

[9] G. Geng, Y. He, J. Zhang, T. Qin, and B. Yang, "Short-term prediction based on VMD and PSO optimized deep belief network," *Energies*, vol. 16, no. 12, pp. 4616,2023.

[10] T. N. Kipf, and M. Welling, "Semi-supervised classification with Graph Convolutional Networks,"arXiv:1609.02907,2016.

[11] S. Minaee, Y. Boykov, F. Porikli, A. Plaza, N. Kehtarnavaz, and D. Terzopoulos, "Image segmentation using deep learning: A survey," IEEE transactions on pattern analysis and machine intelligence, pp. 3523-3542, 2021.

[12] H. Ma, H.Yang, M.R. Lyu, and I.King, "Sorec: social recommendation using probabilistic matrix factorization," In Proceedings of the 17th ACM conference on Information and knowledge management, pp. 931-940, 2008.

[13] H. Ma, D. Zhou, C. Liu, M.R. Lyu, and I. King, "Recommender systems with social regularization," In Proceedings of the fourth ACM international conference on Web search and data mining, pp. 287-296, 2011.

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

[15] B. Yang, Y. Lei, J. Liu, and W. Li, "Social collaborative filtering by trust," IEEE transactions on pattern analysis and machine intelligence, vol. 39, no.8, pp. 1633-1647, 2016.

[16] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T. Chua, "Neural Collaborative Filtering," In Proceedings of the 26th international conference on world wide web, pp.173-182, 2017.

[17] W. Fan, Q. Li, and M. Cheng, "Deep modeling of social relations for recommendation, "In Proceedings of the AAAI Conference on Artificial Intelligence, vol. 32, no. 1, pp. 8075–8076, 2018.

[18] W. Fan, Y. Ma, Q. Li, Y. He, E. Zhao, J. Tang, and D. Yin, "Graph neural networks for social recommendation," In The world wide web conference, pp. 417-426, 2019.

[19] R. Kamani, V. Kumar, and V.R. Kagita, "Multimodal deep learning for cross-domain recommendation systems," arXiv:2306.13887 ,2023.

[20] M. S. Halvagal, and F. Zenke, "The combination of Hebbian and predictive plasticity learns invariant object representations in deep sensory networks," *Nature Neuroscience* vol.26, no.11, pp.1906-1915,2023.

[21] Q. Cao, H. Shen, J. Gao, B. Wei, and X. Cheng, "Popularity prediction on social platforms with coupled graph neural networks," *In Proceedings of the 13th international conference on web search and data mining*, pp. 70-78, 2020.

[22] K. He, X. Zhang, S. Ren, and J. Sun, "Identity mappings in deep residual networks," *In Computer Vision–ECCV 2016: 14th European Conference*, vol.4, no.14, pp. 630-645,2016.

[23] G. Huang, Y. Sun, Z. Liu, D. Sedra, and K.Q. Weinberger, "Deep networks with stochastic depth," *In Computer Vision–ECCV 2016: 14th European Conference*, vol.4, no.14, pp.646-661,2016.

[24] T. T. YANG, S. Y. ZHOU, and A. J. XU, "Rapid Image Detection of Tree Trunks Using a Convolutional Neural Network and Transfer Learning," *IAENG International Journal of Computer Science*, vol. 48, no.2, pp257-265, 2021.

[25] X. P. Chen, and Y. Xu, "A Multi-Dimensional Attention Feature Fusion Method for Pedestrian Re-identification," *Engineering Letters*, vol. 31, no.4, pp.1365-1373, 2023.

[26] X. J. Zhang, G. J. Yu, J. Y. Shang, and B. Q. Zhang, "Short-term Traffic Flow Prediction With Residual Graph Attention Network," *Engineering Letters*, vol. 30, no.4, pp.1230-1236, 2022.