
Article:
Kang, Y., Pu, B., Kou, Y. et al. (2024) A deep graph network with multiple similarity for user clustering in human–computer interaction. ACM Transactions on Multimedia Computing, Communications and Applications, 20 (2). pp. 1-20. ISSN: 1551-6857

https://doi.org/10.1145/3549954

A Deep Graph Network with Multiple Similarity for User Clustering in Human–Computer Interaction

YAN KANG*, Yunnan University, China
BIN PU*, Hunan University, China
YONGQI KOU, Yunnan University, China
YUN YANG†, Yunnan University, China
JIANGUO CHEN, Sun Yat-Sen University, China
KHAN MUHAMMAD, Sungkyunkwan University, Republic of Korea
PO YANG, Sheffield University, UK
LIDA XU, Old Dominion University, US
MOHAMMAD HIJJI, University of Tabuk, Saudi Arabia

User counterparts, such as user attributes in social networks or user interests, are the keys to more natural Human–Computer Interaction (HCI). In addition, users' attributes and social structures help us understand the complex interactions in HCI. Most previous studies have been based on supervised learning to improve the performance of HCI. However, in the real world, owing to signal malfunctions in user devices, large amounts of abnormal information, unlabeled data, and unsupervised approaches (e.g., the clustering method) based on mining user attributes are particularly crucial. This paper focuses on improving the clustering performance of users' attributes in HCI and proposes a deep graph embedding network with feature and structure similarity (called DGENFS) to cluster users' attributes in HCI applications based on feature and structure similarity. The DGENFS model consists of a Feature Graph Autoencoder (FGA) module, a Structure Graph Attention Network (SGAT) module, and a Dual Self-supervision (DSS) module. First, we design an attributed graph clustering method to divide users into clusters by making full use of their attributes. To take full advantage of the information of human feature space, a k-neighbor graph is generated as a feature graph based on the similarity between human features. Then, the FGA and SGAT modules are utilized to extract the representations of human features and topological space, respectively. Next, an attention mechanism is further developed to learn the importance weights of different representations to effectively integrate human features and social structures. Finally, to learn cluster-friendly features, the DSS module unifies and integrates the features learned from the FGA and SGAT modules. DSS explores the high-confidence cluster assignment as a soft label to guide the optimization of the entire network. Extensive experiments are conducted on five

*Both authors contributed equally to this research.
†Corresponding author.

Authors' addresses: Yan Kang, kangyan@ynu.edu.cn, Yunnan University, East Outer Ring Road, Kunming, Yunnan, China, 650091; Bin Pu, pubin@hnu.edu.cn, Hunan University, Lushan Road (S), Changsha, Hunan, China, 410082; Yongqi Kou, yongqikou@163.com, Yunnan University, East Outer Ring Road, Kunming, China; Yun Yang, yunyang@ynu.edu.cn, Yunnan University, Kunming, East Outer Ring Road, China; Jianguo Chen, chenjg33@mail.sysu.edu.cn, Sun Yat-Sen University, China; Khan Muhammad, khan.muhammad@ieee.org, Sungkyunkwan University, Visual Analytics for Knowledge Laboratory (VIS2KNOW Lab), Department of Applied Artificial Intelligence, School of Convergence, College of Computing and Informatics, Seoul 03063, Republic of Korea; Po Yang, po.yang@sheffield.ac.uk, Sheffield University, UK; Lida Xu, LXu@odu.edu, Old Dominion University, US; Mohammad Hijji, m.hijji@ut.edu.sa, University of Tabuk, Tabuk 47711, Saudi Arabia.

0004-5411/2018/8-ART111 $15.00
https://doi.org/10.1145/1122445.1122456

real-world data sets on user attribute clustering. The experimental results demonstrate that the proposed DGENFS model achieves the most advanced performance compared with nine competitive baselines.

CCS Concepts: • **Human-centered computing** → **User models**; **HCI design and evaluation methods**.

Additional Key Words and Phrases: Attributed graph clustering, cluster-friendly features, deep graph embedding, self-supervision module.

ACM Reference Format:
Yan Kang, Bin Pu, Yongqi Kou, Yun Yang, Jianguo Chen, Khan Muhammad, Po Yang, Lida Xu, and Mohammad Hijji. 2018. A Deep Graph Network with Multiple Similarity for User Clustering in Human–Computer Interaction. *J. ACM* 37, 4, Article 111 (August 2018), 20 pages. https://doi.org/10.1145/1122445.1122456

1 INTRODUCTION

In recent years, Human–Computer Interaction (HCI) has gradually changed from a computer-centric approach to a more user-centric approach [1, 2]. In HCI applications, the degree of activity between users and machines is considered from three levels: physical [3], cognitive [4], and emotional [5]. Commercial success has made user-friendly input methods, portable devices, and multi-sensor availability the new standards for personal computing. For example, wearable sports devices are being used by more users. Academic research has begun to focus on the "human aspects" of such technologies to understand their impacts on athlete performance and develop more effective ways of interaction [6, 7]. The training plan of sports can be made by aggregating the training information between the athletes [8]. Another example is smart homes, where users expect not only to be able to easily control their home but also for their home to adapt itself to their needs, actions, and preferences over time. Whenever the home fails to perform as expected, there will be conflicts, resulting in user dissatisfaction [9–12]. So focusing on user attributes (e.g., behavior) tends to lead to better HCI [13, 14].

Since 2010, deep learning (DL) methods have been applied to various HCI applications and achieved improvements in user attributes, as well as won competitions at EmotiW [15] and improved iris detection [16] and AVEC [17]. Today, most of the methods of HCI user attribute mining are based on supervised learning. For example, Pimenta *et al.* [18] proposed the use of neural networks to continuously classify user fatigue and support effective and efficient fatigue management measures. However, due to signal malfunctions in user devices, large amounts of abnormal information, and unlabeled data, unsupervised approaches based on mining user attributes are particularly crucial. Yet, related research is currently ignored. User attribute clustering methods can uncover hidden relationships between users and their interests to better interact with machines and devices.

A user's attributes can reflect its relationships with other users and its habits of interacting with the machine. A common method is to construct an attribute graph based on these connections in a big social network and then perform information mining. Graph clustering methods have great progress in analyzing these complex networks; however, graph clustering methods focus on graph structure and cannot effectively utilize user attributes. Attributed graph clustering has gradually become a popular direction. The attributed graph clustering method needs to integrate the two dimensions of topological relationships and node features. It also needs to balance the influence of these two aspects in the clustering process. Thus, attributed graph clustering can be used as an unsupervised tool in HCI to mine user preferences by analyzing user features and structural relationships. Social networks provide a lens for understanding the user's interactive behavior, and better support HCI to meet the user's needs. Although the motivations of users vary by domain, it is worth noting that users from the same social network will share similar motivational and behavioral patterns. Since accurately clustering user information is a useful and promising technology in HCI, we propose a novel model that takes advantage of users' complex connections

in social networks. We perform attributed graph clustering and group a massive number of users into multiple preference-aware social clusters by using their attributes and social connections.

The attributed graph [19] plays an important role in detecting communities [20] and analyzing these networks [21]. However, the integration of node features and topological structure information remains an unsolved problem. SAE [22], DEC [23], and IDEC [24] execute graph clustering based on node attributes. In particular, autoencoders are usually used to obtain low-latitude representations of node attributes and then subsequently group these nodes through traditional clustering methods. However, this type of method only relies on node attributes and does not consider their connectivity, such as the graph structure. SDNE [25] and LINE [26] explore the connectivity of nodes by manipulating the adjacency matrix of the attribute graphs to group graph nodes with similar structures. Similar to the methods based on node attributes, they perform graph clustering from one aspect of the attributed graph (i.e., node attributes and graph structure). However, neither of the above two methods effectively combines the node characteristics and topological structure.

With the continuous development of deep learning, relatively significant work has been done on graph neural networks (GNNs) [27–29]. Using the structural information within the sample's k-hop neighbors from the spectral or spatial [30] domain, each sample of the graphs recursively aggregates the sample features of its k-hop neighbors. Recently, graph convolutional networks (GCNs) [31] have attracted much attention. GCN-based generative models have been widely used for attributed graph clustering, where GCN updates the graph embedding by combining the features of adjacent nodes. These models have demonstrated strong performance in multiple attributed graph clustering tasks, such as GAE, VGAE [32], and DAEGC [33].

Although GCN has been found to be practical in the integration of structural and feature information, the correlation between features, topological information, and, graph clustering tasks, in reality, is usually complex and agnostic. In addition, most of these methods utilize the original graph structure, and the embeddings obtained cannot well reflect the similarity of features between nodes. In this way, the learned graph embedding results contain less information related to node features. Moreover, the clustering result relies on the loss of graph structure reconstruction, rather than cluster-driven loss. For example, DAEGC attempts to introduce different clustering losses to guide embedding learning to reduce the effect, but the effect may not be obvious. Through the analysis of the existing attributed graph clustering methods, we posit that there are still shortcomings that cannot effectively use the topology and node feature information and learn clustering-friendly information.

To address the above problems and improve the performance of user attribute clustering, we propose a deep graph embedding network model. Feature similarity and topology similarity are dominated by clustering-driven loss. The main idea of our work is to learn specific representations based on node features and topology and combine them according to a defined policy. The underlying principle is that the inter-feature and topology-inferred similarities are complementary. By combining them, we can obtain a more complete and powerful representation of attributed graph clustering tasks. To make full use of the information in the feature space, we first use the k-nearest neighbor graph generated from the inter-feature similarity of nodes. We propagate the node features in the feature and topological spaces and use the FGA module and SGAT module to extract their respective representations. Considering complementarity between the spaces, the importance of different representations is learned layer by layer through an attention mechanism. In addition, we combine them according to their importance to obtain graph embeddings that contain deeper information. Finally, to learn cluster-friendly features, the DSS module unifies the feature graph self-encoder and structure graph attention network module. The module uses a high-confidence clustering assignment as a soft label to guide the optimization of the entire framework. The main contributions of this paper are summarized as follows:

- We incorporate human social factors (e.g., society and citations) to simulate human attention and preference in HCI. To help users operate accurately, we model people as nodes through attributed graph clustering, model people's connectivity as a graph structure, and obtain data representation of user attributes and connection relationships. Because the size of social graph data has increased dramatically, attributed graph clustering is utilized to classify users into clusters by leveraging node attributes and graph structures. Therefore, related nodes are assigned to the same cluster, and the difference between clusters is maximized.
- We propose a novel attributed graph clustering framework (deep graph embedding network with feature and structure similarity, or DGENFS), which provides an effective deep framework to integrate feature and topological space information. The framework jointly optimizes graph embedding learning and graph clustering, making them complementary.
- We design an FGA module and an SGAT module to capture the similarity between features and the similarity of the topological structure, respectively. The features and similarities are used to effectively extract feature space or structure space information. Combined with the attention mechanism, different information can be adequately fused.
- Experiments on a series of attributed graph clustering tasks show that our model significantly outperforms the state-of-the-art graph clustering methods in terms of accuracy and performance.

2 RELATED WORK

Many approaches focus on user attribute analysis in the field of HCI. However, only a few researchers have explored whether the clustering of user attributes in HCI can promote the efficiency and accuracy of HCI. In this section, we review the existing attribute clustering methods, especially in the field of HCI.

2.1 Clustering Methods Based on Graph Attributes and Structure

The clustering method based on node attributes extracts the depth features through the encoder and then uses the decoder to reconstruct the original features. In [23], deep embedding clustering (DEC) was proposed to use an auto-encoder to pre-train the feature data and then remove the decoder. The clustering cohesion is fine-tuned by a defined Kullback–Leibler divergence clustering loss, which destroys the feature space and leads to unrepresentative features. In [24], Guo *et al.* proposed an improved deep embedding clustering (IDEC) method that recombines the decoder and optimizes reconstruction error and clustering loss. The structured auto-encoder [34] uses a deep subspace clustering method with a self-expression layer between the encoder and decoder. The algorithm can simulate the self-expression characteristics in subspace clustering to obtain a more representative representation.

Multiple studies have also focused on clustering methods based on the adjacency matrix of attributed graphs [25, 35–38]. For example, matrix-decomposition-based methods decompose node adjacency matrices into node embeddings. GraRep captures different k-order local relationship information from the graph by manipulating different global transfer matrices defined on it. Moreover, random wander-based methods learn node embeddings by maximizing the probability of each node's neighborhood [26]. For example, DeepWalk uses DFS random walk to sample the nodes in the graph and uses word2vec to learn the vector representations of the nodes in the sampled sequence. Unlike DeepWalk, LINE is another method based on the assumption of neighborhood similarity, which uses DFS to construct neighborhoods [26]. In addition, LINE can be applied to both weighted graphs and unweighted graphs, while DeepWalk can only be used for unweighted graphs. Autoencoder-based approaches try to capture the nonlinear embedding of the adjacency matrix [37, 38]. For example, Wang *et al.* considered that Line step-wise optimization has difficulty

in capturing highly nonlinear network structures [25]. They proposed to use an autoencoder-based network to optimize both first- and second-order similarities to capture such structures.

2.2 Clustering Methods Based on GCN

Representative approaches mainly use GCN [31] to integrate node characteristics and topologies. In particular, GAE and VGAE [32] use a two-layer superposed GCN to learn node representations and then use the autoencoder and variational autoencoder to reconstruct node adjacency matrices. To learn a better node representation, MGAE [39] learns the node representation through a three-layer GCN, and then applies a marginalized denoising autoencoder to reconstruct the node features. DAEGC [33] uses a graph attention network to capture the importance of neighboring nodes to the target nodes. Then, the model combines the graph adjacency structure and node features into graph embeddings, and finally generates soft labels from the graph embeddings to supervise self-training graph clustering. EGAE-JOCAS [40] introduces clustering models into the GCN, which combines relaxed k-means and spectral clustering, and is applicable for learning embedding. SDCN [41] uses a GCN and a deep neural network with feature reconstruction loss functions to separately learn two node representations. Feature representations of the nodes are passed to the GCN layer through a designed transfer factor to integrate their information. SENet [42] integrates both structure and feature information into a kernel matrix via a higher-order graph convolution and improves the graph structure by leveraging the information of shared neighbors.

2.3 Applications of User Attribute Clustering in HCI

In HCI applications, users with specific backgrounds and different usage habits have different needs [43–47]. The relevant attributes of users can be clustered to improve the efficiency of HCI. In [48], Nguyen *et al.* pointed out that proper user understanding and recognition of their hobbies, habits, psychology, and feelings will help designers convey correct ideas and improve the efficiency of the HCI process. In addition, in [49], Shen *et al.* considered the characteristics of a special group of elderly people and obtained the interaction mode. Then, they designed principles to meet their requirements to enable older users to obtain higher efficiency and satisfaction while using computers. In [50], Adeyemi *et al.* found relevance and application in the human-centered graphical user interface design of recommendation systems and e-commerce services. They used cluster dichotomy to distinguish the thinking styles of individuals with different dichotomies.

In addition, Mencarini *et al.* pointed out that HCI needs to consider the special needs of athletes [7]. It was found that different types of athletes and tasks have an important impact on HCI. In [51], Miandashti *et al.* introduced an empirical method for modeling user–system interaction conflicts in smart homes. They used conflict sample scenarios collected from 163 users. Based on the clustering of these scenarios, an empirical definition of user system conflicts for intelligent homes was formed. However, prior studies have mainly focused on user attributed clustering based on the special information but ignored the feature fusion between user characteristics and spatial structure.

3 METHODOLOGY

3.1 Problem Statement and Objective

Given an attributed graph G, $G = (V, E, X)$, where $V = \{v_i\}_{i=1,2,3,...n}$ is the set of nodes, and $E = \{e_{ij}\}$ is the set of edges between nodes. The topology of graph G can be represented by the adjacency matrix A, where $A_{ij} = 1$ means that there is an edge between nodes i and j; otherwise, $A_{ij} = 0$. $X = \{x_1, x_2, x_3, ..., x_n\}$ is the set of user attribute values, where $x_i \in R^d$ is the attribute vector associated with node v_i.

Objective: Given an attributed graph G, the goal of attributed graph clustering is to divide the nodes in G into k disjointed sub-groups $\{G_1, G_2, G_3, ..., G_k\}$. Nodes in the same sub-group are usually close to each other and are more likely to have similar attribute values.

3.2 Overview of the Proposed Model

The framework of the proposed method is shown in Fig. 1. Our model consists of an FGA module, SGAT module, and DSS module. The FGA module allows the propagation of node features in the feature space to learn the representation $\mathbf{Z}_f^{(l)}$, and the feature graph is composed of $\mathbf{X}$. The SGAT module allows node features to be propagated in the topological space to learn the representation $\mathbf{Z}_t^{(l)}$, where the topological graph is the adjacency matrix $\mathbf{A}$. To effectively utilize the learned representations, $\mathbf{Z}_f^{(l)}$ and $\mathbf{Z}_t^{(l)}$ are fused through the attention mechanism to learn a more complete and powerful representation $\mathbf{Z}_{f+t}^{(l)}$ and obtain the graph embedding $\mathbf{Z}$. The DSS module generates the target distribution $\mathbf{P}$ through $\mathbf{Z}_f^{(l)}$. At the same time, it guides the update of the FGA module and SGAT module, making it possible to optimize the graph embedding and clustering module in a unified framework.

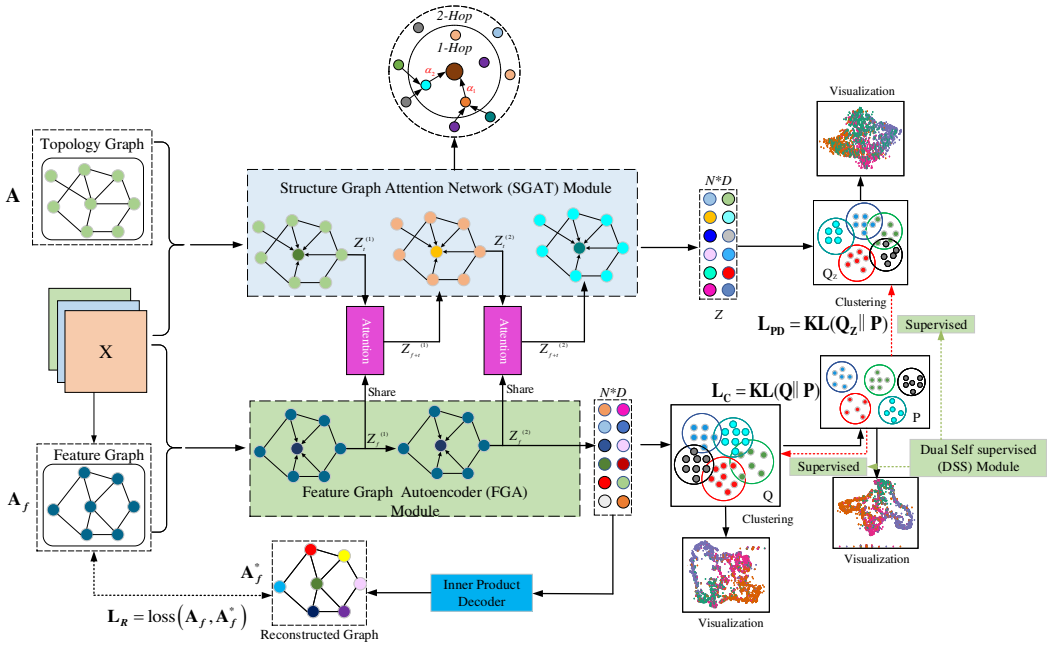


Fig. 1. The overall architecture of the proposed model. The DGENFS model consists of a Feature Graph Autoencoder (FGA) module, a Structure Graph Attention Network (SGAT) module, and a Dual Self-supervision (DSS) module. First, an attributed graph clustering method is designed to divide users into clusters, and a k-neighbor graph is generated as a feature graph based on the similarity between human features. Then, the FGA and SGAT modules extract the representations of human features and topological space, respectively. In addition, an attention mechanism is further adapted to learn the importance weights of different representations and integrate human features and social structures. Finally, the DSS module unifies and integrates the learned features to learn cluster-friendly features and explore the high-confidence cluster assignment.

3.3 Feature Graph Construction

To capture the underlying structure of the nodes in the feature space, we construct a k-nearest neighbor graph $G_f = (\mathbf{A}_f, \mathbf{X})$ based on the node feature matrix $\mathbf{X}$, where $\mathbf{A}_f$ is the adjacency matrix of the k-nearest neighbor graph. We first calculate the similarity matrix $\mathbf{S} \in \mathbb{R}^{n*n}$ for n nodes. There are multiple ways to obtain $\mathbf{S}$, such as Cosine similarity and the heat kernel function.

(1) Cosine similarity: The x_i and x_j be the eigenvectors (feature vector) of the nodes i and j, respectively. As depicted in Eq. (1), we use the cosine of the angle between two vectors to measure similarity, defined as

$$S_{i,j} = \frac{x_i \cdot x_j}{\left|x_i \| x_j\right|}. \tag{1}$$

(2) Heat kernel function: The similarity of the nodes based on the heat kernel function is calculated in Eq. (2):

$$S_{i,j} = e^{-\frac{\left\|\mathbf{x}_i - \mathbf{x}_j\right\|^2}{t}}, \tag{2}$$

where t is the time parameter in the heat conduction equation, and we set t to 2. We use the cosine similarity to obtain $\mathbf{S}$, select the first k pairs of similar nodes to set the edges for each node, and finally obtain the adjacency matrix $\mathbf{A}_f$.

3.4 Feature Graph Autoencoder Module

After obtaining the underlying structure of the feature space, we design an FGA module to learn a specific representation. The FGA module consists of a feature graph encoder and an inner product decoder.

Feature graph encoder. We use the GCN layer as a graph encoder, where the input is $\mathbf{A}_f$. The output of the l-th layer can be calculated in Eq. (3):

$$\mathbf{Z}_f^{(l)} = \mathrm{ReLU}\left(\widetilde{\mathbf{D}}_f^{-\frac{1}{2}} \widetilde{\mathbf{A}}_f \widetilde{\mathbf{D}}_f^{-\frac{1}{2}} \mathbf{Z}_f^{(l-1)} \mathbf{W}_f^{(l)}\right), \tag{3}$$

where $\mathbf{W}_f^{(l)}$ is the weight matrix of the l-th layer of the GCN. We use the *ReLU* activation function with the initialization value $\mathbf{Z}_f^{(0)} = \mathbf{X}$; $\widetilde{\mathbf{A}}_f$=$\mathbf{A}_f$+$\mathbf{I}_f$, and $\widetilde{\mathbf{D}}_f$ represents the pairwise angle matrices of $\widetilde{\mathbf{A}}_f$. In addition, $\mathbf{Z}_f$ is the embedding output of the last layer, allowing us to capture specific information from the feature space.

Inner product decoder. We employ a simple inner product decoder to predict the links between nodes, which is efficient and flexible, as described in Eq. (4):

$$\mathbf{A}_f^* = \mathrm{sigmoid}\left(z_f^{i\,T} z_f^j\right), \tag{4}$$

where $\mathbf{A}_f^*$ is the adjacency matrix of the feature graph reconstructed by $\mathbf{A}_f$. The reconstruction loss objective function is defined as L_R in Eq. (5):

$$L_R = \mathrm{loss}\left(\mathbf{A}_f, \mathbf{A}_f^*\right). \tag{5}$$

3.5 Structure Graph Attention Network Module

After extracting feature space information, we consider that the topological space is usually complicated, and neighbor nodes contribute to the target node. Therefore, it is difficult for low-order neighbors to provide global structure information, so higher-order neighbors need to be used. We adopt a variant of the graph attention network [33] with two effective techniques: attribute value

and topological distance. We have the original input graph $G_t = (\mathbf{A_t}, \mathbf{X})$, where $\mathbf{A_t} = \mathbf{A}$. The output of the node i in the l-th layer can be expressed as in Eq. (6):

$$z_i^{(l)} = \sigma\left(\sum_{j \in N_i} \alpha_{i,j} \mathbf{W}_t z_j^{(l-1)}\right), \tag{6}$$

where N_i is the neighbor of the node i, and $\mathbf{W}_t$ is the weight matrix shared by each node. σ is a nonlinear function, and $z_t^{(l)}$ consists of $z_i^{(l)}, (i \in \{1, ..., n\})$. $\alpha_{i,j}$ is the attention factor, which indicates the importance of the neighbor node j to node i.

Attribute values. Coefficient $\alpha_{i,j}$ is expressed as a single-layer feed-forward neural network after z_i, z_j stacking, as defined in Eq. (7):

$$e_{i,j} = \overrightarrow{a}^T \left(\mathbf{W}_t z_i, \mathbf{W}_t z_j\right), \tag{7}$$

where $\overrightarrow{a} \in \mathbb{R}^{2h'}$ is the weight vector of the single layer of the feed-forward neural network.

Topological distance. An approximation matrix is obtained by considering the t-order neighbor nodes in the graph, as defined in Eq. (8):

$$\mathbf{M} = \left(\mathbf{B} + \mathbf{B}^2 + \cdots + \mathbf{B}^t\right)/t, \tag{8}$$

where $\mathbf{B}$ is the transfer matrix. If $A_{i,j} = 1$, then the elements $B_{i,j} = 1/d_i$; otherwise, $B_{i,j} = 0$. d_i is the degree of node i, so $M_{i,j}$ denotes the topological correlation between nodes j and i up to the t-order. In this case, N_i denotes the neighboring nodes of i in $\mathbf{M}$; that is, if $M_{i,j} > 0$, then j is the neighbor of i. t can be flexibly selected according to the distribution of the target data set, and we set $t = 2$.

To calculate the attention coefficient, it is usually necessary to use the $softmax$ function to normalize the attention coefficients of all neighbors $j \in N_i$. Based on the attention coefficients, we can obtain the comparability between nodes by using applying the $LeakyReLU$ nonlinear function:

$$\alpha_{i,j} = \frac{\exp\left(\text{LeakyReLU}\left(\overrightarrow{a}^T \left[\mathbf{W}_t z_i, \mathbf{W}_t z_j\right]\right)\right)}{\sum_{k \in N_i} \exp\left(\text{LeakyReLU}\left(\overrightarrow{a}^T \left[\mathbf{W}_t z_i, \mathbf{W}_t z_k\right]\right)\right)}. \tag{9}$$

After adding the topological weight $\mathbf{M}$, the coefficients can be expressed as in Eq. (10):

$$\alpha_{i,j} = \frac{\exp\left(\text{LeakyReLU}\left(M_{i,j}\overrightarrow{a}^T \left[\mathbf{W}_t z_i, \mathbf{W}_t z_j\right]\right)\right)}{\sum_{k \in N_i} \exp\left(\text{LeakyReLU}\left(M_{i,k}\overrightarrow{a}^T \left[\mathbf{W}_t z_i, \mathbf{W}_t z_k\right]\right)\right)}. \tag{10}$$

3.6 Attention Mechanism

Node features are propagated in the topological space to obtain a specific representation $z_t^{(l)}$. Based on the inter-feature and topological similarities, graph clustering results are associated with one of these features or a combination of them. We use the specific representations $\mathbf{Z}_f$ and $\mathbf{Z}_t$ to learn the importance $(\boldsymbol{\alpha}_f, \boldsymbol{\alpha}_t)$ through the attention mechanism $att(\mathbf{Z}_f, \mathbf{Z}_t)$. For example, we focus on node i, where its embedding in $\mathbf{Z}_f$ is $z_f^i \in \mathbb{R}^{1\times h}$. First, we transform the embedding through a nonlinear transformation, and we use the one shared attention vector $q \in \mathbb{R}^{h'\times 1}$ to obtain the attention value ω_f^i, which can be expressed as in Eq. (11):

$$\omega_f^i = q^{\mathrm{T}} \cdot \tanh\left(\mathbf{W} \cdot \left(z_f^i\right)^{\mathrm{T}} + b\right), \tag{11}$$

where $\mathbf{W} \in \mathbb{R}^{h' \times h}$ is the weight matrix, and $b \in \mathbb{R}^{h' \times 1}$ is the bias vector. Similarly, we can obtain the attention values ω_t^i for node i in embedding $\mathbf{Z}_t$. We normalize the attention value ω_t^i and ω_f^i with the $softmax$ function, which can be calculated as in Eq. (12):

$$\alpha_f^i = softmax\left(\omega_f^i\right) = \frac{\exp\left(\omega_f^i\right)}{\exp\left(\omega_f^i\right) + \exp\left(\omega_t^i\right)}. \tag{12}$$

Similarly, $\alpha_t^i = softmax(\omega_t^i)$. For all the n nodes in the l-th layer, we have the learned attention value matrix $\boldsymbol{\alpha}_f^{(l)} = [\alpha_f^i, i \in n]$, $\boldsymbol{\alpha}_t^{(l)} = [\alpha_t^i, i \in n]$. Then, we combine $\mathbf{Z}_f$ and $\mathbf{Z}_t$ to obtain a more complete and powerful representation based on importance, as calculated using Eq. (13):

$$\mathbf{Z}_{f+t} = \boldsymbol{\alpha}_f \cdot \mathbf{Z}_f + \boldsymbol{\alpha}_t \cdot \mathbf{Z}_t. \tag{13}$$

In the structure graph attention network module, we use $\hat{z}_j^{(l-1)}$ and $\hat{z}_j^{(l-1)} \in \mathbf{Z}_{f+t}{}^{(l-1)}$ as the input to each layer. The output of node i at the l-th layer can be expressed as in Eq. (14):

$$\hat{\mathbf{Z}}_i^{(l)} = \sigma\left(\sum_{j \in N_i} \alpha_{i,j} \mathbf{W}_t \hat{\mathbf{Z}}_j^{(l-1)}\right). \tag{14}$$

After obtaining the embedding representation $\mathbf{Z}$, the multiple classification layers of the second-order structured graph learning module use the $softmax$ function, and we denote the class predictions for n nodes as $\mathbf{Q}_z = [\hat{q}_{ic}] \in \mathbb{R}^{n \times c}$, where $\hat{q}_{ic}$ is the probability that node i belongs to cluster center c. $\mathbf{Q}_z$ can be considered a probability distribution. $\mathbf{Q}_z$ can be calculated as in Eq. (15):

$$\mathbf{Q}_z = softmax(\mathbf{Z}). \tag{15}$$

3.7 Double Self-supervision Clustering Module

The FGA module and SGAT module integrate feature and structural space information, but they are not designed for clustering. Hence, the learned embedding representation may not be suitable for clustering. A good clustering distribution should ensure that nodes in the same cluster are dense, and nodes between different clusters are far apart. Therefore, we need an objective function to guide the embedding learning process. Inspired by SDCN [41], we adopt the dual self-supervision clustering objective function, which can unify the FGA module and SGAT module and effectively train these two modules end to end for attributed graph clustering.

In particular, for a sample i and cluster j, we use the student t distribution as a kernel function to measure the similarity between the data representation z_f^i and the cluster center vector u_j, as defined in Eq. (16):

$$q_{i,j} = \frac{\left(1 + \left\|z_f^i - \mu_j\right\|^2\right)^{-1}}{\sum_i \left(1 + \left\|z_f^i - \mu_j\right\|^2\right)^{-1}}, \tag{16}$$

where z_f^i is the i-th row of $\mathbf{Z}_f^l$, and u_j is the cluster center. We consider that $q_{i,j}$ is the probability of assigning node i to cluster center u_j, and that $\mathbf{Q} = [q_{i,j}]$ is the soft label distribution of all nodes. After obtaining the soft label distribution $\mathbf{Q}$, we optimize the embedding representation by learning from the high-confidence soft assignment. Then, we want to make the embedding representation close to the cluster centers and improve clustering cohesion. Therefore, we compute the target

distribution $\mathbf{P}$ in Eq. (17):

$$p_{i,j} = \frac{q_{i,j}^2 / \sum_i q_{i,j}}{\sum_j \left(q_{i,j}^2 / \sum_i q_{i,j} \right)}, \tag{17}$$

where $\sum_i q_{i,j}$ is a soft cluster frequency used to normalize the contribution of each center-of-mass loss and prevent larger clusters from distorting the embedding space. As high-confidence nodes (close to the cluster centers) are considered plausible in $\mathbf{Q}$, the target distribution $\mathbf{P}$ will be raised $\mathbf{Q}$ to the second power to emphasize their role, which leads to the objective function, it can be regarded as Clustering Loss as L_C in Eq. (18):

$$L_C = KL(\mathbf{P} \| \mathbf{Q}) = \sum_i \sum_j p_{i,j} \log \frac{p_{i,j}}{q_{i,j}}. \tag{18}$$

By minimizing the Kullback–Leibler divergence loss between the $\mathbf{Q}$ and $\mathbf{P}$ distributions, the target distribution $\mathbf{P}$ can help the FGA module to learn a better feature of space representation. This process is an important step of the attributed graph clustering task.

To train the structure graph attention network module, as the module will eventually provide a soft distribution $\mathbf{Q}_z$, we can use the distribution $\mathbf{P}$ to supervise $\mathbf{Q}_z$. We represent the Probability Distribution Loss as L_{PD} in Eq. (19):

$$L_{PD} = KL\left(\mathbf{P} \| \mathbf{Q}_z\right). \tag{19}$$

Overall objective function. We jointly optimize the two modules of embedding and clustering learning and define the overall objective function as:

$$L = \mu L_R + L_C + L_{PD}, \tag{20}$$

where $\mu > 0$ is the hyperparameter controlling the loss of feature graph reconstruction.

We choose the soft assignment distribution $\mathbf{Q}_z$ obtained in the last iteration as the final clustering result. The label assigned to node i can be obtained as in Eq. (21):

$$l_i = \arg\max_j \hat{q}_{i,j}, \tag{21}$$

where $\hat{q}_{i,j}$ is calculated by using Eq. (15).

3.8 Optimization Strategy

We pre-train the FGA module to obtain a well-trained representation $\mathbf{Z}_f$, and integrate it with the SGAT module representation $\mathbf{Z}_t$ to finally obtain the embedding $\mathbf{Z}$. The dual self-supervision clustering module is used to improve the embedding representation of the learning of two modules. To initialize the clustering centers, we perform k-means on $\mathbf{Z}_f$ to obtain the initial centroid $\{\mu\}_{j=1}^k$. Four parameters must be updated: the FGA $\mathbf{W}_f^{(l)}$, clustering center u_j, SGAT $\mathbf{W}_t^{(l)}$, and attention factor $\boldsymbol{\alpha}$. Then, it updates the feature graph encoder weights and clustering centers. By fixing the target distribution $\mathbf{P}$ and given n nodes, we can calculate the gradient of the clustering center u_j relative to L_c as in Eq. (22):

$$\frac{\partial L_C}{\partial u_j} = 2 \sum_{i=1}^{N} \left(1 + \left\| z_f^i - u_j \right\|^2 \right)^{-1} \left(q_{i,j} - p_{i,j} \right) \left(z_f^i - u_j \right). \tag{22}$$

Given a learning rate of φ, we can update u_j by using Eq. (23):

$$u_j = u_j - \phi \frac{\partial L_c}{\partial u_j}. \tag{23}$$

In addition, the feature graph encoder weights are updated in Eq. (24):

$$\mathbf{W}_f^{(l)} = \mathbf{W}_f^{(l)} - \varphi\left(\frac{\partial L_C}{\partial \mathbf{W}_f^{(l)}} + \mu\frac{\partial L_R}{\partial \mathbf{W}_f^{(l)}}\right). \tag{24}$$

We update the structure graph attention network weights by using Eq. (25):

$$\mathbf{W}_t^{(l)} = \mathbf{W}_t^{(l)} - \varphi\left(\frac{\partial L_{PD}}{\partial \mathbf{W}_t^{(l)}}\right). \tag{25}$$

Note that the coefficients are updated when integrating Z_f and Z_t, as defined in Eq. (26):

$$\boldsymbol{\alpha}^{(l)} = \boldsymbol{\alpha}^{(l)} - \varphi\left(\frac{\partial L_{PD}}{\partial \boldsymbol{\alpha}^{(l)}}\right). \tag{26}$$

4 EXPERIMENTS

Our experiments cluster users to determine user categories based on different networks under a self-supervision system. The data sets of communities, citation networks, and ACM are used to obtain the users' different personality characteristics and preferences to carry out corresponding intelligent interactions. For example, according to the preference of the messages, the interaction system needs to judge whether the message is helpful and whether to show it to the user. The intelligent interaction system needs to be calibrated according to the user's preference. In HCI, dialogue intelligence systems, emotion detection systems, and many other recommendation systems, the execution of user instructions does not explicitly consider the user's background and preference. Therefore, the user clustering model can meet the current massive intelligent interaction requirements.

4.1 Experimental Setup

Data sets: Our method is evaluated on five real-world data sets with the specific user attributes described in Table 6.

- Citeseer [52] is a research paper citation network, where nodes are publications and edges are citation links. Node attributes are bag-of-words representations of papers, and nodes are divided into six regions.
- UAI2010 [53] contains 3,067 nodes and 28,311 edges, and has been verified for community detection.
- ACM [54] network is extracted from the ACM data set, where nodes represent papers, and edges between papers represent the same author. The paper information is divided into different categories, such as database, wireless communication, and data mining. The paper classification function is a bag-of-words representation of the paper keywords.
- DBLP [55] is an author network. The authors represent the fields of database, data mining, machine learning, and information retrieval. In addition, there is a cooperative relationship among coauthors. We mark each author's research area based on the conversation they submitted. The author characteristics are the elements of the word package, which are composed of keywords.
- BlogCatalog [53]: It is a social network with bloggers and their social relationships from the BlogCatalog website. Node attributes are constructed by the keywords of user profiles, labels represent the topic categories provided by the authors, and all nodes are divided into 6 classes.

Baseline methods: We compared our model with attribute-, graph-structure-, and GCN-based methods for graph clustering, including 8 competitive approaches.

Table 1. The details of the experimental data set.

data set	Sample size	Category	Dimension	Edge
UAI2010	3067	19	4973	28311
BlogCatalog	5196	6	8189	171743
ACM	3025	3	1870	13128
DBLP	4057	4	334	3528
Citeseer	3327	6	3703	4552

- K-means [56] is a classical clustering method based on raw data.
- DEC [23] is a deep clustering method that designs clustering objectives to guide the learning of data representations.
- IDEC [24] adds reconstruction loss to DEC to learn a better representation.
- Spectral-g is based on a graph adjacency matrix.
- SDNE [25] is a network embedding algorithm that uses an autoencoder structure to optimize first-and second-order similarity, learning a vector representation that preserves both the local and global structure.
- GAE [32] is an unsupervised graph embedding method that uses a GCN to learn data representations.
- DAEGC [33] uses an attention network (GAT) to learn node representations, and clustering loss to supervise graph clustering.
- SDCN [41] fuses the data representation learned by a self-encoder with the structural representation learned by GCN through a transfer operator, and designs a dual self-supervision approach for clustering.
- AMGCN [53] improves the fusion capability of GCN, and we apply it to graph clustering.

Evaluation metrics: Four clustering evaluation metrics are used: accuracy (ACC), regular mutual information quantity (NMI), Rand index (ARI), and equilibrium mean (F1-score). For each metric, a larger value indicates a better result.

Parameter settings: All baseline methods are initialized using their suggested parameters, and we use them to obtain the best performance. For our model, we pretrain a feature graph autoencoder module. Its hidden layer size is $hid1 \in \{512, 768\}$, and its output size is $hid2 \in \{32, 128, 768\}$. The model is trained for 30 iterations, with a learning rate of 10^{-3}. We introduce a structure graph attention network module with the same size as the structure graph encoder for the first two layers of structure graph learning. Then, we add the third layer with a size of $n_cluster$, which is the cluster class size. We consider the second-order neighbors and set $M = \left(B + B^2\right)/2$. We use $lr = 0.0005$ and $weight_decay \in \left\{5e^{-3}, 5^{-4}\right\}$ in the Adam optimizer. We set $k \in \{3, \ldots, 9\}$ to construct the feature map, and set the clustering loss hyperparameter to $\alpha = 0.1$. We run all comparative methods 10 times and report the average results.

4.2 Comparison of Node Clustering Results

The clustering results of each data set node are reported in Tables 2–6. The proposed DGENFS model performs best on most data sets in all benchmark tests. For the ACC metric, our model achieves a maximum improvement of 5.02% on the Citeseer data set, 8.09% on Uai, and 8.96% on BlogCatalog. The results prove the effectiveness of the DGENFS model. For the ACM data set, note that some metrics are weaker than the baseline methods. The reason may be that the training time is too short owing to the small amount of data, and the model extraction feature and structure

Table 2. Clustering results on Citeseer data set.

Method	ACC%	NMI%	ARI%	F1%
Spectral-g	23.75	2.830	2.07	17.03
SDNE	26.60	3.850	3.35	25.55
K-means	40.91	18.68	14.88	37.98
DEC	55.77	29.86	28.99	52.97
IDEC	56.87	30.66	30.45	53.71
GAE	61.35	34.63	33.55	57.36
DAEGC	64.64	34.45	36.48	60.37
SDCN	65.96	38.71	40.17	63.62
AMGCN	65.13	37.87	38.02	60.40
Ours	**69.27**	**43.59**	**44.72**	**64.98**

Table 3. Clustering results on DBLP data set.

Method	ACC%	NMI%	ARI%	F1%
Spectral-g	30.56	0.98	1.08	27.22
SDNE	30.24	1.46	1.30	29.15
K-means	38.55	11.47	7.04	31.70
DEC	58.38	27.26	27.36	57.17
IDEC	60.58	28.48	26.83	60.79
GAE	61.21	30.80	22.02	61.41
DAEGC	63.81	34.88	39.01	61.03
SDCN	68.05	39.50	39.15	67.11
AMGCN	74.09	41.84	44.33	73.44
Ours	**75.81**	**43.99**	**47.54**	**75.47**

Table 4. Clustering results on ACM data set.

Method	ACC%	NMI%	ARI%	F1%
Spectral-g	34.80	18.04	10.61	32.34
SDNE	38.66	1.01	0.88	38.36
K-means	66.80	32.39	30.31	67.04
DEC	85.47	57.44	62.06	85.11
IDEC	86.07	58.09	63.84	86.18
GAE	84.52	55.38	59.46	84.65
DAEGC	86.71	59.88	65.17	86.62
SDCN	90.45	68.31	73.91	90.42
AMGCN	91.32	**70.78**	**76.15**	**91.32**
Ours	**91.34**	69.68	75.96	91.18

information have not been well learned, so the two kinds of information are not well integrated. This also indicates that our method may not obtain good results on small data sets.

Table 5. Clustering results on BlogCatalog data set.

Method	ACC%	NMI%	ARI%	F1%
Spectral-g	33.97	16.30	7.17	26.91
SDNE	39.71	20.20	13.66	39.24
K-means	31.69	17.78	8.06	25.79
DEC	42.20	29.63	20.35	35.13
IDEC	43.63	29.54	21.65	36.03
GAE	45.30	29.31	24.08	41.84
DAEGC	48.24	28.49	24.61	45.06
SDCN	38.07	17.59	12.59	25.67
AMGCN	53.46	**39.06**	25.21	50.68
Ours	**58.25**	37.64	**33.86**	**50.71**

Table 6. Clustering results on UAI2010 data set.

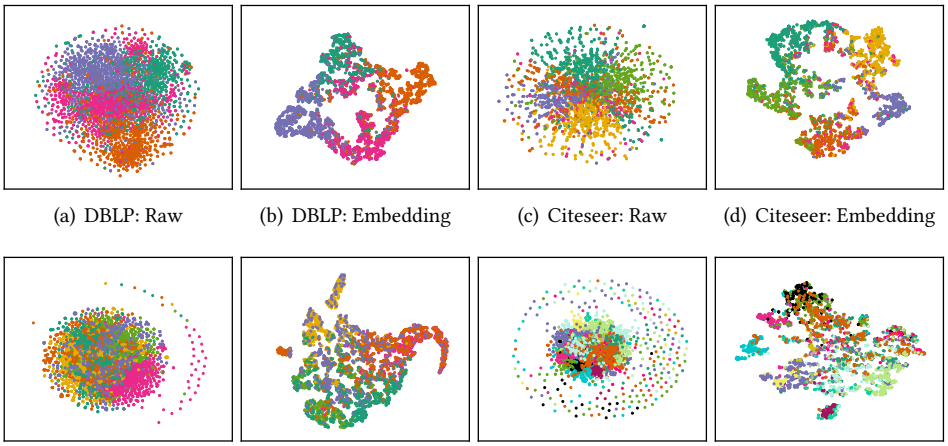
Method	ACC%	NMI%	ARI%	F1%
Spectral-g	30.81	27.88	10.75	25.98
SDNE	27.90	21.83	9.66	22.62
K-means	35.77	36.95	16.93	28.03
DEC	33.61	31.46	15.52	22.42
IDEC	34.44	31.59	17.60	23.94
GAE	33.73	34.22	17.95	28.36
DAEGC	37.24	33.85	20.56	22.48
SDCN	30.06	27.36	12.50	15.77
AMGCN	40.40	41.28	24.18	33.01
Ours	**43.67**	**43.32**	**26.13**	**34.60**

Comparing attribute-based, graph-structure-based, and GCN-based methods, we can see that there are two types of graph clustering methods. One group of methods uses both node features and graph structures, while the other group uses only one aspect of information. We can find that the former group generally has better performance than the latter. For example, in the Citeseer and DBLP data sets, GAE, DAEGC, SDCN, and AMGCN outperform the methods that use only node features or topology. This observation indicates the necessity of fusing attribute features and topology for graph clustering.

The clustering loss function (Eq. (19)) is used in most baseline methods, such as DEC, IDEC, SDCN, and DAEGC. This indicates that this function obtains high-confidence soft labels to make the learned features close to the clustering center.

Our proposed model outperforms the GCN-based method on most of the data sets. GCN-based baseline methods mainly use structural similarity to obtain embedding features. Specifically, GAE and VGAE use graph structure reconstruction loss, while SDCN uses the original graph structure to increase the node feature reconstruction loss. Therefore, the graph embedding results learned from these baseline methods focus more on structural similarity while ignoring the similarity between node features, and they are not insufficient in fusing the two types of information. Although DAEGC and SDCN introduce different clustering loss-driven strategies, the clustering results still

depend on graph structure reconstruction loss or node feature reconstruction loss. Therefore, these methods may not be able to effectively combine node features and graph structure information. In addition to introducing global clustering loss, DGENFS also introduces feature similarity to guide the fusion of structural similarity and feature similarity.



(a) DBLP: Raw (b) DBLP: Embedding (c) Citeseer: Raw (d) Citeseer: Embedding

(e) BlogCatalog: Raw (f) BlogCatalog: Embedding (g) Uai: Raw (h) Uai: Embedding

Fig. 2. T-SNE visualization of DGENFS.

4.3 Visualization

To visually demonstrate the effectiveness of our proposed model, we carry out the visualization of the clustering results of comparative methods on the DBLP, Citeseer, BlogCatalog, and Uai data sets. We use the last layer of the second-order structured graph learning module as the embedding output. Then, we further use T-SNE [41] to describe the distribution of the original data and the learned embedding. The results of the four data sets are color-coded according to the ground-truth labels. As shown in Fig. 2, we know that as the training progresses, the embedding becomes more obvious. Meanwhile, the overlap is reduced, and the learned embedding structures become more compact. Nodes within the same cluster gradually gather, and nodes between different clusters gradually move away from each other.

4.4 Embedding Dimension Analysis

The first two layers in the feature graph autoencoder module and the structure graph attention network module have the same embedding dimensions $d0$ and $d1$. Therefore, this work focuses on the impact of the embedding dimension on the user clustering results. We fix the value of d_0 as $[512, 768]$ and search for the impact of $d1$ on the clustering accuracy in terms of ACC and ARI. Experimental results are shown in Fig. 3 and Fig. 4.

We can see from the experimental results that the overall performance of the model varies in different dimensions. In the Uai2010 and Citeseer cases, the performance of the model improves as the number of dimensions increases from 16 to 256. In the DBLP, ACM, and BlogCatlog cases, the clustering performance fluctuates with the increase of the number of dimensions, but not significantly. This also indicates that our model has better stability under different embedding

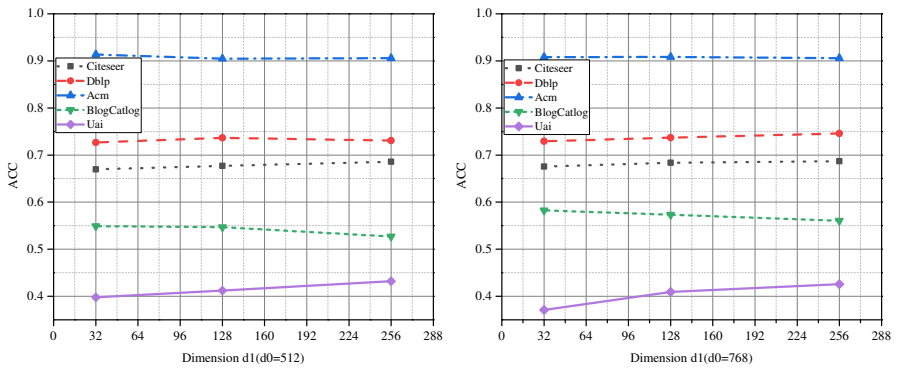


Fig. 3. Impact of $d1$ of ACC on different methods.

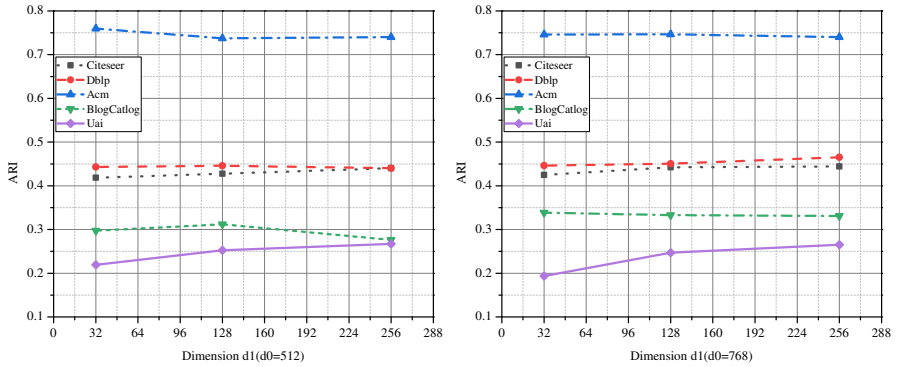


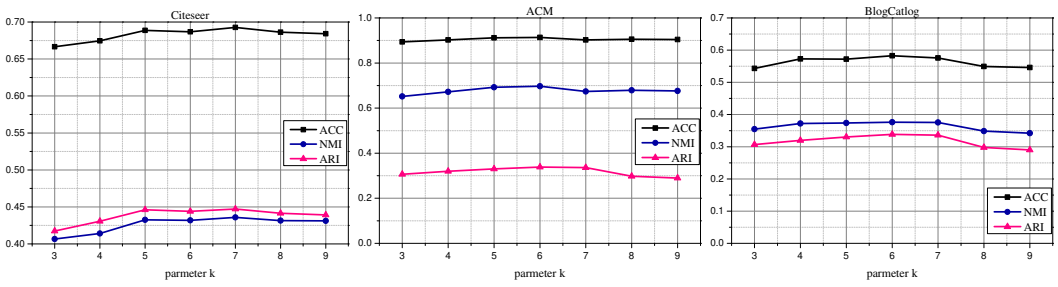
Fig. 4. Impact of $d1$ of ARI on different methods.

dimensions. In addition, we set different embedding dimensions according to the experimental results; namely, we set $d0 = 768$ and $d1 = 256$ for the Citeseer and Dblp data sets; $d0 = 512$ and $d1 = 32$ for the Acm data set; $d0 = 768$ and $d1 = 32$ for BlogCatlog; and $d0 = 512$ and $d1 = 256$ for the Uai data set.

4.5 Parameter Variable Study

To compare the influence of the top k-neighborhood in the k-nearest neighbor (KNN) graph on the clustering results, we design sensitivity experiments for the parameter k on different data sets. We set k to the range of 3 to 9 to investigate the model performance. As shown in Fig. 5, for Citeseer, we find that the ACC, NMI, and ARI gradually increase to achieve the best performance with k=7. For Acm and BlogCatlog, the best performance is obtained when k=6. This proves that our method learns useful feature information from feature similarity, and is complementary to the structural information learned from structural similarity in the fusion process. Another finding is that after the two critical points (k=6 and k=7), the performance begins to gradually decrease, which may be because as the value of k increases, the feature graph becomes denser. Such results may introduce more noise and make the feature information less distinguishable. In this case, it is not conducive to fusing with the structural information extracted from the topological graph, which affects the

clustering results. Moreover, for Citeseer, ACM, and BlogCatalog data sets, we find that ACC, NMI, and ARI gradually increase initially, reaching the best performance at k=6, and then decrease.



(a) Citeseer (b) ACM (c) BlogCatalog

Fig. 5. Cluster results with parameter k on Citeseer, ACM and BlogCatalog.

4.6 Analysis of Attention Mechanisms

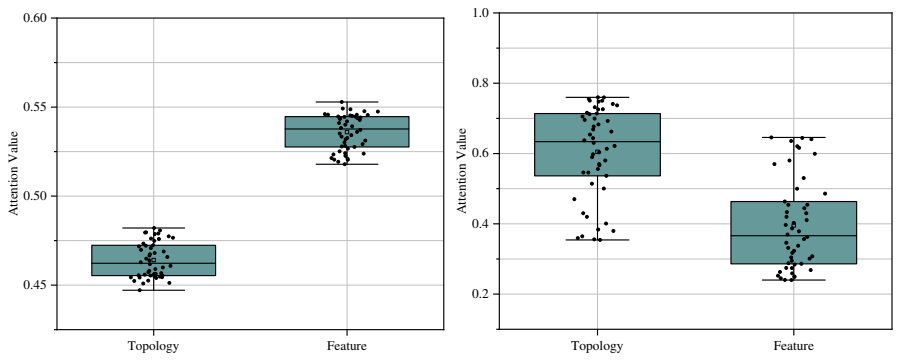
To investigate whether the values learned by the attention mechanism are meaningful, we analyze the last layer of learning values, including the attention distribution and learning trends.

Attention distribution. The DGENFS model learns information related to the attention values from the feature graph autoencoder and the structure graph attention network. We analyze the attention distribution of the Uai and BlogCatalog data sets, and the results are shown in Fig. 5(c). It can be seen that for the Uai data set, the attention value assigned to extract information from the topological space is greater than that of the feature space, which means that the information in the topological space is more important. The BlogCatalog data set is the opposite, which shows that our model can assign more attention to more important information in the double self-supervision module.

Attention learning trends. We analyze the changes in the attention values corresponding to two specific messages during the training iterations. We select the Uai and BlogCatalog data sets. The experimental results are shown in Fig. 6, where the X-axis and Y-axis represent the number of iterations and the attention values, respectively. As shown in Fig. 7, for the Uai data set, the attention value in the topological space is larger than that in the feature space at the beginning. As the iteration proceeds, the attention value in the topological space first decreases and then increases. In contrast, the attention value in the feature spatial first increases and then decreases. For the BlogCatalog data set, the attention value in the feature space is always greater than that in the structure space. We find that DGENFS benefits from the impact of clustering loss. As the iteration progresses, it can learn the importance of different spatial information and make continuous adjustments.

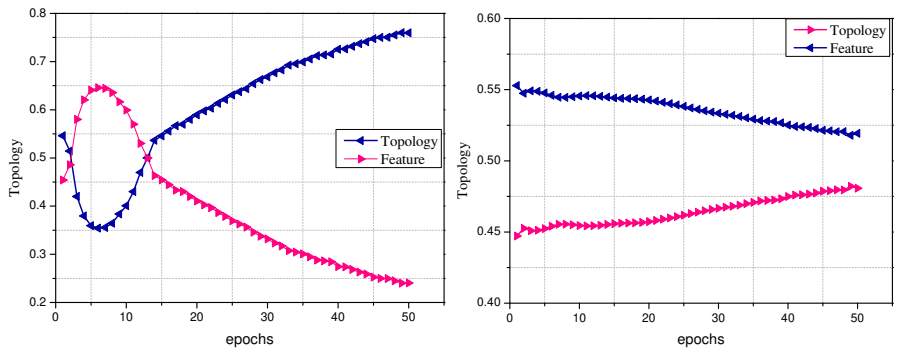
5 CONCLUSION

In this paper, we have explored the capability of network embedding in HCI from the topological and feature similarity of the users. In order to excavate the topology and node feature information, we have proposed a graph embedding network consisting of the FGA module, SGAT module, and DSS module. Moreover, the attention mechanism fuses them according to their importance. Finally, the HCI users are clustered in a self-supervision way according to their different personality characteristics and preferences to carry out corresponding intelligent HCI. The efficiency of the



(a) Uai (b) BlogCatalog

Fig. 6. Analysis of attention distribution.



(a) Uai (b) BlogCatalog

Fig. 7. Attention changing trends.

method has been demonstrated by comparing it with remarkable algorithms on various real work data sets. In the future, we will focus on the larger complex network from a multi-view attributed graph to further improve the clustering performance of HCI users.

REFERENCES

[1] Jun Wang, Lijun Yin, and Jason Moore. Using geometric properties of topographic manifold to detect and track eyes for human-computer interaction. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, 3(4):1–20, 2007.

[2] Zhihan Lv, Alaa Halawani, Shengzhong Feng, Haibo Li, and Shafiq Ur Réhman. Multimodal hand and foot gesture interaction for handheld devices. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, 11(1s):1–19, 2014.

[3] Alphonse Chapanis. *Man-machine engineering*. Wadsworth Pub. Co., Inc., 1965.

[4] Donald A. Norman and Stephen W. Draper. *User centered system design: New perspectives on human-computer interaction*. CUMINCAD, USA, 1986.

[5] Christine L. Lisetti. Affective computing. *Pattern Anal. Appl.*, 1(1):71–73, 1998.

[6] Maja Pantic, Anton Nijholt, Alex Pentland, and Thomas S Huanag. Human-centred intelligent human? computer interaction (hci^2): how far are we from attaining it? *International Journal of Autonomous and Adaptive Communications Systems*, 1(2):168–187, 2008.

[7] Eleonora Mencarini, Amon Rapp, Lia Tirabeni, and Massimo. Zancanaro. Designing wearable systems for sports: A review of trends and opportunities in human–computer interaction. *IEEE Transactions on Human-Machine Systems*, 49(4):314–325, 2019.

[8] Stina Nylander, Jakob Tholander, Florian Mueller, and Joe Marshall. Hci and sports. pages 115–118, 2014.

[9] Azin Semsar and Ali Asghar Nazari Shirehjini. Multimedia-supported virtual experiment for online user–system trust studies. *Multimedia Systems*, 23(5):583–597, 2017.

[10] Yun Yang and Jianmin Jiang. Adaptive bi-weighting toward automatic initialization and model selection for hmm-based hybrid meta-clustering ensembles. *IEEE transactions on cybernetics*, 49(5):1657–1668, 2018,.

[11] Yun Yang and Jianmin Jiang. Hybrid sampling-based clustering ensemble with global and local constitutions. *IEEE transactions on neural networks and learning systems*, 27(5):952–965, 2015.

[12] Yun Yang and Jianmin Jiang. Bi-weighted ensemble via hmm-based approaches for temporal data clustering. *Pattern Recognition*, 76:391–403, 2017.

[13] Xiaofei He, Deng Cai, Ji-Rong Wen, Wei-Ying Ma, and Hong-Jiang Zhang. Clustering and searching www images using link and page layout analysis. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, 3(2):10–es, 2007.

[14] Mingxing Duan, Kenli Li, Xiangke Liao, Keqin Li, and Qi Tian. Features-enhanced multi-attribute estimation with convolutional tensor correlation fusion network. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, 15(3s):1–23, 2019.

[15] Yin Fan, Xiangju Lu, Dian Li, and Yuanliu Liu. Video-based emotion recognition using cnn-rnn and c3d hybrid networks. In *Proceedings of the 18th ACM international conference on multimodal interaction*, pages 445–450, 2016.

[16] Arantxa Villanueva, Victoria Ponz, Laura Sesma-Sanchez, Mikel Ariz, Sonia Porta, and Rafael Cabeza. Hybrid method based on topography for robust detection of iris center and eye corners. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, 9(4):1–20, 2013.

[17] Shizhe Chen, Qin Jin, Jinming Zhao, and Shuai Wang. Multimodal multi-task learning for dimensional and continuous emotion recognition. In *Proceedings of the 7th Annual Workshop on Audio/Visual Emotion Challenge*, pages 19–26, 2017.

[18] André Pimenta, Davide Carneiro, José Neves, and Paulo Novais. A neural network to classify fatigue from human–computer interaction. *Neurocomputing*, 172:413–426, 2016.

[19] HongYun Cai, Vincent W. Zheng, and Kevin Chen-Chuan Chang. A comprehensive survey of graph embedding: Problems, techniques, and applications. *IEEE Transactions on Knowledge and Data Engineering*, 30(9):1616–1637, 2018.

[20] Santo Fortunato. Community detection in graphs. *Physics reports*, 486(3-5):75–174, 2010.

[21] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S Yu Philip. A comprehensive survey on graph neural networks. *IEEE transactions on neural networks and learning systems*, 32(1):4–24, 2020.

[22] Fei Tian, Bin Gao, Qing Cui, Enhong Chen, and Tie-Yan Liu. Learning deep representations for graph clustering. 28(1), 2014.

[23] Junyuan Xie, Ross Girshick, and Ali Farhadi. Unsupervised deep embedding for clustering analysis. pages 478–487, 2016.

[24] Xifeng Guo, Long Gao, Xinwang Liu, and Jianping Yin. Improved deep embedded clustering with local structure preservation. In *Ijcai*, pages 1753–1759, 2017.

[25] Daixin Wang, Peng Cui, and Wenwu Zhu. Structural deep network embedding. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1225–1234, 2016.

[26] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. Line: Large-scale information network embedding. pages 1067–1077, 2015.

[27] Yu Cao, Meng Fang, and Dacheng Tao. Bag: Bi-directional attention entity graph convolutional network for multi-hop reasoning question answering. *arXiv preprint arXiv:1904.04969*, 2019.

[28] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826*, 2018.

[29] Jie Zhou, Ganqu Cui, Shengding Hu, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, and Maosong Sun. Graph neural networks: A review of methods and applications. *AI Open*, 1:57–81, 2020.

[30] Keyulu Xu, Chengtao Li, Yonglong Tian, Tomohiro Sonobe, Ken-ichi Kawarabayashi, and Stefanie Jegelka. Representation learning on graphs with jumping knowledge networks. In *International Conference on Machine Learning*, pages 5453–5462. PMLR, 2018.

[31] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

[32] Thomas N Kipf and Max Welling. Variational graph auto-encoders. *arXiv preprint arXiv:1611.07308*, 2016.

[33] Chun Wang, Shirui Pan, Ruiqi Hu, Guodong Long, Jing Jiang, and Chengqi Zhang. Attributed graph clustering: A deep attentional embedding approach. 2019.

[34] Xi Peng, Jiashi Feng, Shijie Xiao, Wei-Yun Yau, Joey Tianyi Zhou, and Songfan Yang. Structured autoencoders for subspace clustering. *IEEE Transactions on Image Processing*, 27(10):5076–5086, 2018.

[35] Shaosheng Cao, Wei Lu, and Qiongkai Xu. Grarep: Learning graph representations with global structural information. pages 891–900, 2015.

[36] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: Online learning of social representations. pages 701–710, 2014.

[37] Shaosheng Cao, Wei Lu, and Qiongkai Xu. Deep neural networks for learning graph representations. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 30, 2016.

[38] Zhen Zhang, Hongxia Yang, Jiajun Bu, Sheng Zhou, Pinggang Yu, Jianwei Zhang, Martin Ester, and Can Wang. Anrl: Attributed network representation learning via deep neural networks. In *Ijcai*, volume 18, pages 3155–3161, 2018.

[39] Chun Wang, Shirui Pan, Guodong Long, Xingquan Zhu, and Jing Jiang. Mgae: Marginalized graph autoencoder for graph clustering. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, pages 889–898, 2017.

[40] Xuelong Li, Hongyuan Zhang, and Rui Zhang. Embedding graph auto-encoder with joint clustering via adjacency sharing. *arXiv e-prints*, pages arXiv–2002, 2020.

[41] Deyu Bo, Xiao Wang, Chuan Shi, Meiqi Zhu, Emiao Lu, and Peng Cui. Structural deep clustering network. In *WWW '20: The Web Conference 2020*, pages 1400–1410, 2020.

[42] Xiaotong Zhang, Han Liu, Xiao-Ming Wu, Xianchao Zhang, and Xinyue Liu. Spectral embedding network for attributed graph clustering. *Neural Networks*, 142:388–396, 2021.

[43] Elisabeth Andre. Exploiting unconscious user signals in multimodal human-computer interaction. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, 9(1s):1–5, 2013.

[44] Shengping Zhang, Huiyu Zhou, Dong Xu, M Emre Celebi, and Thierry Bouwmans. Introduction to the special issue on multimodal machine learning for human behavior analysis, 2020.

[45] Juan M Silva, Mauricio Orozco, Jongeun Cha, Abdulmotaleb El Saddik, and Emil M Petriu. Human perception of haptic-to-video and haptic-to-audio skew in multimedia applications. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, 9(2):1–16, 2013.

[46] Sheng Li, Kang Li, and Yun Fu. Early recognition of 3d human actions. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, 14(1s):1–21, 2018.

[47] Jitao Sang and Changsheng Xu. Social influence analysis and application on multimedia sharing websites. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, 9(1s):1–24, 2013.

[48] Duc Son Nguyen and Quynh Mai Le. Hacking user in human-computer interaction design (hci). In *International Conference on Information and Computer Technologies*, pages 230–234, 2020.

[49] Wei Shen and Xiaolei. Zhou. Research on the human-computer interaction mode designed for elderly users. In *2015 4th International Conference on Computer Science and Network Technology (ICCSNT)*, volume 1, pages 374–377, 2015.

[50] Ikuesan R Adeyemi, Shukor Abd Razak, and Mazleena. Salleh. Individual difference for hci systems: Examining the probability of thinking style signature in online interaction. In *2016 4th International Conference on User Science and Engineering (i-USEr)*, pages 51–56, 2016.

[51] Fereshteh Jadidi Miandashti, Mohammad Izadi, Ali Asghar Nazari Shirehjini, and Shervin. Shirmohammadi. An empirical approach to modeling user-system interaction conflicts in smart homes. *IEEE Transactions on Human-Machine Systems*, 50(6):573–583, 2020.

[52] Andrew Ng, Michael Jordan, and Yair. Weiss. On spectral clustering: Analysis and an algorithm. In *Proceedings of the 14th International Conference on Neural Information Processing Systems: Natural and Synthetic*, 2001.

[53] Xiao Wang, Meiqi Zhu, Deyu Bo, Peng Cui, Chuan Shi, and Jian Pei. Am-gcn: Adaptive multi-channel graph convolutional networks. In *KDD '20: The 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2020.

[54] Van Der Maaten Laurens and Geoffrey Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(2605):2579–2605, 2008.

[55] Giannis Nikolentzos, Polykarpos Meladianos, and Michalis Vazirgiannis. Matching node embeddings for graph similarity. In *AAAI'17*, 2017.

[56] Matthew B. Hastings. Community detection as an inference problem. *Physical Review E*, 74(3):035102, 2006.