INTEREST COMMUNITY DETECTION BASED ON HETEROGENEOUS GRAPH NEURAL NETWORK AND TEMPORAL CONVOLUTIONAL NETWORK

Pei CAO [1,*], Yongyi LIN [2], Leilei SHI [3], Miaomiao LI [4]

Effectively dividing community structures aids in optimizing resource allocation. However, users with low activity pose challenges for direct interest extraction. Additionally, user interests migrate over time. Therefore, this paper proposes an interest community detection algorithm that leverages heterogeneous graph neural network (HGNN) and temporal convolutional network (TCN). Initially, a Latent Dirichlet Allocation (LDA) model extracts interest sets from multi-user generated content. Subsequently, HGNN dynamically learns node features. TCN then models the event propagation process to track user interests. Finally, interest communities are delineated based on user labels. Experiments verify the effectiveness of the proposed algorithm.

Keywords: community division, LDA, heterogeneous graph neural networks, temporal

1. Introduction

Social networks have become an important platform for communication and interaction in modern society. With the rapid development of social networks, their complexity and size are constantly increasing. In this context, community structure, as an important feature of social networks, has attracted increasing attention from researchers. Communities are usually defined as a set of nodes in a network that have denser connections among themselves than with other nodes [1-3]. Community structure reflects the close relationships and interest similarities among users in the network, which is very important for understanding the organizational structure and function of the network.

* Corresponding author
[1] Experimentalist, Engineering Technology Training Center, Nanjing University of Industry Technology, China, e-mail: caopeilove@yeah.net
[2] Engineer, School of Media Technology, Communication University of China Nanjing, China, e-mail: 349908202@qq.com
[3] Lecturer, School of Computer Science and Communication Engineering, Jiangsu University, China, e-mail: 937109472@qq.com
[4] Engineer, School of Electronic Information Engineering/School of Integrated Circuits, Nanjing University of Industry Technology, China, e-mail: 401904014@qq.com

A community is a set of nodes that appear in clusters within social networks, characterized by varying degrees of connections between these nodes [4]. It is generally accepted that relationships within the same community are closer, while relationships between different communities are sparser. Community detection is a key task in social network analysis, aiming to identify and divide the community structure in the network. Accurate community detection can provide support for many practical applications, such as personalized recommendations, information propagation analysis, and public opinion monitoring [5]. However, the dynamics and complexity of social networks pose enormous challenges to community detection. In particular, for new users or users with low activity, it is difficult to directly extract their interest features due to the limited content they generate. Moreover, user interests migrate over time, making it even more difficult to identify communities.

To address these issues, researchers have proposed various methods of community detection. Among them, methods based on user-generated content analysis have received widespread attention [6-9]. These methods extract user interest features by analyzing text, images, and other content posted by users, and then perform community division. However, these methods still have limitations in dealing with data sparsity and dynamic changes in interests.

Recently, advances in deep learning technology have led to significant improvements in community detection methods using graph neural networks. [10-12]. Graph neural networks can effectively learn high-order features of nodes and capture complex network structure information. At the same time, temporal convolutional networks have performed excellently in processing time-series data, providing the possibility for modeling dynamic changes in user interests. [13]

Based on this context, this paper presents an interest community detection algorithm that uses heterogeneous graph neural networks (HGNN) in conjunction with temporal convolutional networks (TCN). The algorithm first extracts interest sets from multi-user generated contents using the Latent Dirichlet Allocation (LDA) topic model, then uses HGNN to dynamically learn node features. Next, TCN is used to model the event propagation process and realize user interest tracking. Finally, interest communities are divided according to user labels to achieve the discovery of overlapping and non-overlapping communities.

The main contributions of this paper are as follows:

1) This paper proposes a novel community detection framework that combines HGNN and TCN, which can effectively handle data sparsity and dynamic changes in interests.

2) This paper designs an LDA-based interest extraction method that can extract potential interests from diverse user-generated content.

3) This paper proposes an interesting evolution tracking method based on TCN that can capture dynamic changes in user interests.

4) This paper conducts extensive experiments on the LFR benchmark network and four commonly used real network datasets to verify the effectiveness of the proposed algorithm.

The structure of this paper is arranged as follows: Section II reviews relevant literature and related work; Section III elaborates on the proposed algorithm; Section IV presents and analyzes experimental results; finally, Section V concludes the paper and outlines directions for future research.

2. Related work

The concept of community was first introduced by Newman and Girvan [1] in 2002, sparking significant interest in community detection among scholars. Through extensive research and experimentation, researchers have proposed numerous community detection algorithms. These community detection algorithms can be approximately classified into traditional community detection algorithms and those based on deep learning.

Current mainstream traditional community detection algorithms can be precisely summarized as hierarchical clustering [2,3], modularity optimization [4,5], label propagation [6-8], spectral clustering [9] and information theory [10] algorithms. Hierarchical clustering algorithm is an unsupervised machine learning method, which includes top-down split hierarchical clustering method and bottom-up agglomerative hierarchical clustering method. It divides the dataset into multiple hierarchical clusters to reflect the structure and similarity of the data. Modularity optimization algorithm originated from the modularity function proposed by Girvan and Newman [4], which measures the strength of network community structure. A higher modularity value indicates a more distinct community structure and better community quality.

Li et al. [5] combined modularity optimization with genetic algorithms to detect community structures using a defined local search operator. The Label Propagation Algorithm (LPA), first proposed by Raghavan et al. [6], updates unlabeled node labels based on the labeled node information. In this algorithm, each node is initially assigned a unique label, which is iteratively updated to adopt the most frequent label among its neighboring nodes until the system reaches convergence. Finally, nodes with the identical label are classified as the same community. Variants such as the parallel LPA [7] and the COPRA algorithm [8], which incorporates label membership, further improve the accuracy of this approach. While traditional community detection methods have proven effective for many network analysis tasks, they often struggle with scalability on large-scale networks and cannot easily incorporate node attributes or temporal dynamics. Additionally, most traditional algorithms require hand-crafted features and predefined similarity metrics, limiting their ability to capture complex,

hierarchical community structures. Deep learning approaches address these limitations by automatically learning effective representations and allowing for more flexible integration of multiple data modalities.

Deep learning-based community detection methods include deep neural network, deep graph embedding and graph neural networks (GNNs) [14,15]. Deep neural networks excel at constructing and capturing global relationships, with convolutional neural networks, autoencoders, and generative adversarial networks being the most commonly used models in community detection [16,17]. Deep graph embedding maps network nodes to a low-dimensional vector space while preserving the underlying structural properties and topological relationships of the original network. [12]. GNNs [18] iteratively aggregate feature information from local neighborhoods of the graph, enabling node information to propagate through the graph after transformation and aggregation. Bruna et al. [19] introduced a graph neural network-based approach for addressing the challenge of data-driven community detection. Zhang et al. [20] extended the Gumbel Softmax [21] method and proposed a new neural network community detection method.

However, for some new users or users with low activity, there are few user-generated contents on social networks. In the process of community division, it is difficult to directly extract user interests, and user interests will change with time. To solve these problems, this paper proposes a novel community detection algorithm that integrates HGNN and TCN to identify interest-based communities. Firstly, the LDA topic model is applied to extract user interests. Next, the HGNN is leveraged to capture high-order features. Then, the TCN is employed to track the evolution of user interests. Finally, interest communities are categorized based on user tags.

3. The HGNN-TC method

The modeling for interest community detection method, which is based on a HGNN and a TCN, is depicted in Fig. 1. Firstly, the LDA topic model is employed to analyze the content generated by the user and extract their potential interest. Secondly, the HGNN is utilized to dynamically learn and update node features to obtain high-order features. Subsequently, the TCN is applied to model the event propagation process and facilitate user interest tracking. Finally, the interest communities are categorize based on user tags.

3.1 User interest extraction

In this paper, the LDA document topic generation model is employed to generate user interests. The LDA model is a hierarchical three-layer Bayesian probabilistic model that incorporates documents, topics, and words as distinct generative layers. By training on document data through unsupervised learning, it

effectively extracts latent topic structures from large-scale document collections and textual corpora. The LDA model posits that documents consist of multiple topics, with each topic comprising a distinct distribution of words. The model represents each document as a multinomial distribution over topics, while each topic is characterized by a multinomial distribution over words. Through this hierarchical structure, it generates both topics and their associated word labels, effectively capturing underlying user interests. Fig. 2 shows the LDA topic generation model.

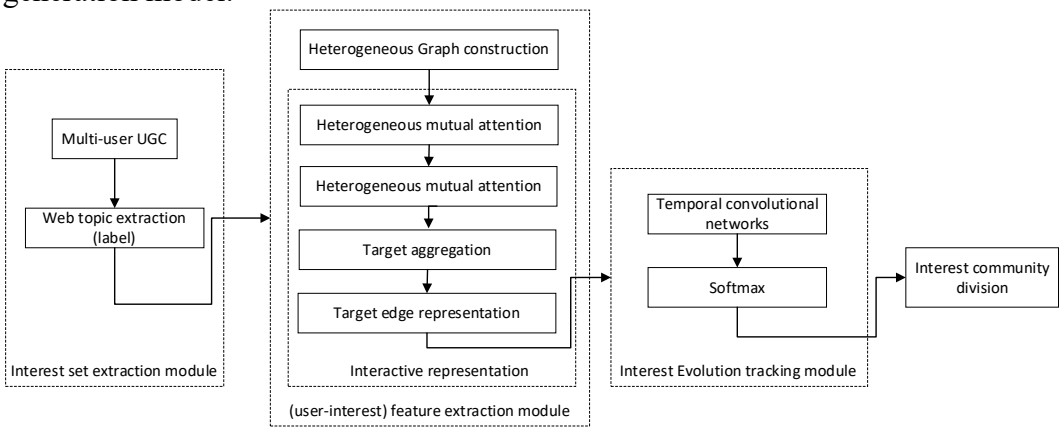


Fig. 1. Modeling of HGNN-TC

Suppose document D contains K topics and N words. The modeling process is as follows:

1) Sample the document D with Dirichlet distribution using parameter α to generate the document-topic distribution θ_m.

2) Sample from the document-topic polynomial distribution θ_m to generate topic $Z_{m,n}$ for the nth word in document D.

3) Sample the topic $Z_{m,n}$ with a Dirichlet distribution using parameter β to generate the topic-word distribution φ_k.

4) Sample from the topic-word multinomial distribution φ_k, and finally generate the word $\omega_{m,n}$.

From the above modeling process, it is evident that the LDA model adds a layer of Dirichlet prior to both the document-topic distribution and topic-word distribution, skillfully avoiding the overfitting problem caused by the increase of the user corpus. In addition, the simple and convenient Gibbs Sampling algorithm is used to obtain the topic posterior distribution of feature words through

continuous iterative sampling. Then the label set $K=\{k_1,k_2,\cdots,k_q\}$ of document D is obtained, where q denotes the number of labels.

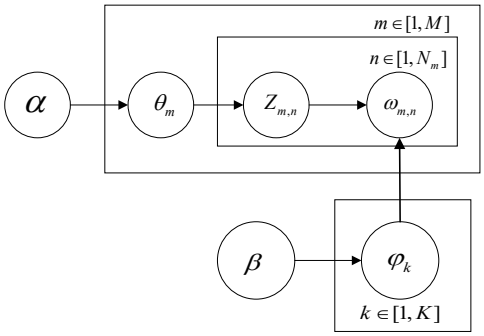


Fig. 2. LDA Topic Generation Mode

3.2 Heterogeneous graph neural network module

HGNN module can be divided into two main components: heterogeneous graph construction module and interaction representation module. The interaction representation module comprises four sub-modules: heterogeneous cross-attention, message passing, information aggregation and target edge representation. In this paper, user reaction records to events are restated as graph structures, where connections to interests are added to the response record and reformulated as heterogeneous graph data. Before using the HGNN to extract features, the sequence of user experience events needs to be converted into a heterogeneous graph data type.

3.2.1 Heterogeneous graph building blocks

Three entity types are selected as nodes: users, events and tags. $S=\{s_1,s_2,\cdots,s_n\}$ represents the set of users, where n represents the number of users. $E=\{e_1,e_2,\cdots,e_m\}$ represents the set of events with m denoting the number of events. $K=\{k_1,k_2,\cdots,k_q\}$ represents the set of labels, where q is the number of labels. Two types of edges: (Event-Label) and (User-Event) are contained.

The (Event-Label) edge, denoted as $(E-K)=\{e_1k_x,e_1k_{x+1},\cdots\}$, represents the relationship between events and labels, indicating the labels associated with each event. For example, e_1k_x indicates that the first event contains label x.

The (User-Event) edge, denoted as $(S-E)=\{(s_1e_x,r_{1,x}),(s_1e_{x+1},r_{1,x+1})\cdots\}$, represents the relationship between users and events, indicating the sequence of events experienced by users. For example, s_1e_{x+1} indicates that the first user

experienced the x+1-th event. And $r_{1,x+1}$ indicates whether the user is interested in the event, with 1 for interested and 0 for not interested. The PyTorch Geometry library is used to construct heterogeneous graphs, and the general process is shown in Fig. 3.

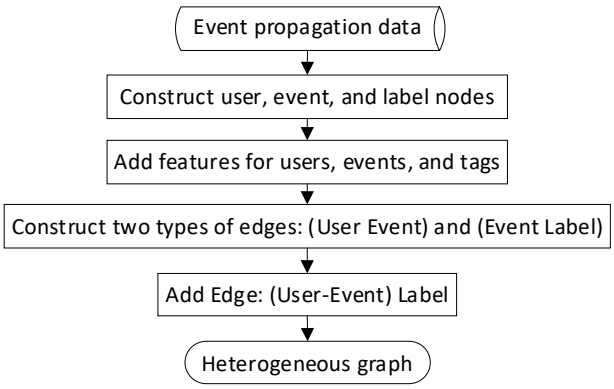


Fig. 3. Heterogeneous Graph Construction Process Diagram

Let's consider a small example with 2 users(s_1 and s_2), 3 events(e_1, e_2 and e_3) , and 3 labels (k_1 ="Music", k_2 ="Sports" , and k_3 = "Technology") . The example scenario are as follow: Event e_1 is a "Jazz Concert" tagged with label k_1 (Music). Event e_2 is a "Basketball Game" tagged with label k_2 (Sports). Event e_3 is a "Tech Conference" tagged with label k_3 (Technology). There are two users with different interaction patterns. User s_1 experienced events e_1 and e_3; he was interested in e_1 , but not in e_3. User s_2 experienced events e_2 and e_3, and was interested in both.

The (Event-Label) Edge (E - K) can be described as follow:

e_1k_1: The Jazz Concert is tagged with Music

e_2k_2 : The Basketball Game is tagged with Sports

$e_3\,k_3$: The Tech Conference is tagged with Technology

The (User-Event) Edges with Interest Indicators (S - E) can be described as follow:

s_1e_1, $r_{1,1} = s_1e_1$,1: User 1 experienced the Jazz Concert and was interested

s_1e_3, $r_{1,3}= s_1e_3$,0: User 1 experienced the Tech Conference but wasn't interested

s_2e_2, $r_{2,2}= s_2e_2$,1: User 2 experienced the Basketball Game and was interested

s_2e_3, $r_{2,3}= s_2e_3$,1: User 2 experienced the Tech Conference and was interested

3.2.2 Interaction representation module

Firstly, the node features within the heterogeneous graph are mapped to a high-dimensional space as inputs to the interaction representation module. Then, the Heterogeneous Graph Transformer (HGT) algorithm [22] is employed to obtain the features of each node in the heterogeneous graph. HGT is a specialized HGNN that adapts transformer architectures to heterogeneous graphs, emphasizing type-aware attention, temporal dynamics, and scalability. The specific steps of the HGT algorithm include heterogeneous mutual attention, message passing, target aggregation, and target edge representation. This paper takes the attention calculation, message passing, and target aggregation of event node *e* as an example. The update principles for user node *s* and label node *k* are analogous to those for event node *e*.

Step 1: Heterogeneous mutual attention

In heterogeneous graphs, the neighbors of a node can be different types of nodes, and the distribution and length of the representation vectors for these nodes may vary. And a node may be connected to neighboring nodes through multiple types of edges, each carrying distinct information. Heterogeneous graphs calculate the importance of neighboring nodes through heterogeneous mutual attention.

This paper considers event node *e* as the target node, with user node *s* and label node *k* as its neighboring nodes. The weights between them are calculated using a triplet relationship. (*s*, (*s*-*e*), *e*) represents node *s* pointing to node *e* through edge (*s*-*e*), and (*k*, (*k*-*e*), *e*) represents node *k* pointing to node *e* through edge (*k*-*e*). The target node e is mapped to a vector $Q(e)$, while its neighboring nodes *s* and *k* are mapped to vectors $K(s)$ and $K(k)$, respectively. The attention calculation process for the h-head of the edge (*s*-*e*) is as follows:

$$Attention_{HGT}(s-e) = Soft\max(\bigoplus_{i\in[1,h]} ATT-head^i(s-e)) \tag{1}$$

$$ATT-head^i(s-e) = (K^i(s)W^{ATT}_{(s-e)}Q^i(e)^T)\frac{\mu_{\langle s,(s-e),e\rangle}}{\sqrt{d}} \tag{2}$$

$$K^i(s) = K-Linear^i(H^{(l-1)}[s]) \tag{3}$$

$$Q^i(e) = Q-Linear^i(H^{(l-1)}[e]) \tag{4}$$

As shown in (3) and (4), each distinct node type in the heterogeneous graph corresponds to a unique linear projection to maximize the simulation of distribution differences. When calculating the output ATT-$head^i$(*s*-*e*) of the *i*-th head in multi-head attention, the input feature vectors $H(l\text{-}1)$ [*s*] and H(*l*-1) [*e*] of the *l*-th layer are first transformed into Key vectors and Query vectors through *K*-

Linear^i and Q-*Linear*^{i} according to the types of nodes *s* and *e*. Then the importance of node *s* to node *e* is calculated based on the weight matrix $W_{(s-e)}^{ATT}$ related to the type of edge. Finally, $\frac{\mu_{\langle s,(s-e),e\rangle}}{\sqrt{d}}$ is used to adjust the importance level through, where $\mu_{<s,(s-e),e>}$ represents a learnable parameter that quantifies the relative importance of various types of < node, edge, neighbor node > triples in heterogeneous graphs. $\frac{1}{\sqrt{d}}$ is used to balance the effect of vector dimensions on the result. After completing the single head attention calculation, all *h* attention heads are connected together.

Step 2: Message passing

Message passing involves transferring information from the source node to the target node, and its computation process runs in parallel with heterogeneous mutual attention computation. To mitigate the distributional disparities among distinct node types, the edge meta-relationship is integrated into the message passing process.

$$Message_{HGT}(s-e) = \bigoplus_{i\in[1,h]} MSG-head^i(s-e) \tag{5}$$

$$MSG-head^i(s-e) = M-Linear^i(H^{(l-1)}[s])W_{(s-e)}^{MSG} \tag{6}$$

MSG-*head*^{i}(*s*-*e*) represents the output of the *i*-th header during message passing. M-*Linear*^{i} maps the representation vector *H*(*l*-1)[*s*] of its *l*-1st layer to an information vector based on the type of node *s*. Since two types of nodes may be connected by multiple types of edges, the information vector also needs to be transformed by the matrix $W_{(s-e)}^{MSG}$.

Step 3: Information aggregation

After completing the above calculation and transmission, it is necessary to aggregate the neighbor nodes *s* and *k* with different feature distributions to the target node *e*. Then the updated vectors are connected by residuals after linear projection. Finally, by concatenating with the initial target node vector, the feature *H*(*l*)[*e*] of the target node is calculated by the *l*-th layer HGT is obtained. The calculation process is shown in (7) and (8).

When aggregating neighboring node information, the information vectors of neighboring nodes with different feature distributions are first weighted and summed, as shown in (7). Next A-*Linear* maps $H^{(l)}[e]$ to the representation space

corresponding to node type of *e*. Then we concatenate it with the representation vector $H(l\text{-}1)[e]$ of e in the l-1 layer to obtain the representation vector $H(l)[e]$ of the l-th layer. The calculation is shown in (8).

$$H^{(l)}[e] = \bigoplus_{\forall s \in N(e)} (Attention_{HGT}(s-e) \square Message_{HGT}(s-e)) \tag{7}$$

$$H^{(l)}[e] = \sigma(A - Linear H^{(l)}[e]) + H^{(l-1)}[e] \tag{8}$$

Step 4: Target edge representation

In order to more accurately represent the features of the edge (User-Label), this paper uses the updated user node feature $H(l)[s]$ and label feature $H(l)[k]$ to concatenate the feature $H[s\text{-}k]$ of the edge (User-Label).

$$H[s-k] = [H^{(l)}[s]; H^{(l)}[k]] \tag{9}$$

3.3 Interest evolution tracking module

The interest evolution tracking module uses a TCN to model the serialization of user label changes. TCN, an improvement over the convolutional neural network (CNN), contains three basic modules: causal convolution, dilated convolution and residual connection. Causal convolution ensures that the modeling process is sequential, whereby the output at any given time step is dependent solely on the current and preceding inputs. Dilated convolution increases the receptive field using a certain input interval, allowing each convolution to cover a larger range of information. Residual connection enables information to pass across layers, which solves the gradient problem in deep network training.

The TCN infrastructure is shown in Fig. 1, with a convolution kernel k=3 and expansion factor d= [1, 2, 4] [23]. The input of TCN is the feature of the (User-Label) edge updated by the interactive characterization module, and the output is the user interest state matrix P_t^s. After linear mapping and softmax function, a two-dimensional vector is obtained to represent the probabilities of user interest and disinterest in the label. The calculation formula is as follows:

$$P_t^s = Soft\max(Linear(P_t^s)) \tag{10}$$

3.4 Interest community division

Based on the calculation result P_t^s of the interest evolution tracking module, the label with the highest probability of user interest at time t is selected as the user label. Users sharing the identical label are aggregated into an interest

community set $C=\{C_1,C_2,...,C_q\}$, where q denotes the number of communities. At this point, the division of non-overlapping communities can be achieved.

By selecting the top n tags with the highest interest probability for users at time t as user labels and subsequently classifying users with the same labels into the same interest community, it is possible to achieve the division of overlapping communities.

4. Experiments

To evaluate the efficacy of our proposed algorithm, comparative analyses against four established methods (GCN [24], GAT [25], GraphSAGE [26], and LCFS [27]) are conducted on both the LFR benchmark network and four widely-adopted real-world networks (Karate, Football, Polbooks, and DBLP).

The performance of the proposed algorithm is evaluated using Precision, Recall, and their harmonic mean (F1-score). The true community membership of node v is denoted as C_i, while the community detection algorithm assigns it to community D_i, then

$$Precision=\frac{|D\cap C_i|}{|D|} \tag{11}$$

$$Recall=\frac{|D\cap C_i|}{|C_i|} \tag{12}$$

$$F1-score=2\times\frac{Precision\times Recall}{Precision+Recall} \tag{13}$$

A single metric, either Precision or Recall, cannot effectively reflect the performance of an algorithm. In some cases, an increase in Precision may reduce Recall, and vice versa. Therefore, this paper comprehensively adopted both metrics and their harmonic mean, the F1-score, to verify the effectiveness of the algorithm. The F1-score ranges from 0 to 1, where higher values indicate superior accuracy in identifying overlapping nodes by the algorithm.

4.1 Experiments on simulated network data sets

Both the node degree and community size in the LFR benchmark network follow a power-law distribution. The LFR benchmark is a commonly used tool for testing community discovery algorithms. The parameters of the LFR network generation program are presented in Table 1.

In this experiment, the parameters are configured as follows: n=5000, k=10, $kmax$=50, with the mu value ranging from [0.1, 0.5] in steps of 0.05, totaling 10 values, while other parameters are set by default. The parameter mu represents the mixed proportion, defined as the ratio of external links (connections

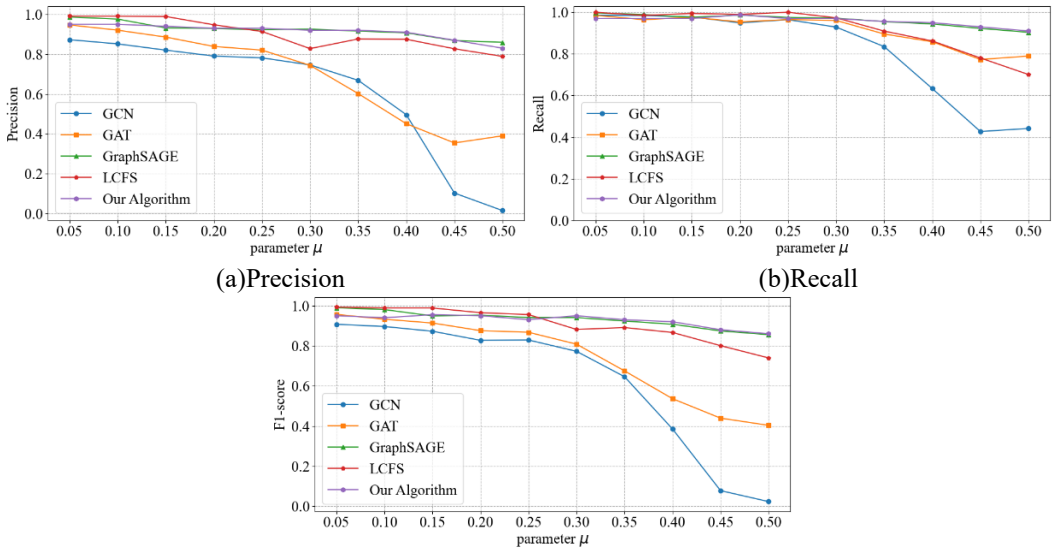
between a node and nodes outside its community) to the node's total degree. For example, when *mu* = 0.2, 80% of a node's connections are intra-community while 20% are inter-community links. The higher the *mu*, the greater the connection ratio between nodes in and outside the community, making community discovery more difficult. Conversely, a lower *mu* indicates a clearer community structure.

Table 1

The parameter of LFR

parameter	meaning
n	Number of nodes in the network
k	Average degree of a node
mu	Mixed parameter
k_{max}	Maximum degree of a node
C_{min}	Minimum community size
C_{max}	Maximum community size
t_1	Power rate distribution index of node degree
t_2	Power rate distribution index of community size

Each experiment is replicated 5 times per dataset, with the mean values reported as the final results and illustrated in Fig. 4.



(a)Precision (b)Recall

(c)F1-score

Fig. 4. Experimental results on the LRF dataset

When *mu*=0.05, although there are differences in the experimental results of the five algorithms, the values of F1 score, Precision, and Recall are all high, indicating good performance of the algorithms. As the mixing coefficient *mu* increases, the community structure becomes more complex, leading to varying

degrees of performance degradation across all five algorithms. Among them, GCN and GAT algorithms exhibit the most significant performance degradation. When *mu*>0.35, the performance of the GCN algorithm rapidly decreases, with F1 score and Precision approaching 0 and Recall values below 0.5.

The performance of the proposed algorithm, along with GraphSAGE and LCFS algorithms, remains relatively stable. They maintain good performance across different *mu* values, with F1-score, Precision, and Recall values consistently above 0.7. When *mu*<0.3, LCFS algorithm performs well. The proposed algorithm yields marginally lower F1-score, Precision, and Recall values compared to the LCFS algorithm. When *mu*=0.3, the Precision and Recall values of our algorithm are slightly lower than that of GraphSAGE. At *mu* = 0.5, our algorithm exhibits marginally lower Precision than GraphSAGE. After *mu*>0.3, the values of F1-score, Precision and Recall of this algorithm are generally the highest. It can be seen that our algorithm performs normally in simple networks but excels in complex networks.

As *mu* increases, our algorithm's performance decreases the slowest and fluctuates the least, with almost no noticeable decline in Precision, Recall, and F1-score. Tables 2 and 3 present the absolute decreases and decline rates, respectively, of Precision, Recall, and F1-score metrics across all five algorithms.

Table 2

The decline values of the Precision, Recall and F1-score

Algorithm	Precision	Recall	F1-score
GCN	0.86	0.54	0.88
GAT	0.56	0.20	0.55
GraphSAGE	0.13	0.09	0.13
LCFS	0.20	0.30	0.25
Our algorithm	0.12	0.06	0.09

Table 3

The decline rates of the Precision, Recall and F1-score

Algorithm	Precision	Recall	F1-score
GCN	98.4%	55.1%	97.4%
GAT	58.9%	19.9%	57.8%
GraphSAGE	12.9%	9.4%	13.6%
LCFS	20.4%	29.9%	25.5%
Our algorithm	12.6%	6.2%	9.5%

The minimal decline in performance metrics as μ increases demonstrates superior robustness to community structure blurring compared to baseline methods. This stability can be attributed to the complementary nature of our hybrid approach. While HGNN effectively captures complex heterogeneous relationships between users and content, TCN provides temporal resilience by

modeling interest evolution patterns that remain detectable even as community boundaries become less distinct.

4.2 Experiments on real network data sets

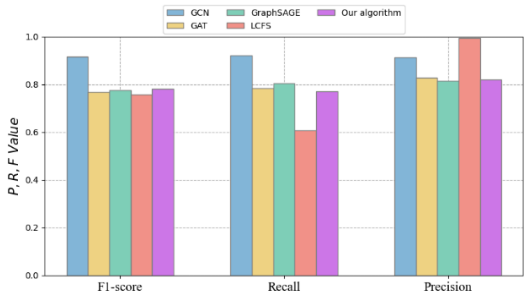
Comparative experiments were conducted on four widely recognized real-world datasets: Karate, Football, Polblogs, and DBLP. The data summaries for the four networks are provided in Table 4. The experimental results of the five algorithms on these four data sets are illustrated in Fig. 5.

Table 4

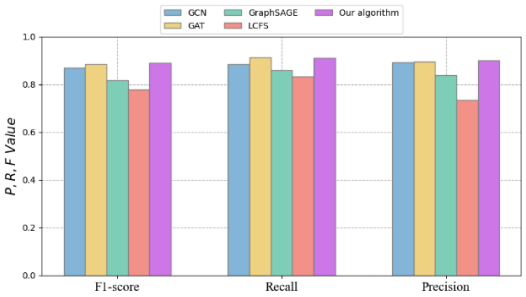
The information of four Data sets information

Data set	Nodes	Edges	Community
Karate	34	78	2
Football	115	616	12
Polblogs	1490	19090	2
DBLP	17725	105781	4

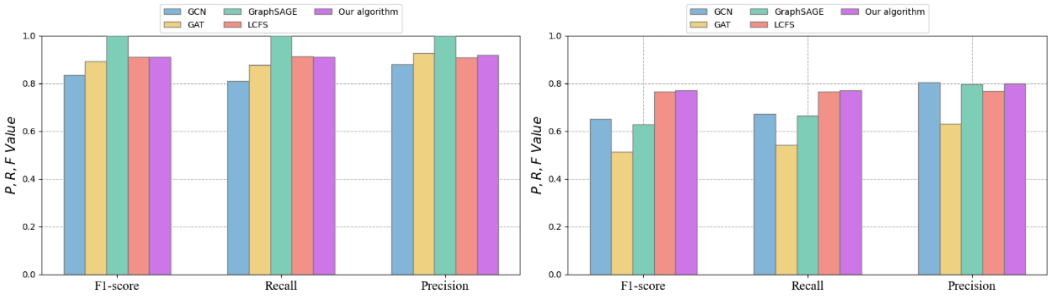
On the Karate dataset, the GCN algorithm achieves the highest F1-score and Recall values, while the LCFS algorithm has the highest Precision, nearly 1. Our algorithm, GraphSAGE, and GAT perform similarly. On the Football dataset, our algorithm and GAT algorithm perform best, with our algorithm having the highest F1-score and Precision values, and GAT algorithm having the highest Recall values. On the Polbooks dataset, the GraphSAGE algorithm has experienced overfitting. Our algorithm demonstrates comparable performance to LCFS, achieving similarly high values across all three metrics: F1-score, Precision, and Recall. On the DBLP dataset, our algorithm has the highest F1 score and Recall values, while the GCN algorithm has the highest Precision.



(a) Karate



(b) Football



(c) Polbooks (d) DBLP

Fig. 5. Experimental results on real network dataset

Our algorithm achieves the highest F1-score on the Football and DBLP datasets, slightly lower than LCFS on Polbooks dataset, and second only to GCN on Karate dataset. This demonstrates that our algorithm has good performance on both simple and complex datasets, with particularly strong performance on complex networks. Our algorithm can effectively balance Precision and Recall to make the community division more reasonable.

The mean and variance of the three performance indicators (F1-score, Precision, and Recall) for the five algorithms across the four data sets are shown in Table 5 and Table 6.

Table 5

The mean of the Precision, Recall and F1-score on the four data sets

	F1-score	Recall	Precision
GCN	0.82	0.82	0.87
GAT	0.76	0.78	0.82
GraphSAGE	0.81	0.83	0.86
LCFS	0.80	0.78	0.85
Our algorithm	0.84	0.84	0.86

Table 6

The variance of the Precision, Recall and F1-score on the four data sets

	F1-score	Recall	Precision
GCN	0.013	0.012	0.002
GAT	0.031	0.028	0.018
GraphSAGE	0.023	0.019	0.009
LCFS	0.005	0.017	0.015
Our algorithm	0.005	0.007	0.003

Our algorithm has the highest mean F1-score and Recall, and the second-highest mean Precision. The variance of our algorithm's F1-score and Recall is the highest, and the variance of its Precision is the second lowest, significantly lower than those of other algorithms.

The advantage of marginal performance stems from our integration of temporal dynamics and interest in tracking mechanisms absent in purely structural

approaches. This underscores the importance of incorporating time-evolving user interests when detecting communities in dynamic social networks.

5. Conclusion and future work

This paper proposes an interest community discovery algorithm based on the HGNN and TCN to address the challenges posed by "cold start users", specifically the difficulty in directly extracting user interests and tracking interest evolution. First, the LDA model is used to extract multi-user interests, forming an interest set. Next, a HGNN dynamically learns node features to obtain higher-order features. Then, a TCN is used to model the event propagation process, achieving user interest tracking. Finally, the interest communities are divided according to the user labels. This method can effectively discover both non-overlapping and overlapping communities.

While the proposed algorithm shows promising results, there are several directions for future research:

1) Scalability: As social networks continue to grow in size and complexity, further work is needed to optimize the algorithm's performance on large-scale networks (e.g., parallel computing, graph sampling).

2) Real-time processing: Developing techniques for real-time community detection and interest tracking could enhance the algorithm's applicability in dynamic social media environments (e.g., incremental learning procedures, efficient index structures).

3) Multi-modal data integration: Incorporating diverse types of user-generated content (e.g., images, videos) could provide richer insights into user interests and community structures.

4) Interpretability: Enhancing the interpretability of the model's decisions could provide valuable insights for network analysts and improve trust in the algorithm's outputs. We can develop visualization tools that highlight temporal patterns in interest evolution.

5) Privacy preservation: As community detection often involves sensitive user data, developing privacy-preserving techniques for interest extraction and community discovery is an important area for future work. We can develop privacy-aware interest representations that intentionally obscure individual-identifying information while preserving community-level patterns.

6) Cross-platform analysis: Extending the algorithm to handle data across diverse social media platforms would facilitate a holistic analysis of user interests and community structures across different online environments.

7) Adaptive learning: Developing mechanisms for the algorithm to adaptively adjust its parameters based on changing network dynamics could further improve its performance and robustness. We can design reinforcement

learning frameworks to optimize community detection strategies based on quality metrics.

In conclusion, this paper presents a significant step forward in addressing key challenges in community detection within social networks. By combining advanced machine learning techniques with a deep understanding of social network dynamics, our approach offers a powerful tool for elucidating significant community structures. As social networks continue to evolve and shape our digital interactions, the development of sophisticated community detection algorithms will remain a crucial area of research, with far-reaching implications for fields such as social science, marketing, and information dissemination.

REFERENCES

[1] M. Girvan and M. E. J. Newman, “Community structure in social and biological networks,” in Proc. Natl Acad, USA, 2002, pp. 7821–7826.

[2] A. Ashourvan, Q. K. Telesford, T. Verstynen, J. M. Vettel, and D. S. Bassett, “Multi-scale detection of hierarchical community architecture in structural and functional brain networks,” PLoS One, vol. 14, no. 5, pp. e0215520, May 2019.

[3] F. Wang, B. Zhang, and S. Chai, "Deep Auto - encoded Clustering Algorithm for Community Detection in Complex Networks," Chin. J. Electron., vol. 28, no. 3, pp. 489–496, May 2019.

[4] M. E. J. Newman, “Fast algorithm for detecting community structure in networks,” Phys. Rev. E, vol. 69, no. 6, pp. 066133-066137, Jun. 2004.

[5] S. Li, Y. Chen, H. Du, and M. W. Feldman, “A genetic algorithm with local search strategy for improved detection of community structure,” Complexity, vol. 15, no. 4, pp. 53–60, Mar. 2010.

[6] U. N. Raghavan, R. Albert, and S. Kumara, “Near linear time algorithm to detect community structures in large-scale networks,” Phys. Rev. E, vol. 76, no. 3, pp. 036106-036116, Sep. 2007.

[7] K. Guo, W. Guo, Y. Chen, Q. Qiu, and Zhang Qishan, “Community discovery by propagating local and global information based on the MapReduce model,” Inf. Sci., vol. 323, no. C, pp. 73–93, Dec. 2015.

[8] S. Gregory, “Finding overlapping communities in networks by label propagation,” New J. Phys., vol. 12, no. 10, pp. 103018-103043, Oct. 2010.

[9] L. Huang, R. Li, H. Chen, X. Gu, K. Wen, and Y. Li, “Detecting network communities using regularized spectral clustering algorithm,” Artif. Intell. Rev., vol. 41, no. 4, pp. 579–594, Mar. 2012.

[10] M. Rosvall and C. T. Bergstrom, “Maps of random walks on complex networks reveal community structure,” in Proc. Natl. Acad. Sci., USA, 2008, pp. 1118–1123.

[11] J. Yang, J. Mcauley, and J. Leskovec, “Community Detection in Networks with Node Attributes,” in Proc. ICDM, USA, 2013, pp. 1151–1156.

[12] S. Xue, J. Lu, and G. Zhang, “Cross-domain network representations,” Pattern Recognit., vol. 94, pp. 135–148, Oct. 2019.

[13] Z. Wang, J. Zheng, and F. Liu, “Improvement of continuous emotion recognition of temporal convolutional networks with incomplete labels,” IET Image Proc., vol. 18, no. 4, pp. 914–925, Nov. 2023

[14] L. Ruiz, N. T. Huang, and S. Villar, “A Spectral Analysis of Graph Neural Networks on Dense and Sparse Graphs,” in PROC. ICASSP, Seoul, Korea, 2024, pp. 9936–9940.
[15] J. Li, S. Lai, Z. Shuai, et al., “A comprehensive review of community detection in graphs,” Neurocomput., vol 600, no C, pp. 129169-128192, oct. 2024.
[16] B. L. Rosa, “Explaining Deep Neural Networks by Leveraging Intrinsic Methods,” Jul. 2024, arXiv:2407.12243.
[17] A. Radhakrishnan, M. Belkin and C. Uhler, “Wide and deep neural networks achieve consistency for classification”, in Proc. Natl. Acad. Sci. USA, 2023, pp. e2208779120-e2208779127.
[18] Y. Ma, S. Wang, C. C. Aggarwal, D. Yin, and J. Tang, “Multi-dimensional Graph Convolutional Networks,” in Proc. SDM, Canada, 2019, pp. 657–665.
[19] Z. Chen, X. Li and J. Bruna, “Community Detection with Graph Neural Networks,” presented at the 7th Int. Conf. Learning Representations (ICLR 2019), New Orleans, Louisiana, United States, May. 6-9, 2019.
[20] D. B. Acharya and H. Zhang, “Community Detection Clustering via Gumbel Softmax,” SN Comput. Sci., vol. 1, no. 5, pp. 262-272, Aug. 2020.
[21] B. W. Kernighan and S. Lin, “An Efficient Heuristic Procedure for Partitioning Graphs,” Bell Syst. Tech. J., vol. 49, no. 2, pp. 291–307, Feb. 1970.
[22] Z. Hu, Y. Dong, K. Wang, and Y. Sun, “Heterogeneous Graph Transformer,” in Proc. WWW '20, New York, NY, USA, 2020, pp. 2704–2710.
[23] W. ZHANG, H. WANG, and G. ZHU, “Temporal Convolutional Knowledge Tracing Method with Heterogeneous Graph Neural Network,” J. Chin. Comput. Syst., pp. 1–8, Sep. 2023.
[24] T. N. Kipf and M. Welling, “Semi-Supervised Classification with Graph Convolutional Networks,” presented at the 5nd Int. Conf. Learning Representations (ICLR 2017), Toulon, France, Apr. 24-26, 2017.
[25] P. Velikovi, G. Cucurull, A. Casanova, A. Romero, P. Lio and Y. Bengio, “Graph Attention Networks,” presented at the 6nd Int. Conf. Learning Representations (ICLR 2018), Vancouver, BC, Canada, May 2018.
[26] W. Hamilton, Z. Ying, and J. Leskovec, “Inductive Representation Learning on Large Graphs,” in Proc. NIPS, Long Beach, CA, USA, 2017, pp. 1024–1034.
[27] J. Liu, D. Wang, S. Feng, and Y. Zhang, “Local community discovery approach based on fuzzy similarity relation,” J. Software, vol. 31, no. 11, pp. 3481−3491, 2020.