Research Article

UserRBPM: User Retweet Behavior Prediction with Graph Representation Learning

Huihui Guo ¹, Li Yang ^{2,3} and Zeyu Liu ²

¹School of Cyber Engineering, Xidian University, 710126, China

²School of Computer Science and Technology, Xidian University, 710071, China

³Shaanxi Key Laboratory of Blockchain and Secure Computing, Xi'an, 710071, China

Correspondence should be addressed to Li Yang; yangli@xidian.edu.cn

Received 22 April 2021; Accepted 3 June 2021; Published 9 July 2021

Academic Editor: Lei Chen

Copyright © 2021 Huihui Guo et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Social and information networks such as Facebook, Twitter, and Weibo have become the main social platforms for the public to share and exchange information, where we can easily access friends' activities and in turn be influenced by them. Consequently, the analysis and modeling of user retweet behavior prediction have an important application value, such as information dissemination, public opinion monitoring, and product recommendation. Most of the existing solutions for user retweeting behavior prediction are usually based on network topology maps of information dissemination or designing various handcrafted rules to extract user-specific and network-specific features. However, these methods are very complex or heavily dependent on the knowledge of domain experts. Inspired by the successful use of neural networks in representation learning, we design a framework, UserRBPM, to explore potential driving factors and predictable signals in user retweet behavior. We use the graph embedding technology to extract the structural attributes of the ego network, consider the drivers of social influence from the spatial and temporal levels, and use graph convolutional networks and the graph attention mechanism to learn its potential social representation and predictive signals. Experimental results show that our proposed UserRBPM framework can significantly improve prediction performance and express social influence better than traditional feature engineering-based approaches.

1. Introduction

Due to their convenient capability to share real-time information, social media sites (e.g., Weibo, Facebook, and Twitter) have grown rapidly in recent years. They have become the main platforms for the public to share and exchange information and to a great extent meet the social needs of users. Under normal circumstances, online social networks will record a large amount of information generated by people through interactive activities, including various user behavior data. User behaviors (also called actions) in online social networks consist of posting messages, purchasing products, retweeting information, and establishing friendships. By analyzing the distribution and causality of these behaviors, we can evaluate the influence of the initiator and the communicator of the behavior, we can predict people's behaviors on social networks, and we can deepen our under-

standing of human social behavior [1, 2]. Till now, there is little doubt that the large amount of data generated by user interaction provides an opportunity to study user behavior patterns, and the analysis and modeling of retweet behavior prediction have become a research hotspot. In addition to analyzing the retweeting behavior itself, retweeting can also help with a variety of tasks such as information spreading prediction [3, 4], popularity prediction [5, 6], and precision marketing [7, 8].

Owing to the enormous usefulness of prediction, a variety of studies have been conducted on the task of automatic prediction in social networks. Previous researches investigated the problem of user retweet behavior prediction from different points of view. On the first approach, some researchers build retweet behavior prediction models through network topology maps of information dissemination. Matsubara et al. [9] studied the dynamics of information diffusion in

social media by extending an analysis model for information dissemination from the classic “Susceptible-Infected” (SI) model. Wang and Wang [10] proposed an improved SIR model, which used the mean field theory to study the dynamic behavior in uniform and heterogeneous network models. Their experiment showed that the existence of the network would influence information communication. This kind of research method studied retweeting behavior by modeling the propagation path of the message from a global perspective. The other approach is the machine learning method based on feature engineering. Liu et al. [11] proposed a retweeting behavior prediction model based on fuzzy theory and neural network algorithm, which can effectively predict the user retweeting behavior and dynamically perceive the changes in hotspot topics. This research method relies on the knowledge of domain experts, and the process of feature selection may take a long time. The methods above build user behavior prediction models from different perspectives. The main purpose is to collect user behavior data in social networks, cluster and label user behavior data, and then exploit machine learning models to predict the retweeting behavior. However, in many online applications, such as personalized recommendation [12, 13] and advertising [8], or personalized services [14], it is critical to effectively analyze the social influence of each individual and further predict the retweeting behavior of users.

In this paper, we focus on user-level social influence. We aim to predict the action statuses of the target user according to the action statuses of her near neighbors and her local structural information. For example, in social networks, a person’s behavior will be affected by her neighbors. As shown in Figure 1, for the central user u , if some friends (red node) around her posted a microblog and other friends (white node) did not post it, whether the action statuses of user u will be affected by the surrounding friends and forward this tweet can be regarded as a user retweeting behavior prediction problem. The social influence hidden behind the retweeting behavior not only depends on the number of active users, but may also be related to the local network structure formed by “active” users. The problem mentioned above are common in practical applications, such as presidential elections [15], innovation adoption [16], and e-commerce [17]. Therefore, it has inspired many research work on user-level influence models, most of which [18–20] consider complicated handcrafted features, which require extensive knowledge of specific domains.

In recent years, graph convolution networks (GCN) [21, 22] are the best choice for graph data learning tasks. Inspired by the successful application of neural networks [23] in representation learning [24, 25], we designed an end-to-end framework UserRBPM to explore potential driving factors and predictive signals in user retweeting behaviors. We expect deep learning frameworks to have better expressive capability and prediction performance. The designed solution is to represent both influence driving factors and network structures into a latent space, and then use graph neural networks to effectively extract spatial features for learning, and further construct a user retweet behavior prediction model. To predict the status of a target user u , we first

FIGURE 1: A motivating example of user retweet behavior prediction.

sample her k -order local neighbors through random walks with restart. After obtaining the r -ego network as shown in Figure 1, we leverage both graph convolution and attention mechanism to learn latent predictive signals. We demonstrate the effectiveness and efficiency of our proposed framework on Weibo social networks. We compare UserRBPM with several conventional methods, and experiment results show that the UserRBPM framework can significantly improve the prediction performance. The main contributions of this work can be summarized as follows:

- (i) We designed an end-to-end learning framework UserRBPM to explore potential driving factors and predictive signals in user retweeting behaviors
- (ii) We convert the retweeting behavior prediction into a binary graph classification, which is more operable and comprehensible
- (iii) Experiment results demonstrate that the UserRBPM framework can achieve better prediction performance than existing methods

The rest of this paper is organized as follows. Section 2 summarizes related work. Section 3 formulates the user retweet behavior prediction problem. We detail the proposed framework in Section 4. In Section 5 and Section 6, we conduct extensive experiments and analyze the results. Finally, we conclude our work in Section 7.

2. Related Work

2.1. User Retweet Behavior Prediction. Many studies on user retweet behavior in social networks are based on the analysis and modeling of the dynamics in the process of information dissemination. Currently, researches on user behavior prediction in social networks take primarily two approaches. On the first approach, Ota et al. [26] constructed the user topology network based on the user’s following relationship and discovered users who retweeted many tweets by overlapping propagation paths of retweeting. Yuan et al. [27] investigated the dynamics of friend relationships through online social interaction and proposed a model to predict repliers or retweeters according to a particular tweet posted at a certain time in online social networks. Tang et al. [28] studied the conformity phenomenon of user behavior in social networks and proposed a probabilistic model called Confluence

to predict user behavior. This model can distinguish and quantify the effects of the different types of conformity. Zhang et al. [29] proposed three metrics, namely, user enthusiasm, user engine, and user duration, to describe the user retweet behavior in the message spreading process and studied the relationship between these three metrics and the influence obtained by the user retweet behavior.

The other approach is the machine learning method based on feature engineering, which solved the problem of user behavior analysis and prediction by manually formulating rules to extract the basic features of users and network structural features. Luo et al. [30] explored features such as followers' status, retweet history, followers' interests, and followers' active time with a learning-to-rank framework to discover who would retweet a tweet poster on Twitter. Zhang et al. [18] analyzed the influence of the number of active neighbors of a user on retweeting behavior, proposed two instantiation functions based on structural diversity and pairwise influence, and applied a classifier based on logistic regression to predict users' retweet behaviors. Jiang et al. [19] pointed out that the retweeting prediction is a sing-type setting problem. By analyzing the basic influence factors of retweet behavior in Weibo, the sing-type collaborative filtering method is used to measure users' personal preferences and social influence to predict retweet behavior.

Recently, there have been efforts to detect those global patterns using deep learning. Li et al. [31] proposed an end-to-end predictor that incorporated recurrent neural network (RNN) and representation learning to infer the cascade size. This method significantly improved the performance of cascade prediction. Zhang et al. [32] proposed a novel attention-based deep neural network to obtain the user's attention interests from an attention-based neural network. Wang et al. [33] transformed the social influence prediction problem into a neural network multilabel classification problem and proposed the NNMLInf social influence prediction model. The experimental results showed that the node2vec method is more effective than the traditional manual feature extraction method in obtaining representative features of the network structure.

2.2. Graph Representation Learning. Graph representation learning has emerged as a powerful technique for solving real-world problems. Various downstream graph learning tasks have benefited from its recent developments, such as node classification [34], similarity search [35], and graph classification [36, 37]. Network embedding is a bridge connecting the original data of the network and network application tasks. It is aimed at representing the nodes in the network as low-dimensional, real-valued, and dense vectors. The resulting vectors can be represented and reasoned in the vector space. Therefore, the primary challenge in this field is to find a way to represent or encode the structure of graphs so that they can be easily exploited by machine learning models.

Traditional machine learning approaches relied on user-defined heuristics to extract features encoding structural information about a graph (e.g., degree statistics or kernel functions). However, recent years have seen a surge in

approaches for automatically learning to encode a graph structure into low-dimensional embedding using techniques based on deep learning and nonlinear dimension reduction. Chen et al. [38] exploited graph attention networks (GAT) to learn user node representation by spreading information in heterogeneous graphs and then leveraged limited labels of users to build end-to-end semisupervised user profiling predictor. Zhang et al. [25] introduced the problem of heterogeneous graph representation learning and proposed a heterogeneous graph neural networks model HetGNN. Extensive experiments on various graph mining tasks, i.e., link prediction, recommendation, and node classification, demonstrated that HetGNN can outperform state-of-the-art methods. Fan et al. [39] first provided a principled approach to jointly capture interaction and opinions in the user-item graph, and then presented a Graph Network model (GraphRec) to model social recommendation for rating prediction.

3. Problem Formulation

In this section, we introduce necessary definitions and then formulate the problem of user retweet behavior prediction.

Definition 1 (ego network). The ego network model is one of the important tools for studying human social behavior and social networks. Compared with the global network version, the research version of the ego network pays more attention to individual users, and it is in line with the need for personalized services in actual application systems. The research version of this paper can also be extended to other scenarios that include network relationships.

(*r-neighbors*) Let $G = (V, E)$ denote a social network, where V is a set of users' nodes and $E \subseteq V \times V$ is a set of relationships between users. We use $v_i \in V$ to represent a user and $e_{ij} \in E$ to represent a relationship between v_i and v_j . In this work, we consider undirected relationships. For a user u , its r -neighbors' nodes are defined as $\Gamma_u^r = \{v : d(u, v) \leq r\}$, where $d(u, v)$ is the shortest path distance (in terms of the number of hops) between u and v in the network G , and $r \geq 1$ is a tunable integer parameter to control the scale of the ego network.

(*r-ego network*) The r -ego network of user u is the sub-network induced by Γ_u^r , denoted by G_u^r .

Definition 2 (social action). In sociology, social action is an act that takes into account the actions and reactions of individuals. Users in social networks perform social actions, such as retweeting behaviors and citation behaviors. At each time stamp t , we observe a binary action status of user u , $s_u^t \in \{0, 1\}$, where $s_u^t = 1$ indicates that user u has performed this action before or on the timestamp t , and $s_u^t = 0$ indicates that the user has not performed this action yet.

Definition 3 (historical behavior). Let L denote a stream of action logs, where each log entry $l \in L$ is a triple (v, a, t) , representing user $u \in V$ performed action $a \in A$ before or on the timestamp t . Here, A is a set of action types, $A = \{(v_i, a,$

$t) \mid t \in \psi, v_i \in V\}$, and ψ denotes the time scope of historical behavior. For example, a retweeting behavior is an action in Twitter, and a citation is an action in academic social networks.

In this paper, our research motivation of the user retweeting behavior prediction problem can be vividly illustrated through the example shown in Figure 1. For a user u in her 2-ego network (i.e., $r = 2$), if some users retweet the message m before or on the timestamp t , they are considered to be active. We can observe the action statuses of u 's neighbors, such as $s_{v_1}^t = 1$, $s_{v_2}^t = 1$, and $s_{v_5}^t = 0$. Moreover, the set of active neighbors of user u is represented by $\psi_u^t = \{v_1, v_2, v_3, v_4, v_6\}$. As shown in Figure 1, we study whether the action statuses of user u will be influenced by the surrounding friends and forward this message. Next, we will formalize the problem of user retweet behavior prediction.

Problem 1 (user retweet behavior prediction, [18]). User retweet behavior prediction models the probability of u 's action states conditioned on her r -ego network and the action states of her r -neighbors. More formally, given G_u^r and $S_u^t = \{s_v^t : v \in \Gamma_u^r \setminus \{u\}\}$, it can be concluded that the user retweet behavior prediction formula of user u after a given time interval Δt is as follows:

$$A_v = P(s_u^{t+\Delta t} \mid G_u^r, S_u^t). \quad (1)$$

Practically, A_v denotes the predicted social action status of user u . Suppose we have N instances, and each instance is a 3-tuple (u, a, t) , where u is a user, a is a social action, and t is a timestamp. For such a 3-tuple (u, a, t) , we also know u 's r -ego network G_u^r , the action states of u 's r -neighbors S_u^t , and u 's future action states at $t + \Delta t$, i.e., $s_u^{t+\Delta t}$. We then formulate user retweet behavior prediction as a binary graph classification problem which can be solved by minimizing the following negative log-likelihood objective w.r.t. model parameter θ :

$$L(\theta) = - \sum_{i=1}^n \log (P_{\theta}(s_u^{t+\Delta t} \mid G_u^r, S_u^t)). \quad (2)$$

4. Model Framework

In this paper, we formally propose the UserRBPM to address the problem of user retweet behavior prediction. The framework is based on graph neural networks to parameterize the probability in equation (2) and automatically detect the potential driving factors and predictive signals of user retweet behavior prediction. As shown in Figure 2, UserRBPM consists of the pretrained network embedding layer, the input layer, the GCN/GAT layer, and the output layer.

4.1. Sampling Near Neighbors. Given a user u , the most straightforward way to extract its r -ego network is to perform a Breadth-First-Search (BFS) starting from user u . However, for different users, the r -ego network scale (regarding the number of nodes) may vary greatly. Meanwhile, the size of

user u 's r -ego network can be very large due to the small-world property in social networks [40]. In real-world application scenarios, when sampling neighbor nodes of an ego user node, the problem that may arise is that each node has a different number of neighbors. Specifically, because of the small-world phenomenon in social networks, the size of user u 's r -ego network may be relatively very large or small. In addition, these different sizes of data are not suitable for most deep learning models.

In order to address the above problem, we select to perform random walk with restart (RWR) [41] from the original r -ego network to fix the sample size. Inspired by [42, 43] which suggested that people are more susceptible to be influenced by active neighbors than inactive ones, we start a random walk on from user u or its active neighbors. The walk iteratively travels to its neighborhood with a probability proportional to the weight of each edge. Besides, the walk returns back to the starting vertex u with a positive probability at each step. In this way, a fixed number of vertices can be collected, denoted by $\Gamma_u^{\rightarrow r}$ with $|\Gamma_u^{\rightarrow r}| = n$. We then regard the subnetwork $G_u^{\rightarrow r}$ induced by $\Gamma_u^{\rightarrow r}$ as a proxy of the r -ego network G_u^r , and denote $S_u^{\rightarrow t} = \{s_v^t : v \in \Gamma_u^{\rightarrow t} \setminus \{u\}\}$ to be the action statuses of u 's sampled neighbors.

When we use RWR, the starting node can be an ego user or its active neighbors. Firstly, the purpose of setting as described above is to make the starting node in the sequence obtained by walking as much as possible to keep in touch with surrounding neighbors, instead of being relatively single. Thus, it can further support that people are more susceptible to active neighbors. Accordingly, the random walk with a restart strategy can meet this requirement. Secondly, the starting nodes include ego users and active neighbors. This setting allows ego users and active neighbors to participate in the next embedding process as much as possible, which also shows that active neighbors will affect its surrounding nodes.

4.2. Graph Neural Network Model. We design an effective graph neural network model to incorporate both the structural properties in $G_u^{\rightarrow r}$ and action statuses in $S_u^{\rightarrow r}$, learn a hidden embedding vector for each ego user, then use it to predict the action statuses $S_u^{t+\Delta t}$ of the ego user in the next period of time. As shown in Figure 2, the graph neural network model includes the embedding layer, instance normalization layers, the input layer, the graph neural network layer, and the output layer.

4.2.1. Embedding Layer. Representation learning [44, 45] has been a hot topic in the academic community in recent years. In the context of graph mining, there have been many studies on graph representation learning. For graph structure data such as social networks, we want to learn the users' social representation from users' relationship network data, that is, our main purpose is to discover the network structural properties and encode them into low-dimensional latent space. More formally, network embedding learns an embedding matrix $X \in \mathbb{R}^{d \times |V|}$, with each column corresponding to the representation of a vertex (user) in network G . In our scheme, we learn a low-dimensional dense real number

FIGURE 2: Our proposed framework of UserRBPM (User Retweet Behavior Prediction Model).

vector $x_v \in R^d$ for each node v in the network, where $d \ll N$. The process of network representation learning can be unsupervised or semisupervised.

In social networks, when considering the structural information, we can take the triadic closure patterns with strong ties as an example [46]. As shown in Figure 3, there will be such a case: the figure on the left contains a triadic closure. For the green node, it is equivalent to a different tree structure on the right (there is no triadic closure) after two neighborhood aggregations, which ignore the structural information of triadic closure. Therefore, there is a need for a method of graph representation learning that can adapt to different local structures.

In our work, we utilize the GraLSP model [47] for graph representation learning, which explicitly incorporates local structural patterns into the neighborhood aggregation through random anonymous walks. Specifically, the framework captures the local structural patterns via random anonymous walks, and then these walk sequences are fed into the feature aggregation, where various mechanisms are designed to address the impact of structural features, including adaptive receptive radius, attention, and amplification. In addition, GraLSP can capture similarities between structures and are optimized jointly with near objectives of nodes. The process of the GraLSP model for graph representation learning is shown in Figure 4. In the case of making full use of the structural model, GraLSP can outperform competitors in various prediction tasks in multiple datasets.

4.2.2. Instance Normalization Layer. In the training process of the UserRBPM model, we applied *Instance Normalization* (IN) [48] to prevent overfitting, which is a regularization technique that loosens the model and allows for greater generalization. And for such tasks that focus on each sample, the information from each sample is very important. Therefore, we adopt such a technique in the task of retweet behavior prediction. After original data is normalized, the indicators are between $[0, 1]$, which is suitable for comprehensive comparative analysis. Furthermore, it helps to speed up learning and also reduces overfitting.

As illustrated in Figure 2(c), for each user $v \in \Gamma_u \xrightarrow{r}$ after retrieving her representation x_v from the embedding layer,

the instance normalization y_v is given by

$$y_{vd} = \frac{x_{vd} - \mu_d}{\sqrt{\sigma_d^2 + \varepsilon}}, \quad (3)$$

for each embedding dimension $d = 1, 2, \dots, D$, where

$$\mu_d = \frac{1}{n} \sum_{v \in \Gamma_u \xrightarrow{r}} x_{vd}, \quad (4)$$

$$\sigma_d^2 = \frac{1}{n} \sum_{v \in \Gamma_u \xrightarrow{d}} (x_{vd} - \mu_d)^2. \quad (5)$$

Here, μ_d and σ_d are the mean and variance, respectively, and ε is a small number for numerical stability. Intuitively, such normalization can remove instance-specific mean and variance, which encourages the downstream model to focus on relative positions of users in latent embedding space rather than their absolute positions. As we will see later in Section 6, instance normalization can indeed help avoid overfitting during training.

4.2.3. Input Layer. As claimed in Figure 2(d), the input layer constructs a feature vector for each user. The feature vector considered in our work consists of three parts: (1) The normalized low-dimensional embedding comes from the upstream instance normalization layer. (2) Two binary variables are also considered: the first variable represents the user's action statuses and the second variable represents whether the user is an ego user. (3) The input layer also includes other personalized vertex features, such as spatial-level features (e.g., social roles) and temporal-level features (e.g., similarity, exposure, and retweet rate.)

4.2.4. GCN Layer. Social networks can be regarded as graph structure data, where users represent nodes and the connection relationships between users represent edges in the graph. The characteristics of users can no longer fully represent all the information of the individual. It may be due to the loss of errors in the data collection process or the disguise of some individuals, causing some important features to be biased in the training process. Thus, we need the information of its

FIGURE 3: Computational tree of a triadic closure graph.

FIGURE 4: Overview of the GraLSP model.

neighbor nodes to supplement the information of the current node. In this way, we can get more complete information than a single individual characteristic. Therefore, how to combine the characteristics of neighbors with the current node is a critical part of its realization.

The recently developed GCN is a successful attempt to generalize the convolutional neural networks used in Euclidean space to graph structure data modeling. The GCN model naturally integrates the connection mode and feature attributes of graph structure data, and it is much better than many state-of-the-art methods on benchmarks. GCN is a semisupervised learning algorithm for graph structure data, which can effectively extract spatial features for machine learning on such a network topology. Simultaneously, it can perform end-to-end learning of node feature and structure information, which is one of the best choices for graph data learning tasks at present.

Suppose an undirected graph has n nodes, each node has d -dimensional features, the adjacency matrix of the graph is denoted as A , and the matrix composed of all node features is denoted as $\mathbf{X}$, where $\mathbf{X} = [x_1, x_2, \dots, x_n]^T \in \mathbf{R}^{n \times d}$ and $x_i \in \mathbf{R}^d$ is the d -dimensional feature vector of node i . If the labels of a set of nodes are given, our goal is to predict the labels of the remaining nodes. Thus, for the GCN network, the input is a node feature matrix $\mathbf{X} \in \mathbf{R}^{n \times d}$ and the propagation between layers can be expressed as follows:

$$\mathbf{H}^{l+1} = \sigma(\mathbf{L}(G)\mathbf{H}^l\mathbf{W}^l), \quad (6)$$

where $\mathbf{H}^l$ is the l th activation matrix, $\mathbf{W}^l$ is the trainable weight matrix of the l th layer, and this layer functions as a feature map. σ is a nonlinear activation function, $\mathbf{L}(G)$ is a $n \times n$ matrix that can capture the structure information of graph G , and GCN uses symmetric normalization to perform aggregation operations. The conventional graph convolution operation is as follows:

$$\mathbf{L}(G) = \mathbf{D}^{-1/2} \tilde{\mathbf{A}} \mathbf{D}^{-1/2}, \quad (7)$$

where $\mathbf{A} = [a_{ij}] \in \mathbf{R}^{n \times n}$ is a nonnegative adjacency matrix, $\tilde{\mathbf{A}} = \mathbf{A} + \mathbf{I}$, $\mathbf{I}$ is the identity matrix, $\mathbf{D} = \text{diag}(d_1, d_2, \dots, d_n)$ is the degree matrix of $\mathbf{A}$, $d_i = \sum_j a_{ij}$ is the degree of node i .

We can divide the above learning process into three parts. The first part is transformation: transform and learn the current node features. The second part is aggregation: aggregate the features of neighboring nodes to get the new features. The third part is activation: use an activation function to increase nonlinearity.

4.2.5. GAT Layer. Essentially, both GCN and GAT are aggregation operations that aggregate the characteristics of neighbor nodes into the central node. GCN uses the Laplacian matrix to perform graph convolution operations, while GAT introduces the attention mechanism into GCN, which can add weight to the influence of neighboring nodes, thereby differentiating the influence of neighboring nodes. GAT assigns different weights to each node, paying attention to those nodes with greater effects, while ignoring some nodes with smaller effects. To a certain extent, the performance ability of GAT will be stronger, because the correlation between node features will be better integrated into the model.

4.2.6. Output Layer. In the output layer, each node corresponds to a two-dimensional representation, which is used to represent the behavior state (retweet/unretweet, cite/uncite, etc.). The calculation process is shown in equation (8). By comparing the representation of the ego user with ground truth, we then optimize the log-likelihood loss:

$$\hat{Y} = \log_{\text{softmax}}(\mathbf{H}'\Theta + \mathbf{b}), \quad (8)$$

where Θ is a weight matrix, $\mathbf{b}$ represents the bias term.

5. Experiment Setup

In this section, we first introduce the construction process of the dataset and the experimental hypothesis. Then, we

present the existing representative methods and evaluation metrics. Finally, we introduce the implementation details of the UserRBPM framework.

5.1. Dataset Presentation and Processing

5.1.1. The Presentation of Raw Datasets. We use real-world datasets to quantitatively and qualitatively evaluate the proposed UserRBPM framework. We used the Weibo dataset in the work of Zhang et al. and Qiu et al. [18, 49], and Ulyanov et al. [48] also used the Weibo dataset in their work, and then we performed data preprocessing according to our research question. The microblogging network used in our research work is used to crawl data from Sina Weibo, similar to Twitter social media, which is a broadcast-style social network platform that shares brief real-time information through the follow mechanism. The follow mechanism of Weibo is divided into one-way following and mutual following. Particularly, when user u_1 follows user u_2 , u_2 's activities (such as tweet and retweet) will be visible to u_1 . User u_1 can choose to tweet or retweet user u_2 . User u_1 is called the follower of user u_2 , and user u_2 is called the followee of user u_1 . The dataset was crawled in the following ways. To start with, 100 users are randomly selected as seed users, and then user information of their followers and followees was collected. A total of about 1.8 million users and 300 million social relationships were obtained during the crawling process. A total of 1 billion microblogs were generated in this process, and all user profiles were also crawled which contain the name, gender, verification status, #bifollowing, #followers, #followees, and #microblogs. Besides that, to protect the privacy of users, we have desensitized the user id. The statistical information of raw data is shown in Table 1.

5.1.2. The Assumption of Experiments. The main problem studied in our paper is as follows: when a certain microblog is visible to a certain user, we predict whether the user will retweet the microblog within a certain period, which is a supervised binary classification problem. The experiments in our work are based on the following assumption:

- (i) The assumption of visibility: as long as a user sends (an original post or retweeted) a microblog, all his followers will see the microblog
- (ii) We aim to predict the action status of users at a specific time
- (iii) The assumption of timeliness: if the retweeting time of a user is more than 72 hours from the time that a microblog was first sent out, the sample will be discarded
- (iv) For a microblog, only when a user has active neighbors, that is, the user has followed the original creator or his followers have retweeted the microblog, we only consider whether the user will retweet this microblog

- (v) For a microblog, if a user has appeared as a positive sample at time t , we will not predict the action status after time t

Specifically, the assumption of Weibo Visibility is set because, with more than 100 million daily active users of Sina Weibo, each user receives a large amount of information every day. If a user follows a large number of users, some microblogs will likely be overwhelmed by other messages before the user can decide whether to retweet them.

We give a restriction on whether a microblog will be retweeted at a certain time. Because for a user and a microblog, the user may not retweet the microblog when he saw it for the first time. At this time, we generate a negative sample. When he retweeted the microblog the second time he saw it, we create a positive sample. In other words, users have different action statuses in different periods. For users who did not retweet, they were regarded as negative samples since the first time they saw a microblog. If the status of each time is retained and predicted, there will be many negative samples generated. Therefore, we give the second assumption. Besides, as Zhang et al. [18] pointed out that for most microblogs, 72 hours after their first posting, the number of retweets drops dramatically, and there will be almost no user retweeting. So, we set up the third assumption.

Sina Weibo does not restrict users from retweeting messages sent by users who have not followed them in real application scenarios. There are often phenomena that users actively search for microblogs and retweet them or retweet popular microblogs. This situation is beyond the scope of our work, so we set up the fourth assumption. The last assumption is that once a user has a retweeting behavior, it will be regarded as a positive sample at every subsequent time point. Repeated predictions in the experiment are of little significance.

5.1.3. The Generation of Samples. In retweeting behavior prediction, since we can directly learn from the microblogs' record which users have retweeted the microblogs, the extraction process of positive samples is relatively simple. Thus, for a user v who is affected by others, he performs a social action at a certain timestamp t , then we generate a positive sample. Compared with the extraction of positive samples, it is impossible to directly know from the microblogs' records which users saw the message but did not retweet the microblogs. Therefore, the extraction method of negative samples is much more complicated. Our work is based on the assumption of visibility. We suppose that after a user posts a microblog, all of her followers can see the microblog. If someone saw the microblog but did not retweet it, we create a negative sample.

Through the analysis of the experimental data, we found that about 1.25% are original microblogs, most of which were retweeted microblogs, and 85.39% of the microblogs have been retweeted at least ten times, verifying that the retweeting behavior of users is pervasive. However, for our research scenarios, in the process of solving research problems, there are two data imbalance problems in our dataset. The first one comes from the number of active neighbors. As Zhang et al.

TABLE 1: Statistics of raw data.

Dataset	#users	#followrelationship	#original microblogs	#retweet	#ego users
Weibo	1,776,950	308,489,736	300,000	23,755,810	779,164

[18] observed, structural features are significantly related to user retweeting behavior when the ego user has a relatively large number of active neighbors. However, in most social influence datasets, although retweeting behavior is ubiquitous, the ratio between the number of active neighbors and the number of inactive neighbors is not balanced. For example, in the Weibo dataset, 80% of users have only one active neighbor and users with more than 3 active neighbors account for only 8.57%. Therefore, when we train our model on such an imbalanced dataset, the model will be controlled by observation samples with few active neighbors. To address the issues caused by data imbalance and to illustrate the superiority of our proposed model in capturing local structural information, we established a balanced subdataset *Edata* (as shown in Table 2) for fair data analysis and a further training-test scheme. Specifically, we filter out samples in which the followers or followees did not have Weibo content. Besides, we only considered samples in which ego users have at least 3 active neighbors.

Imbalanced labels are the second problem. For instance, in our Weibo data set, the ratio between positive and negative instances is about 1 : 300. Normally, the model is trained to optimize the overall accuracy, and the weights of different types of misclassification are the same when calculating the overall error. As a result, the trained model tends to judge samples belonging to the minority classification as the samples of the majority classification. Moreover, the generalization ability of the model is poor and the minority classification cannot be accurately judged. Our goal is to find out those users who will retweet, rather than pay attention to those who do not retweet. Therefore, the datasets need to be balanced.

To address the above problem, the most direct way is to select a relatively balanced dataset, that is, set the ratio of positive samples and negative samples to 1 : 3. In addition, we also used the global random downsampling method and microblog granularity-based downsampling method to process imbalanced datasets. Among them, when we use the global random downsampling method, the number of microblogs involved in the negative samples in the obtained dataset is small, and there is a case where only positive samples of the same microblog are not sampled to their corresponding negative samples. The downsampling method based on microblog granularity can try its best to ensure that the number of positive and negative samples of the same microblog is also the same. The data statistics of the balanced sample set are shown in Table 3. To visually verify from the results that the downsampling strategy adopted in our work is more suitable for our research scenarios, we conducted a comparative analysis in Section 6.

5.1.4. The Features of Our Design. We made detailed data observations and analyzed how the characteristics of users

at the spatial and temporal levels influence users' retweeting behavior in addition to the structural attributes of social networks. To visualize the observation results, we design several examples of statistical information, which, respectively, represent spatial-level features and temporal-level features. These characteristics can be regarded as user node features. In our work, the spatial-level features are specifically analyzed in terms of social roles. We studied the influence of social roles played by different users on the prediction performance of retweeting behavior. Inspired by the previous research work of Wu et al. [50] and Yang et al. [51], we divide users into three groups according to their network attributes: opinion leaders (OpnLdr), structural hole spanners (StrHole), and ordinary users (OrdUtr). Specifically, we consider that 5% of users with the highest PageRank score are opinion leaders, 5% of users with the lowest Burts Constraint score are structural hole spanners, and the rest are ordinary users. A detailed analysis of users' social roles and social behaviors is shown in Table 4. For the temporal-level feature, we mainly analyzed the content of the messages posted by users, and we defined the following features:

- (i) Similarity: the TF-IDF similarity between ego user's and followees' post content within a month
- (ii) Exposure: the number of microblogs posted by followees within a month
- (iii) Retweet rate: the retweet rate of ego users to their followees

5.2. Comparison Methods. In order to verify the effectiveness of our proposed framework, we compared the prediction performance of UserRBPM in this paper with existing representative methods. Firstly, we compared UserRBPM with previous retweeting behavior prediction methods which usually extract rule-based features. Secondly, by comparing the GraLSP method with other network embedding methods, it is verified that the local structure information plays a more important role in the prediction of retweeting behavior than the global information. The comparison method is as follows:

- (i) Handcrafted features + Logistic Regression (LR): we use the logistic method to train the classification model. The features we constructed manually include two categories: one is the user node features designed in our work, including spatial-level and temporal-level features; the other is the ego network features designed by Qiu et al. [49]. The features we used are listed in Table 5
- (ii) Handcrafted features + Support Vector Machine (SVM): we also use SVM as the classification model. The model uses the same features as the LR method

TABLE 2: Statistics of subdataset *Edata*.

Subdataset	#users	#followrelationship	#original microblogs	#retweet	#ego users
Weibo	1,500,290	20,297,550	274,150	15,755,810	151,300

TABLE 3: Data statistics of the balanced sample set.

Methods	The size of negative samples	The number of tweets involved
Before subsampling	32,855,360	189,163
Subsampling of microblog granularity	899,756	122,568
Global random subsampling	899,756	109,330

- (iii) DeepWalk: DeepWalk [52] is a network embedding method that learns a social representation of a network by truncated random walks to obtain the structural information of each vertex. It can get better results even though there are few vertices in the network
- (iv) Node2vec: Node2vec [53] further extends the DeepWalk method by changing the way that the random walk sequence is generated. This is a network embedding method that designs a biased random walk that can tradeoff between homophily and structural equivalence of the network
- (v) Our proposed method: in our proposed UserRBPM framework, we use GraLSP to extract the structural attributes of the r -ego network, design the user node features at the spatial level and temporal level, and finally apply GCN and GAT to learn latent predictive signals

5.3. Evaluation Metrics

5.3.1. Performance Metrics. In order to quantitatively evaluate our proposed framework, we use the following metrics to evaluate the performance of retweeting behavior prediction. Specifically, we evaluate the performance of the UserRBPM in terms of Area Under Curve (AUC) [35], Precision, Recall, and F1-score.

- (i) Precision: it is for the predicted result. It measures the probability that a predicted positive instance would be the true positive
- (ii) Recall: it is for the original sample. It measures the probability that the true positive would be predicted to be the positive instance
- (iii) Area Under ROC Curve (AUC): it measures the probability that a classifier will rank a randomly chosen positive instance higher than a randomly chosen negative one.

- (iv) F1-score: it is a comprehensive evaluation metric that integrates precision and recall

5.3.2. Parameter Sensitivity. In addition, parameter sensitivity is also considered in our work. We analyzed several hyperparameters in the model and tested how different hyperparameter choices affect the prediction performance.

5.4. Implementation Details. There are two stages for training our UserRBPM framework. In the first stage, we pretrain each module of UserRBPM, and in the second stage, we integrate the three modules of UserRBPM for fine-tuning.

5.4.1. Stage I: Pretraining of Each Module. For our framework, UserRBPM, we first perform a random walk with a restart probability of 0.8 and set the size of the sampled sub-network to be 30. For the embedding layer, the embedding dimension of the GraLSP model is set to three dimensions of 32, 64, and 128, and we train GraLSP for 1000 epochs. Then, we choose to use a three-layer GCN or GAT network structure; the first and second GCN/GAT layers both contain 128 hidden units, while the third layer (output layer) contains 2 hidden units for binary prediction. In particular, for UserRBPM with multihead graph attention, both the first and second layers consist of $K = 8$ attention heads, and each attention head computes 16 hidden units (total $8 \times 16 = 128$ hidden units). The network is optimized by the Adam optimizer with a learning rate of 0.1, a weight decay of $5e-4$, and a dropout rate of 0.2. To evaluate the model performance and prevent information leakage, we performed fivefold cross-validation on our datasets. Specifically, we select 75% instances for training, 12.5% instances for validation, and 12.5% instances for testing. In addition, the minibatch size is set to be 1024 in our experiments.

5.4.2. Stage II: Global Fine-Tuning. In the global fine-tuning stage, if the dimension of an embedding layer is set too large, then the training process will be too slow, while a small setting will affect the performance of our model. After fine-tuning the model, we found that the model performance is relatively stable when the embedding dimension is set to 64. Then, we fix the parameters of the pretrained embedding module, and train the GCN/GAT layer with the Adam optimizer for 1000 epochs, with a learning rate of 0.001. A larger learning rate can make the model learn faster, thereby accelerating the convergence speed, but the performance of the model will be affected to some extent. Therefore, we set a relatively large learning rate at the beginning, and then we gradually decrease it during training. Finally, we choose the best model by stopping using the loss on the validation sets as early as possible. The training process of the UserRBPM model is shown in Algorithm 1.

TABLE 4: The statistics of social roles and relation statuses.

Social role	OrdUsr	OpnLdr	StrHole	Sum
Retweet behavior	6,617,440 (42%)	3,623,836 (23%)	5,514,534 (35%)	15,755,810
Original post	68,537 (25%)	123,367 (45%)	82,245 (30%)	274,150
Sum	1,125,217 (75%)	150,029 (10%)	225,043 (15%)	1,500,290

TABLE 5: List of features used in this work.

Name	Description
Spatial-level features	Social role_Opinion leader (OpnLdr)
	Social role_Structure hole (StrHole)
	Social role_Ordinary users (OrdUsr)
Temporal-level features	The TF-IDF similarity between an ego user and its followees' post content within a month (similarity)
	The number of microblogs posted by the followees within a month (exposure)
	The retweet rate of ego users to their followees (retweet rate)
Handcrafted ego network features [49]	The number/ratio of active neighbors
	Density of subnetwork induced by active neighbors
	Connected components formed by active neighbors

```

Input:  Datasets(train_loader, valid_loader, test_loader),
        Learning rate, Weight decay, Epochs, Batch size.
Output: The predictive value of the test samples and the prediction performance of the model.
1: Training process:
2:   Define the model: Three layers of GCN (two hidden layers and one output layer)
3:   Load the datasets: train_loader, valid_loader
4:   # model.train()
5:   for epoch = 1 to Epochs do:
6:     All data is propagated forward, and the activation function uses Leaky ReLU
7:     Use cross-entropy loss function to calculate the loss value
8:     Clear the gradient: optimizer.zero_grad()
9:     Backpropagate and calculate the gradient of the parameter
10:    Use Adam optimizer to update the gradient: optimizer.step()
11:    Calculate the accuracy of the current model on the training set
12:    Calculate the accuracy of the current model on validation set
13:  end for
14: Test process:
15:   # model.test()
16:   Load the trained model
17:   Load the dataset: test_loader
18:   Calculate the predicted value  $y_{pred}$  of the test samples
19:   Evaluate the model performance: Precision, Recall, F1-score, AUC

```

ALGORITHM 1: The training process of UserRBPM model.

6. Experiment Results

In this section, we give the quantitative and qualitative results of retweeting behavior prediction, then analyze the structural attributes and the interaction between spatial-level features and temporal-level features. Finally, the robustness of the UserRBPM framework is verified.

6.1. Prediction Performance Analysis

6.1.1. Overall Performance Analysis. To verify the influence of the structural attributes of users' ego network and the characteristics of user nodes (extracted from the spatial and temporal levels) on the prediction performance, as well as the interaction between features at different levels, we made the

TABLE 6: Prediction performance of different methods for retweeting behavior (%).

Methods	Precision	Recall	F1-score	AUC
Spatial- & Temporal-level & Handcrafted features + LR (ST&HC+LR)	69.74	71.58	70.65	77.27
Spatial- & Temporal-level & Handcrafted features + SVM (ST&HC+SVM)	68.38	69.15	68.76	78.01
DeepWalk + ST + GAT	78.21	78.46	78.28	82.81
DeepWalk + ST&HC + GAT	79.68	80.24	79.96	82.75
Node2vec + ST + GAT	78.54	81.50	79.99	82.53
Node2vec + ST&HC + GAT	79.88	81.25	80.55	82.96
Our method (UserRBPM)	81.97	82.58	82.27	83.21

following comparison. The prediction performance of different models is shown in Table 6.

Based on the analysis of four evaluation metrics used in our work, the performance of UserRBPM is better than the abovementioned benchmark methods, which demonstrate the effectiveness of our proposed framework. From the comparison among DeepWalk + ST&HC + GAT, Node2vec + ST&HC + GAT, and UserRBPM, we can observe that the GraLSP model we leverage in the embedding layer can indeed capture local structural patterns and significantly outperforms the first two methods in the experiment, confirming that the GraLSP can indeed capture local structural patterns in retweeting behavior prediction. Experiment results show that UserRBPM outperforms DeepWalk + ST + GAT by 3.76% in terms of precision and by 0.40% in terms of AUC. Moreover, the performance is also better from the perspective of Recall and F1-score. Meanwhile, from the comparison among ST&HC + LR, ST&HC + SVM, and UserRBPM, we notice that UserRBPM achieve an improvement of 13.59% in terms of precision. Such improvement verifies that the end-to-end learning framework UserRBPM can effectively detect potential driving factors and predictive signals in retweeting behavior prediction.

Comparing the first four methods (HC + LR, ST&HC + LR, HC + SVM, and ST&HC + SVM) with our proposed UserRBPM, it can be shown that the model which takes handcrafted features as input hardly represents interaction effects, while network embedding technology and graph attention can effectively extract high-dimensional structural attributes and can express highly nonlinear interaction mechanisms. Furthermore, from the comparison between HC + LR and ST&HC + LR (HC + SVM and ST&HC + SVM), Figure 5 shows that ST&HC + LR is notably better than HC + LR for retweeting behavior prediction. It reveals that users' spatial-level features and temporal-level features are the potential driving factors of retweeting behavior in social networks. Additionally, we observe that ST&HC + LR performs 4.42% better than HC + LR in terms of precision, verifying that the spatial-level and temporal-level features we designed have improved the prediction performance to a certain extent.

From the comprehensive analysis of the part to the whole, the space-level features and time-level features we designed have improved prediction performance. The combination of structural attributes and node features can further

improve prediction performance. Therefore, it demonstrates the effectiveness of our UserRBPM framework for retweeting behavior prediction.

6.1.2. Prediction Performance of Different Sampling Strategies. When dealing with data imbalance, we apply downsampling based on microblog granularity, instead of completely random downsampling. This sampling method would ensure that the number of positive and negative samples covered by the same microblog is the same, and the number of microblogs covered by negative samples is sufficient.

We use three sampling methods to obtain different training models. Among them, the directly sampling method (DSM) represents that we directly extract relatively balanced samples based on the ratio of the original positive and negative samples, that is, the ratio between positive and negative samples is set to 1:3. The number of positive samples and negative samples in completely random downsampling (CRDM) and our downsampling method (ODM) is the same. Experiment results are illustrated in Figure 6. Compared to the completely random downsampling method, the model trained with the samples obtained by our downsampling method has better prediction performance. The better the prediction effect of the model obtained by the training data training, the more it shows that the dataset has universal significance and the learned model has a stronger generalization ability. In the original imbalanced datasets, the direct extraction of positive and negative samples with a ratio of 1:3 is simple, but the difference in the number of microblogs covered by the positive and negative samples is ignored. Therefore, the downsampling method based on microblog granularity is more suitable for the user retweeting behavior prediction problem that we researched.

6.1.3. Comparative Analysis of Graph Convolution and Graph Attention. Table 7 is the prediction performance of two variants of graph deep learning, that is, the experimental results of using graph convolutional network (GCN) and graph attention mechanism (GAT) to build models, respectively. In previous research work, we have seen the success of GCN in classification tasks. However, in the application scenarios of our work, we observe that the performance of GCN in models constructed by different graph embedding

FIGURE 5: Prediction performance of different features.

technologies is generally worse than that of GAT. We attribute its disadvantage to the homophily assumption of GCN, that is, connected vertices tend to be similar (for example, have the same label). Under such assumption, for a specific vertex, GCN computes its hidden representation by taking an unweighted average over its neighbors' representation. This homophily exists in many real networks, but in our research scenario, different neighbor nodes may have different importance. Therefore, the graph attention mechanism (GAT) is introduced to assign different weights to different neighboring nodes. In essence, GAT is an aggregation func-

tion that focuses on the differences between neighbor nodes, rather than simple mean aggregation.

Besides, we wanted to avoid using handcrafted features and make UserRBPM a pure end-to-end learning framework, so we compared the prediction performance with additional vertex features and without additional vertex features. Comparison results of prediction performance with or without vertex features are presented in Table 8. It is observed that UserRBPM_GCN with handcrafted vertex features achieved an improvement of 2.18% in terms of precision, 1.46% in terms of recall, 0.44% in terms of F1-score, and

FIGURE 6: Prediction performance of different sampling strategies.

TABLE 7: Prediction performance of variants of UserRBPM (%).

Methods	Precision	Recall	F1-score	AUC
DeepWalk + ST&HC + GCN	77.49	79.28	78.37	74.89
DeepWalk + ST&HC + GAT	79.68	80.24	79.96	82.75
Node2vec + ST&HC + GCN	78.56	80.07	79.31	79.64
Node2vec + ST&HC + GAT	79.88	81.25	80.55	82.96
UserRBPM_GCN	80.82	80.58	80.70	82.23
UserRBPM_GAT	81.97	82.58	82.27	83.21

TABLE 8: The effect of handcrafted vertex features on prediction performance (%).

Methods	Handcrafted vertex features	Precision	Recall	F1-score	AUC
UserRBPM_GCN	Y	80.82	80.58	80.70	82.23
	N	78.64	79.12	80.26	81.64
UserRBPM_GAT	Y	81.97	82.58	82.27	83.21
	N	80.49	81.77	81.12	81.92

0.59% in terms of AUC. UserRBPM_GAT with handcrafted vertex features outperforms UserRBPM_GAT without handcrafted vertex features by 1.48% in terms of precision, 0.81% in terms of recall, 1.15% in terms of F1-score, and 1.29% in terms of AUC. Experiment results demonstrate that, in addition to the pretrained network embedding, we can still obtain comparable performance even without considering handcrafted features.

6.2. Parameter Sensitivity Analysis. In addition, we consider parameter sensitivity in our work. We analyzed several hyperparameters in the model and tested how different hyperparameter choices affect the prediction performance.

6.2.1. Robustness Analysis. To verify the robustness of the UserRBPM framework, we changed the proportion of the training set, validation set, and test set and then redo the experiments. The results in Figure 7 show that the model is effective under limited training data size. Even with a small size of the training set (20%-40%), our model can still have an acceptable and steady performance.

6.2.2. Effect of Instance Normalization. As mentioned in Section 4, this paper studied the technique used to accelerate model learning called *Instance Normalization* (IN). This technique provides benefits to improve the classification performance. For instance, it can learn faster while maintaining or even increasing accuracy. Moreover, it also partially serves as a parameter tuning method. Therefore, we applied IN and obtained a boost in both performance and generalization. Figure 8 shows the changes in the training loss of UserRBPM_GAT with or without the IN layer during training. We can see that when there is an instance normalization layer, as the number of epochs increases, the training loss first drops rapidly and then remains stable. Instance normalization significantly avoids overfitting and makes the training process more stable.

However, as shown in Figure 9, we observe that the model without the IN layer takes about 1011 seconds per epoch during the training process, while the model with the

FIGURE 7: Prediction performance with different training and test data size.

FIGURE 8: The effect of instance normalization.

IN layer takes about 1892 seconds per epoch. It was calculated that the model with the IN layer increased the training time for each epoch by about 87% compared to the model without the IN layer. Yet, we believe it is worthwhile to apply

IN, as the additional training time is compensated with a faster learning rate (it requires less number of epochs to reach the same level of precision) and can ultimately achieve higher testing precision.

FIGURE 9: Time overhead during each epoch.

7. Conclusion

In this work, we focus on user-level social influence in social networks and formulate the problem of user retweet behavior prediction from a deep learning perspective. Unlike previous work that built a prediction model of retweet behavior based on network topology maps of information dissemination or feature engineering-based approaches, we proposed a UserRBPM framework to predict the action status of a user given the action statuses of her near neighbors and her local structural information. Experiments on a large-scale real-world dataset have shown that the UserRBPM significantly outperforms baselines with handcrafted features in user retweet behavior prediction. This work explores the potential driving factors and predictable signals in user retweet behavior in hope that the deep learning framework has the better expressive ability and prediction performance.

For future research, experimental datasets related to this research field still contain rich social dynamics, which are worth further exploration. We can study user behavior in a semisupervised manner, develop a generic solution based on heterogeneous graph learning, and then extend it to many network mining tasks, such as link prediction, social recommendation, and similarity search. Through such a learning scheme, we can leverage both unsupervised information and limited labels of users to build the predictor, and verify the effectiveness and rationality of user behavior analysis on real-world datasets.

Data Availability

The original data used to support the findings of this study is from public data resources (<http://arnetminer.org/Influencelocality>). The data processing and parameter settings have been introduced Experiment Setup, so the data were not additionally provided.

Disclosure

This is an extended version of a conference paper, and the conference name is EAI MOBIMEDIA 2021-14th EAI International Conference on Mobile Multimedia Communications.

Conflicts of Interest

The authors declare that there is no conflict of interest regarding the publication of this paper.

Acknowledgments

This work was supported by the National Key R&D Program of China (2017YFB0801805, 2016YFB0801100), the National Natural Science Foundation of China (62072359, 62072352), and the Key Research and Development Plan of Xinjiang Production and Construction Corps. (No. 2019AB001).

References

- [1] R. Xiang, J. Neville, and M. Rogati, "Modeling relationship strength in online social networks," in *Proceedings of the 19th International Conference on World Wide Web*, pp. 981–990, 2010.
- [2] F. Riquelme and P. González-Cantergiani, "Measuring user influence on Twitter: a survey," *Information Processing & Management*, vol. 52, no. 5, pp. 949–975, 2016.
- [3] T. R. Zaman, R. Herbrich, J. Van Gael, and D. Stern, "Predicting information spreading in twitter," in *Workshop on Computational Social Science and the Wisdom of Crowds, Nips*, vol. 104, no. 45pp. 17599–17601, Citeseer, 2010.
- [4] M. C. Yang, J. T. Lee, S. W. Lee, and H. C. Rim, "Finding interesting posts in twitter based on retweet graph analysis," in *Proceedings of the 35th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 1073–1074, 2012.
- [5] L. Hong, O. Dan, and B. D. Davison, "Predicting popular messages in twitter," in *Proceedings of the 20th International Conference Companion on World Wide Web*, pp. 57–58, 2011.
- [6] Q. Zhao, M. A. Erdogdu, H. Y. He, A. Rajaraman, and J. Leskovec, "Seismic: a self-exciting point process model for predicting tweet popularity," in *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 1513–1522, 2015.
- [7] W. Fang, X. Yao, X. Zhao, J. Yin, and N. Xiong, "A stochastic control approach to maximize profit on service provisioning for mobile cloudlet platforms," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 48, no. 4, pp. 522–534, 2016.
- [8] E. Bakshy, D. Eckles, R. Yan, and I. Rosenn, "Social influence in social advertising: evidence from field experiments," in *Proceedings of the 13th ACM Conference on Electronic Commerce*, pp. 146–161, 2012.
- [9] Y. Matsubara, Y. Sakurai, B. A. Prakash, L. Li, and C. Faloutsos, "Rise and fall patterns of information diffusion: model and implications," in *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 6–14, 2012.

- [10] J. Wang and Y. Q. Wang, "SIR rumor spreading model with network medium in complex social networks," *Chinese Journal of Physics*, 2015.
- [11] Y. Liu, J. Zhao, and Y. Xiao, "C-RBFNN: a user retweet behavior prediction method for hotspot topics based on improved RBF neural network," *Neurocomputing*, vol. 275, pp. 733–746, 2018.
- [12] J. Xiong, X. Chen, Q. Yang, L. Chen, and Z. Yao, "A task-oriented user selection incentive mechanism in edge-aided mobile crowdsensing," *IEEE Transactions on Network Science and Engineering*, vol. 7, no. 4, pp. 2347–2360, 2020.
- [13] B. Yi, X. Shen, H. Liu et al., "Deep matrix factorization with implicit feedback embedding for recommendation system," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 8, pp. 4591–4601, 2019.
- [14] R. Bi, Q. Chen, L. Chen, J. Xiong, and D. Wu, "A privacy-preserving personalized service framework through Bayesian game in social IoT," *Wireless Communications and Mobile Computing*, vol. 2020, 13 pages, 2020.
- [15] R. M. Bond, C. J. Fariss, J. J. Jones et al., "A 61-million-person experiment in social influence and political mobilization," *Nature*, vol. 489, no. 7415, pp. 295–298, 2012.
- [16] J. Tang, J. Sun, C. Wang, and Z. Yang, "Social influence analysis in large-scale networks," in *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 807–816, 2009.
- [17] Y. A. Kim and J. Srivastava, "Impact of social influence in e-commerce decision making," in *Proceedings of the Ninth International Conference on Electronic Commerce*, pp. 293–302, 2007.
- [18] J. Zhang, J. Tang, J. Li, Y. Liu, and C. Xing, "Who influenced you? Predicting retweet via social influence locality," *ACM Transactions on Knowledge Discovery from Data (TKDD)*, vol. 9, no. 3, pp. 1–26, 2015.
- [19] B. Jiang, J. Liang, Y. Sha et al., "Retweeting behavior prediction based on one-class collaborative filtering in social networks," in *Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 977–980, 2016.
- [20] J. Zhang, J. Tang, Y. Zhong et al., "Structinf: mining structural influence from social streams," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 31, 2017no. 1.
- [21] Q. Li, Z. Han, and X. M. Wu, "Deeper insights into graph convolutional networks for semi-supervised learning," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, 2018no. 1.
- [22] M. Chen, Z. Wei, Z. Huang, B. Ding, and Y. Li, "Simple and deep graph convolutional networks," in *International Conference on Machine Learning*, pp. 1725–1735, 2020.
- [23] J. Xiong, R. Bi, M. Zhao, J. Guo, and Q. Yang, "Edge-assisted privacy-preserving raw data sharing framework for connected autonomous vehicles," *IEEE Wireless Communications*, vol. 27, no. 3, pp. 24–30, 2020.
- [24] J. Sun, X. Wang, N. Xiong, and J. Shao, "Learning sparse representation with variational auto-encoder for anomaly detection," *IEEE Access*, vol. 6, pp. 33353–33361, 2018.
- [25] C. Zhang, D. Song, C. Huang, A. Swami, and N. V. Chawla, "Heterogeneous graph neural network," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 793–803, 2019.
- [26] Y. Ota, K. Maruyama, and M. Terada, "Discovery of interesting users in twitter by overlapping propagation paths of retweets," in *2012 IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology*, vol. 3, pp. 274–279, Macau, China, 2012.
- [27] N. J. Yuan, Y. Zhong, F. Zhang, X. Xie, C. Y. Lin, and Y. Rui, "Who will reply to/retweet this tweet? The dynamics of intimacy from online social interactions," in *Proceedings of the Ninth ACM International Conference on Web Search and Data Mining*, pp. 3–12, 2016.
- [28] J. Tang, S. Wu, and J. Sun, "Confluence: conformity influence in large social networks," in *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 347–355, 2013.
- [29] X. Zhang, D. D. Han, R. Yang, and Z. Zhang, "Users' participation and social influence during information spreading on Twitter," *PLoS One*, vol. 12, no. 9, article e0183290, 2017.
- [30] Z. Luo, M. Osborne, J. Tang, and T. Wang, "Who will retweet me? Finding retweeters in Twitter," in *Proceedings of the 36th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 869–872, 2013.
- [31] C. Li, J. Ma, X. Guo, and Q. Mei, "DeepCas: an end-to-end predictor of information cascades," in *Proceedings of the 26th International Conference on World Wide Web*, pp. 577–586, 2017.
- [32] Q. Zhang, Y. Gong, J. Wu, H. Huang, and X. Huang, "Retweet prediction with attention-based deep neural network," in *Proceedings of the 25th ACM International Conference on Information and Knowledge Management*, pp. 75–84, 2016.
- [33] X. Wang, Z. Guo, X. Wang, S. Liu, W. Jing, and Y. Liu, "Nnmlinf: social influence prediction with neural network multi-label classification," in *Proceedings of the ACM Turing Celebration Conference—China*, pp. 1–5, 2019.
- [34] C. Donnat, M. Zitnik, D. Hallac, and J. Leskovec, "Learning structural node embeddings via diffusion wavelets," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 1320–1329, 2018.
- [35] J. Zhang, J. Tang, C. Ma, H. Tong, Y. Jing, and J. Li, "Panther: fast top-k similarity search on large networks," in *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 1445–1454, 2015.
- [36] J. Zhang, J. Tang, C. Ma, H. Tong, Y. Jing, and J. Li, "An end-to-end deep learning architecture for graph classification," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, 2018no. 1.
- [37] F. Y. Sun, J. Hoffmann, V. Verma, and J. Tang, "Infograph: unsupervised and semi-supervised graph-level representation learning via mutual information maximization," 2019, <https://arxiv.org/abs/1908.01000>.
- [38] W. Chen, Y. Gu, Z. Ren et al., "Semi-supervised user profiling with heterogeneous graph attention networks," *IJCAI*, vol. 19, pp. 2116–2122, 2019.
- [39] W. Fan, Y. Ma, Q. Li et al., "Graph neural networks for social recommendation," in *The World Wide Web Conference*, pp. 417–426, 2019.
- [40] D. J. Watts and S. H. Strogatz, "Collective dynamics of 'small-world' networks," *Nature*, vol. 393, no. 6684, pp. 440–442, 1998.
- [41] H. Tong, C. Faloutsos, and J. Y. Pan, "Fast random walk with restart and its applications," in *Sixth International Conference*

- on Data Mining (ICDM'06), pp. 613–622, Hong Kong, China, 2006.
- [42] L. Backstrom, D. Huttenlocher, J. Kleinberg, and X. Lan, “Group formation in large social networks: membership, growth, and evolution,” in *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 44–54, 2006.
 - [43] J. Ugander, L. Backstrom, C. Marlow, and J. Kleinberg, “Structural diversity in social contagion,” *Proceedings of the National Academy of Sciences*, vol. 109, no. 16, pp. 5962–5966, 2012.
 - [44] D. Wang, P. Cui, and W. Zhu, “Structural deep network embedding,” in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 1225–1234, 2016.
 - [45] P. Cui, X. Wang, J. Pei, and W. Zhu, “A survey on network embedding,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 31, no. 5, pp. 833–852, 2018.
 - [46] H. Huang, J. Tang, L. Liu, J. D. Luo, and X. Fu, “Triadic closure pattern analysis and prediction in social networks,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 27, no. 12, pp. 3374–3389, 2015.
 - [47] Y. Jin, G. Song, and C. Shi, “GraLSP: graph neural networks with local structural patterns,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34no. 4, pp. 4361–4368, 2020.
 - [48] D. Ulyanov, A. Vedaldi, and V. Lempitsky, “Instance normalization: the missing ingredient for fast stylization,” 2016, <https://arxiv.org/abs/1607.08022>.
 - [49] J. Qiu, J. Tang, H. Ma, Y. Dong, K. Wang, and J. Tang, “DeepInf: social influence prediction with deep learning,” in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 2110–2119, 2018.
 - [50] H. Wu, Z. Hu, J. Jia, Y. Bu, X. He, and T. S. Chua, “Mining unfollow behavior in large-scale online social networks via spatial-temporal interaction,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34no. 1, pp. 254–261, 2020.
 - [51] Y. Yang, J. Jia, B. Wu, and J. Tang, “Social role-aware emotion contagion in image social networks,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 30, 2016no. 1.
 - [52] B. Perozzi, R. Al-Rfou, and S. Skiena, “DeepWalk: online learning of social representations,” in *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 701–710, 2014.
 - [53] A. Grover and J. Leskovec, “node2vec: scalable feature learning for networks,” in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 855–864, 2016.