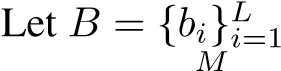
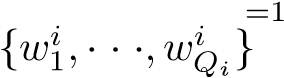
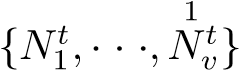
II. BOTRGCN METHODOLOGY

# A. Problem Definition

 denotes a user’s description with *L* words.

Let *T* = {*ti*}*i* be a user’s *M* tweets and each tweet *ti* = contains *Qi* words. Let *P* = {*Pnum,Pcat*} be a user’s numerical and categorical user property set. Let

*N* = {*Nf,Nt*} be a user’s neighborhood information, where  denotes user’s followings and *Nt* =  denotes user’s followers. The task of Twitter bot detection is to identify bots among users with the help of user information *B*, *T*, *P* and *N*.

# B. User Feature Encoding

BotRGCN is designed to address the challenge of disguise by leveraging multi-modal user information, which leaves bot operators no venue to achieve malicious goals. Specifically,

BotRGCN jointly encodes user semantic information of description and tweets as well as both numerical and categorical user property information.

1. *Overall user feature vector:* We encode user description, tweets, numerical and categorical properties and concatenate them to serve as user features:

 (1)

where *D* is the user embedding dimension. We present BotRGCN’s strategy of encoding user desciption, tweets, numerical and categorical property items in the following.

1. *Feature set 1: user description:* We adopt pre-trained RoBERTa [12] to encode user descriptions. We firstly transform words in user description with RoBERTa:

 (2)

where ¯*b* denotes representation of user description and *Ds* is the RoBERTa embedding dimension. We then derive representation vectors for user’s description:

*rb* = *φ*(*WB* ·¯*b* + *bB*)*, rb* ∈ R*D/*4×1 (3)

where *WB* and *bB* are learnable parameters, *φ* is the activation function and *D* is the embedding dimension of Twitter users. We adopt leaky-relu as *φ* for the rest of the paper.

1. *Feature set 2: user tweets:* We use RoBERTa to similarly encode user tweets. We average the representation of all tweets to obtain representation of user tweets *rt*.
2. *Feature set 3: user numerical properties:* BotRGCN leaves handling of user property items to MLPs and graph neural networks. Specifically, we adopt numerical featrues that are directly available from the Twitter API without feature engineering and present them in Table I. Specifically, we conduct z-score normalization and obtain representation of user numerical features *rpnum* with a fully connected layer.
3. *Feature set 4: user categorical properties:* Similar to user numerical properties, we avoid feature engineering and apply MLPs and graph neural networks to encode them. We leverage directly available user categorical features from the Twitter API and they are presented in Table II. Specifically, we adopt one-hot encoding, concatenate and transform them with a fully connected layer and leaky-relu to derive representation for user’s categorical features.

TABLE II

CATEGORICAL USER PROPERTIES ADOPTED IN BOTRGCN.

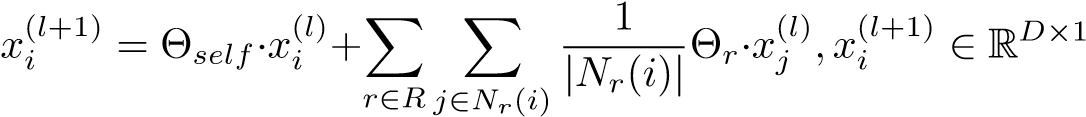
# C. GNNs Architecture

BotRGCN is designed to address the challenge of community by leveraging user follow relationship and the dense graph structure it forms. Specifically, BotRGCN constructs a heterogeneous graph from the Twitter network and apply relational graph convolutional networks to learn user representations.

1. *Graph construction:* BotRGCN treats Twitter users as nodes. Given that following and being followed signal different information, BotRGCN leverages two types of edges, *R* = {*r*1*,r*2} = {”*following*”*,*”*follower*”}. We denote user *u*’s following and follower neighborhood as *Nr*1(*u*) = *Nf*(*u*) and *Nr*2(*u*) = *Nt*(*u*). By defining two sets of relational neighborhood for each Twitter user, BotRGCN constructs a heterogeneous graph that reflects the interactions between Twitter users. BotRGCN could incorporate more relation types between users if supported by the data set.
2. *BotRGCN architecture:* We apply R-GCNs [13] to the heterogeneous graph and learn user representations. Specifically, we firstly transform user features to derive the initial hidden vectors for nodes in the graph:

 (4)

where *W*1 and *b*1 are learnable parameters. We then apply the *l*-th R-GCN layers:



(5)

where Θ is the projection matrix. After *L* layers of R-GCN, we transform the user representation with MLP:

 (6)

where *W*2 and *b*2 are learnable parameters and *hi* is the representation for user *i*.

TABLE III

# D. Learning and Optimization

We apply a softmax layer to conduct Twitter bot detection based on user representations derived from R-GCN :

*y*ˆ*i* = *softmax*(*WO* · *hi* + *bO*) (7)

where *WO* and *bO* are learnable parameters.

The loss function of BotRGCN is constructed as follows:

*L* = −X[*yilog*(*y*ˆ*i*) + (1 − *yi*)*log*(1 − *y*ˆ*i*)] + *λ* X *w*2 (8)

*i*∈*Y w*∈*θ*

where *Y* denotes annotated users, *yi* is the ground-truth label and *θ* are all learnable parameters in the BotRGCN framework.