Rumor Detection on Social Media with Graph Structured Adversarial Learning

Xiaoyu Yang, **Yuefei Lyu**, **Tian Tian**, **Yifei Liu**, **Yudong Liu** and **Xi Zhang***
Key Laboratory of Trustworthy Distributed Computing and Service (BUPT),
Ministry of Education, Beijing University of Posts and Telecommunications,
Beijing, China
{littlehaes, lvyuefei, tiantian_96727, liuyifei, yudong.liu, zhangx}@bupt.edu.cn

Abstract

The wide spread of rumors on social media has caused tremendous effects in both the online and offline world. In addition to text information, recent detection methods began to exploit the graph structure in the propagation network. However, without a rigorous design, rumors may evade such graph models using various *camouflage* strategies by perturbing the structured data. Our focus in this work is to develop a robust graph-based detector to identify rumors on social media from an adversarial perspective. We first build a heterogeneous information network to model the rich information among users, posts, and user comments for detection. We then propose a graph adversarial learning framework, where the attacker tries to dynamically add intentional perturbations on the graph structure to fool the detector, while the detector would learn more distinctive structure features to resist such perturbations. In this way, our model would be enhanced in both robustness and generalization. Experiments on real-world datasets demonstrate that our model achieves better results than the state-of-the-art methods.

1 Introduction

With the rapid growth of social media such as Twitter and Weibo, information campaigns are frequently carried out by misinformation producers with various commercial and political purposes. Consequently, large amounts of fake or unverified information have emerged and spread to affect online social network users, which also leads to tremendous effects on the offline society. For example, the wide spread of fake news on social media has influenced the 2016 US presidential election [Allcott and Gentzkow, 2017]. Thus, identifying rumors automatically is beneficial for providing early precautions to minimize its negative influence.

Traditional rumor detection methods such as [Yang *et al.*, 2012; Ma *et al.*, 2015; Kwon *et al.*, 2017] mainly focused on extracting hand-crafted features from post contents, user profiles, and diffusion patterns as inputs for supervised learning algorithms. Recently, powerful deep learning models without heavy feature engineering have been applied in this task. RNN-, CNN-, and autoencoder-based methods have achieved significant improvements over traditional methods. However, existing methods mostly treat the detection of each piece of information independently, ignoring their correlations. On social networks, there are rich structure correlations among different pieces of information. For instance, multiple posts can be connected if they are posted or retweeted by the same user. Such correlations enable sharing knowledge between connected instances, assisting each other's detection to improve the performance. Recent efforts on utilizing structure information for rumor and fake news detection have shown promising performance. In [Yuan *et al.*, 2019], based on a heterogeneous graph constructed from posts, comments, and the related users on social networks, structure semantics are captured by graph attention network[Veličković *et al.*, 2018]. In [Zhang *et al.*, 2018], a deep diffusive network model is built to learn the representations of news articles, creators, and subjects simultaneously.

Despite the success of deep graph neural networks, the lack of interpretability and robustness would make it risky for rumor detection. A fundamental assumption of these models is that a piece of information connecting with high-credit users would also have high credit, but the credit can be manipulated

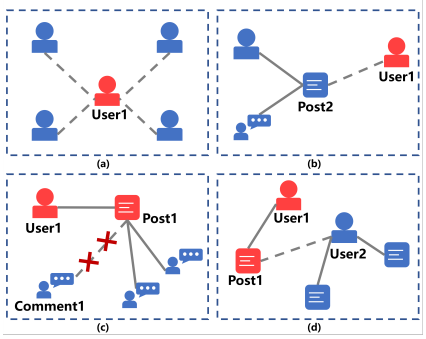

Figure 1: Four types of camouflage strategies on a social graph: (a) rumor spreader User1 buys many fans; (b) rumor spreader User1 forwards a normal post Post2; (c) rumor spreader User1 deletes an unfavorable comment Comment1; (d) rumor spreader User1 hacks a normal account User2 and uses User2 to forward the rumor Post1

*Corresponding author

by rumor producers. A rumor could try to disguising itself by posting with hijacked or traded accounts from honest users. For example, in 2013, Associated Press's Twitter account was hijacked and the rumor of explosion at White House was posted, causing the stock market to briefly plunge. Other cheaper and more common camouflage strategies include connecting the posting user with high-credit users, removing opposing comments, and creating fake supporting comments. Figure 1 shows four types of camouflages on the social graph. Such attacks would make the rumor evade graph-based models by intentionally modifying the graph structures. However, little attention has been paid to this severe issue.

We argue that considering camouflages are necessary to complement prior work on rumor detection. We focus on the attack type that fools the graph neural network-based detection model by manipulating the graph structure. In this regard, RL-based and gradient-based adversarial attack methods on graph-structured data have been proposed in [Dai *et al.*, 2018], but the target graph they consider is homogeneous and the structure can be modified arbitrarily. In contrast, the graph constructed for rumor detection is heterogeneous and involves domain constraints. For example, a post cannot be connected with another post directly.

In this paper, we investigate: (i) how to simulate various camouflage methods on social networks to evade the rumor detector (ii) how to make the detector reveal camouflages as much as possible. To address these two issues, we rigorously define the possible camouflage, i.e., attack types in consideration of domain constraints. We then propose a novel graph adversarial learning framework that enables such attacks to fool the detector by dynamically adding intentional perturbations. Meanwhile, the detector would be enhanced to learn more distinctive structural features to resist such perturbations. The attacker and detector would strengthen each other iteratively, making our model produce robust detection results.

The main contributions of this paper can be summarized as follows:

- We study a novel problem of rumor detection on social media in the face of camouflage.
- We propose an end-to-end framework that jointly exploits text and structure information for detection.
- We develop a graph adversarial learning method that encourages the model to provide robust predictions under the perturbed graph structure.
- We conduct extensive experiments on real-world datasets to demonstrate the effectiveness of our model.

2 Related Work

Compared to the traditional machine learning models such as [Zhao *et al.*, 2015; Ma *et al.*, 2017; Castillo *et al.*, 2011], deep learning models have achieved state-of-the-art performance for rumor detection. Most of these models adopt the text information for detection. [Ma *et al.*, 2018] uses recursive neural networks with a bottom-up structure and a top-down tree structure to extract text features. [Khattar *et al.*, 2019] uses a variational autoencoder to obtain features of texts and pictures to determine whether the post is a rumor. [Liu and Wu, 2018] models the propagation path of a tweet as a multivariate time series, and uses RNN and CNN to capture the global and local variations of user characteristics along propagation paths. The common limitation of these studies is that they don't fully utilize the social network structure information. In contrast, [Yuan *et al.*, 2019] not only uses text features but also extracts network structure features through the Graph Attention Network [Veličković *et al.*, 2018], which shows good performance. However, it doesn't consider the camouflage behaviors in the social network, and thus may fail to detect the camouflaged rumors.

[Wang *et al.*, 2018] obtains non-event-specific features from texts and pictures with the idea of the generative adversarial network (GAN) [Goodfellow *et al.*, 2014] for detection. [Ma *et al.*, 2019] uses GAN to generate uncertain or conflicting voices to learn more effective features. Different from these GAN-based works that manipulate the texts, we perturb the network structure to learn stronger structure features. Although there are some recent studies on graph adversarial attacks [Xu *et al.*, 2019; Wu *et al.*, 2019; Dai *et al.*, 2019], they perform general attacks without considering the heterogeneity and domain constraints that exist in real-world applications such as rumor detection.

3 Problem Definition

Let $\mathcal{P} = \{p_1, p_2, ..., p_{|\mathcal{P}|}\}$ be a set of posts on social media, $\mathcal{R}(p_i) = \{r_1, r_2, ..., r_{|\mathcal{R}|}\}$ be a set of comments of p_i, $\mathcal{U} = \{u_1, u_2, ..., u_{|\mathcal{U}|}\}$ be a set of users who deliver one or more posts or comments.

To exploit the posting and retweeting behaviours on the social network for classification, we construct the heterogeneous graph $G = (V, E, A)$, where V is the set of nodes, E is the set of edges and $A \in \{0, 1\}^{|V|\times|V|}$ is the adjacency matrix. There are three types of nodes, users, social posts, and user comments. To construct the graph, there would be an edge between a user and her post. The user and her comment would also be connected. Users are connected according to the follower/followee relationships. Moreover, there would be an edge between two comments if one is the comment for the other. For simplicity, we don't consider the directions on the edges and thus treat G as an undirected graph.

We take the rumor detection task as a binary classification problem. $c \in \{0, 1\}$ denotes the class labels where $c = 1$ represents the rumor and the other represents the non-rumor. Our goal is to train a model $f(\cdot)$ to predict the label of a given post where $f(p_i) = 1$ denotes p_i as a rumor and $f(p_i) = 0$ denotes p_i as a non-rumor

4 Methodology

4.1 Model Overview

The proposed rumor detection model consists of three main components, *encoding camouflaged graph*, *encoding text contents*, and *classification with adversarial learning*. Figure 2 shows the architecture of the proposed model.

We define several widely used camouflage strategies, and automatically learn to generate camouflages on the graph. We then extract the structure representations from camouflaged

Node type	Good User	Bad User	Rumor	Non-rumor	Comment
Good User	0	1	1	0	0
Bad User	1	0	0	1	0
Rumor	1	0	0	0	1
Non-rumor	0	1	0	0	0
Comment	0	0	1	0	0

Table 1: Camouflage compatibility matrix. The entry "1" indicates allowing a camouflaged attack (i.e., addition or deletion of an edge) between these two types of nodes, while the entry "0" does not allow.

graphs for each post. Besides structure information, we also exploit the post contents to extract the text representations for each post and fuse text and structure representations to make a classification. This classification loss with camouflaged structure features would be combined with another standard classification loss with non-camouflaged structure features. To optimize the sum of these two losses, we train the model end-to-end with an adversarial learning method.

4.2 Encoding Camouflaged Graph

Generating Camouflages on Graph

To simulate the camouflage strategies of a rumor spreader, we divide users into two categories: bad user and good user. Bad users refer to those who deliver or forward at least one piece of rumor and the others are good users. We support the following four types of camouflages as shown in Figure 1:

(a) a bad user node links to good user nodes which represents the bad user disguises himself by buying some fans or following some good users.

(b) a bad user node links to non-rumor post nodes, indicating a rumor spreader carries out normal social activities.

(c) a rumor spreader node deletes the edges between the rumor post node and the opposing comment nodes, avoiding being caught.

(d) a good user node links to rumor post nodes, indicating a good user account was hacked or bought to forward a rumor.

To formally describe the camouflage strategies, we define a camouflage compatibility matrix in Table 1 which corresponds to Figure 1 exactly. The entry values "0" and "1" denote it does and doesn't allow a camouflage between the corresponding two types of nodes respectively. We then define an action mask matrix $\boldsymbol{M} \in \{0,1\}^{|V|\times|V|}$, where $\boldsymbol{M_{ij}} = 1$ if it allows a camouflage between node types of n_i and n_j ($\{n_i, n_j\} \in V$) in Table 1. Otherwise, $\boldsymbol{M_{ij}} = 0$. For example, if n_i is a rumor post and n_j is a comment, in this case, $\boldsymbol{M_{ij}} = 1$ because a camouflaged modification between a rumor and a comment is allowed in Table 1.

Apart from the action mask matrix $\boldsymbol{M}$, we also define an action matrix $\boldsymbol{S} \in \{0,1\}^{|V|\times|V|}$ to specify the camouflage actions that are actually taken. Specifically, $\boldsymbol{S_{ij}} = 1$ means a camouflage action is performed between n_i and n_j. Otherwise, $\boldsymbol{S_{ij}} = 0$ indicates no action is performed. For example, given $\boldsymbol{A_{ij}} = 1$ (resp. $\boldsymbol{A_{ij}} = 0$), $\boldsymbol{S_{ij}} = 1$ would lead to the deletion (resp. addition) of the edge between n_i and n_j as a camouflage. The adjacency matrix is thus modified and we obtain $\boldsymbol{A_{ij}} = 0$ (resp. $\boldsymbol{A_{ij}} = 1$) accordingly. To connect the modification of $\boldsymbol{A_{ij}}$ with the action $\boldsymbol{S_{ij}}$, we further introduce an assistant matrix $\boldsymbol{C} \in \{0,1,-1\}^{|V|\times|V|}$. $\boldsymbol{C} = \bar{\boldsymbol{A}} - \boldsymbol{A}$, where $\bar{\boldsymbol{A}} = \boldsymbol{11^{\mathrm{T}}} - \boldsymbol{I} - \boldsymbol{A}$. $\boldsymbol{I}$ is the identity matrix and $(\boldsymbol{11^{\mathrm{T}}} - \boldsymbol{I})$ represents the fully-connected graph. Since $\boldsymbol{C} = \bar{\boldsymbol{A}} - \boldsymbol{A}$, directly apply $\boldsymbol{C}$ to $\boldsymbol{A}$ would make all the entries (except the diagonal ones) in $\boldsymbol{A}$ are modified from 1 to 0 or from 0 to 1, indicating performing camouflages on all the pairs of nodes. Actually, whether it is allowed to perform a specific camouflage between a pair of nodes is determined by $\boldsymbol{M}$, and even if it is allowed, whether it is performed is determined by $\boldsymbol{S}$. Thus, the final camouflaged adjacency matrix $\boldsymbol{A'}$ is obtained by jointly involving these matrices, that is,

$$\boldsymbol{A'} = \boldsymbol{A} + \boldsymbol{C} \circ \boldsymbol{S} \circ \boldsymbol{M} \tag{1}$$

where $\circ$ represents element-wise product.

Note that $\boldsymbol{A}$, $\boldsymbol{M}$, and $\boldsymbol{C}$ are predetermined according to the dataset. $\boldsymbol{S_{ij}}$ is updated in the training process. For ease of optimization, we relax $\boldsymbol{S_{ij}} \in \{0,1\}$ to its convex hull $\boldsymbol{S_{ij}} \in [0,1]$. To enable this learning, we use the attention mechanism to derive the elements of $\boldsymbol{S}$.

For example, suppose n_i is a rumor node and n_j is a good user node. In this work, representations of all kinds of nodes have the same dimension of d. $\{\boldsymbol{v_i}, \boldsymbol{v_j}\} \in \mathbb{R}^d$ represent the node vectors of n_i and n_j respectively. Considering the heterogeneity of the graph, we first transform $\boldsymbol{v_i}$ and $\boldsymbol{v_j}$ to the same feature space through $\boldsymbol{W_p v_i}$ and $\boldsymbol{W_u v_j}$ where $\boldsymbol{W_p} \in \mathbb{R}^{d'\times d}$ and $\boldsymbol{W_u} \in \mathbb{R}^{d'\times d}$ are two parameterized weight matrices for posts and users respectively. Similarly, there is also $\boldsymbol{W_r} \in \mathbb{R}^{d'\times d}$ for comments. In this way, we transform the d-dimensional $\boldsymbol{v_i}$ to the d'-dimensional vector. Then the attention coefficient is obtained through a single-layer neural network $g : \mathbb{R}^{d'} \times \mathbb{R}^{d'} \to \mathbb{R}$

$$e_{ij} = \sigma(\boldsymbol{a_1^{\mathrm{T}} W_p v_i} + \boldsymbol{a_2^{\mathrm{T}} W_u v_j}) \tag{2}$$

where σ is an activation function such as $ReLU$. $\boldsymbol{a_1} \in \mathbb{R}^{d'}$ and $\boldsymbol{a_2} \in \mathbb{R}^{d'}$ are parameters in the attention corresponding to posts and users respectively. There is also $\boldsymbol{a_3} \in \mathbb{R}^{d'}$ corresponding to comments. e_{ij} indicates the importance of n_j to n_i. We denote $\mathcal{N}_i$ as a set of nodes that could have camouflaged actions with n_i. Then softmax is applied to normalize each attention coefficient by

$$\alpha_{ij} = softmax_j(e_{ij}) = \frac{exp(e_{ij})}{\sum_{k\in\mathcal{N}_i} exp(e_{ik})}. \tag{3}$$

The more important n_j is to n_i, the greater value α_{ij} has. In our application, there is $\boldsymbol{S}_{ij} = \alpha_{ij}$, indicating the extent of camouflage action between n_j and n_i. In this way, we parameterize $\boldsymbol{A'}$. With the camouflaged graph, we then discuss how to extract the graph structure features.

Encoding Structures

Graph neural networks (GNNs) have shown superior capability in capturing the graph structures. GCN [Kipf and Welling, 2017] and GAT [Veličković *et al.*, 2018] are two widely-used models. We chose GCN in this work.

First, the camouflaged adjacency matrix is added with the self-connections, that is

$$\tilde{\boldsymbol{A}} = \boldsymbol{A'} + \boldsymbol{I} \tag{4}$$

Then the propagation step from the l-th layer to the $(l+1)$-th layer is

$$\boldsymbol{H^{(l+1)}} = \sigma(\tilde{\boldsymbol{D}}^{-\frac{1}{2}}\tilde{\boldsymbol{A}}\tilde{\boldsymbol{D}}^{-\frac{1}{2}}\boldsymbol{H^{(l)}}\boldsymbol{W^{(l)}}) \tag{5}$$

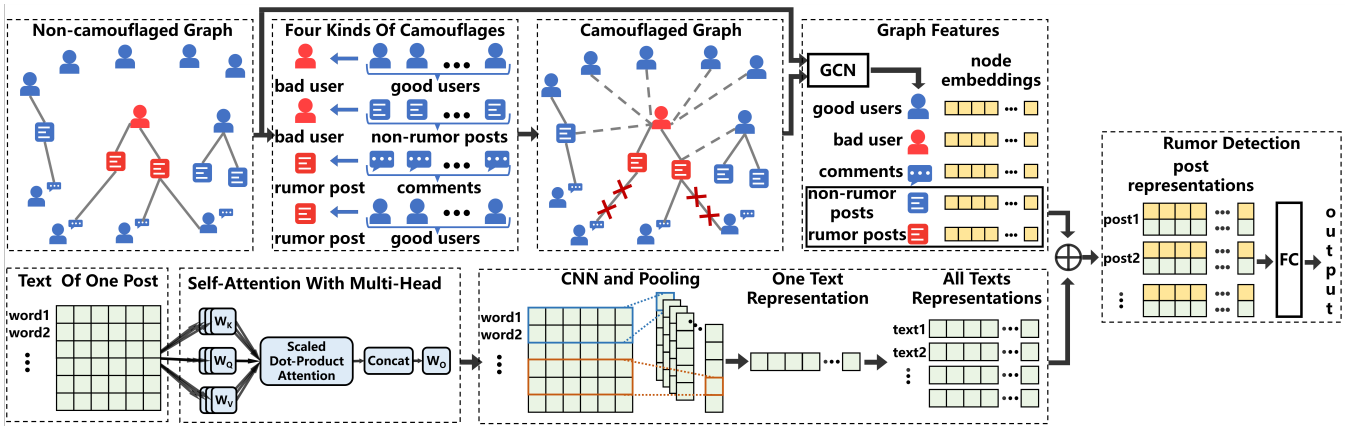

Figure 2: Framework. For a social graph, we generate camouflages on it, and fed it into GCN to obtain the perturbed structure representation for each post. We also extract the text representation for each post with multi-head self-attention and CNN. The two representations are concatenated for classification. We also consider the standard method that extracts structure representations from the non-camouflaged graph and combine it with the text representation for classification. These two classification mechanisms are optimized jointly.

where σ is the ReLU activation function. $\boldsymbol{H^{(l)}} \in \mathbb{R}^{|V|\times d}$ is the matrix of activations in the l-th layer. $\boldsymbol{H^{(0)}}$ is the node embedding matrix $\boldsymbol{B} \in \mathbb{R}^{|V|\times d}$. The element of $\tilde{\boldsymbol{D}}$ is $\tilde{\boldsymbol{D}}_{\boldsymbol{ii}} = \sum_j \tilde{\boldsymbol{A}}_{\boldsymbol{ij}}$. $\boldsymbol{W^{(l)}}$ is a trainable weight matrix. We use a two-layer GCN to extract structure features. It is calculated as

$$f(\boldsymbol{B}, \boldsymbol{A}) = softmax(\hat{\boldsymbol{A}}\sigma(\hat{\boldsymbol{A}}\boldsymbol{B}\boldsymbol{W^{(0)}})\boldsymbol{W^{(1)}}) \quad (6)$$

where $\hat{\boldsymbol{A}} = \tilde{\boldsymbol{D}}^{-\frac{1}{2}}\tilde{\boldsymbol{A}}\tilde{\boldsymbol{D}}^{-\frac{1}{2}}$ and $\boldsymbol{W^{(0)}} \in \mathbb{R}^{d\times d'}$ is the input-to-hidden weight matrix. $\boldsymbol{W^{(1)}} \in \mathbb{R}^{d'\times d'}$ is the hidden-to-output weight matrix. $\boldsymbol{f(B, A)} \in \mathbb{R}^{|V|\times d'}$ is the updated node embedding matrix. We denote $\boldsymbol{h'_i} \in \mathbb{R}^{d'}$ as the i-th node embedding in $f(\boldsymbol{B}, \boldsymbol{A})$ which represents the structure feature of p_i.

Besides extracting the structure representation $\boldsymbol{h'_i}$ from the camouflaged graph $\boldsymbol{A'}$, we also extract the structure representation of p_i from the original graph $\boldsymbol{A}$, denoted as $\boldsymbol{h_i}$. Both $\boldsymbol{h_i}'$ and $\boldsymbol{h_i}$ would be concatenated with the text representation (described in the next subsection) for classification.

4.3 Encoding Text Contents

To obtain the text representation for each post, we adopt the multi-head self-attention and CNN with pooling.

For preprocessing, we truncate the text if its length is larger than L and pad zero if the length is smaller than L. Let $\boldsymbol{x_j^i} \in \mathbb{R}^d$ represent the word embedding of the j-th word in the text of post p_i. The text representation of p_i is thus represented as

$$\boldsymbol{X^i_{1:L}} = [\boldsymbol{x_1^i}; \boldsymbol{x_2^i}; ...; \boldsymbol{x_L^i}] \quad (7)$$

where $\boldsymbol{X^i_{1:L}} \in \mathbb{R}^{L\times d}$.

Then, we refine word embeddings using the multi-head self-attention [Vaswani *et al.*, 2017] to capture dependencies between words. Each of the heads captures a hidden relationship from a specific aspect. Considering a self-attention with h heads. There are three input matrices for the j-th head which are the query matrix $\boldsymbol{Q_j} \in \mathbb{R}^{L\times \frac{d}{h}}$, the key matrix $\boldsymbol{K_j} \in \mathbb{R}^{L\times \frac{d}{h}}$ and the value matrix $\boldsymbol{V_j} \in \mathbb{R}^{L\times \frac{d}{h}}$. Take $\boldsymbol{X^i_{1:L}} \in \mathbb{R}^{L\times d}$ as an example. For simplicity, we denote $\boldsymbol{X^i_{1:L}}$ as $\boldsymbol{X}$. There are $\boldsymbol{K_j} = \boldsymbol{X}\boldsymbol{W_j^K}$, $\boldsymbol{Q_j} = \boldsymbol{X}\boldsymbol{W_j^Q}$ and $\boldsymbol{V_j} = \boldsymbol{X}\boldsymbol{W_j^V}$ with $\{\boldsymbol{W_j^Q}, \boldsymbol{W_j^K}, \boldsymbol{W_j^V}\} \in \mathbb{R}^{d\times \frac{d}{h}}$. Besides, we denote the output of the self-attention with h heads as $\boldsymbol{Z} = [\boldsymbol{Z_1}; \boldsymbol{Z_2}; ...; \boldsymbol{Z_h}]$. The output of the j-th head is calculated as

$$\boldsymbol{Z_j} = Attention(\boldsymbol{Q_j}, \boldsymbol{K_j}, \boldsymbol{V_j}) = softmax(\frac{\boldsymbol{Q_j}\boldsymbol{K_j^T}}{\sqrt{d}})\boldsymbol{V_j} \quad (8)$$

with $\boldsymbol{Z_j} \in \mathbb{R}^{L\times \frac{d}{h}}$. Then $\boldsymbol{Z}$ is calculated as

$$\boldsymbol{Z} = MultiHead(\boldsymbol{X}, \boldsymbol{X}, \boldsymbol{X}) = Concat(\boldsymbol{Z_1}, ..., \boldsymbol{Z_h})\boldsymbol{W^O} \quad (9)$$

where $\boldsymbol{W^O} \in \mathbb{R}^{d\times d}$ and $\boldsymbol{Z} \in \mathbb{R}^{L\times d}$.

We then capture the semantic representation of a post with CNN. $\boldsymbol{X^i_{e:e+k-1}}$ is convolved with a filter $\boldsymbol{W} \in \mathbb{R}^{k\times d}$ where k is the size of the receptive field. The feature t_j is generated from a window of word embeddings $\boldsymbol{X^i_{e:e+k-1}}$

$$t_j = \sigma(\boldsymbol{W} * \boldsymbol{X^i_{e:e+k-1}} + b) \quad (10)$$

where $*$ is the convolution operation and $b \in \mathbb{R}$ is the bias term. σ is an activation function such as $tanh$. The filter $\boldsymbol{W}$ is applied to each possible window of word embeddings in $\boldsymbol{X^i_{1:L}}$ to produce a feature map $\boldsymbol{t}$ for p_i

$$\boldsymbol{t} = [t_1, t_2, ..., t_{L-k+1}] \quad (11)$$

where $\boldsymbol{t} \in \mathbb{R}^{L-k+1}$. Then a max pooling with a stride of $L - k + 1$ is applied over the feature map:

$$\hat{t} = max\{\boldsymbol{t}\}. \quad (12)$$

We use $d/3$ filters with varying receptive filed $k \in \{5, 6, 7\}$ to obtain the semantics from different granularities. Each of the receptive fileds corresponds to a feature vector of length $d/3$. Next, we concatenate all feature vectors to form $\boldsymbol{m_i} \in \mathbb{R}^d$, which is the final text representation of post p_i.

4.4 Classification with Adversarial Learning

Here we describe how to combine the structure and text representations for classification.

Objective of Optimization

For post p_i, we have obtained its structure features $\boldsymbol{h_i'}$ and $\boldsymbol{h_i}$ from the camouflaged graph and the non-camouflaged graph respectively, as well as its text representation $\boldsymbol{m_i}$.

Then we concatenate them to form two feature vectors of p_i, which are $\boldsymbol{P_i'} = [\boldsymbol{h_i'}; \boldsymbol{m_i}]$ and $\boldsymbol{P_i} = [\boldsymbol{h_i}; \boldsymbol{m_i}]$. Next, we feed $\boldsymbol{P_i'}$ and $\boldsymbol{P_i}$ into the fully-connected layers respectively and obtain

$$\hat{\boldsymbol{y}}_{\boldsymbol{i}}' = softmax(\boldsymbol{W_1}^{\mathrm{T}}\boldsymbol{P_i'} + \boldsymbol{b_1}) \tag{13}$$

$$\hat{\boldsymbol{y}}_{\boldsymbol{i}} = softmax(\boldsymbol{W_2}^{\mathrm{T}}\boldsymbol{P_i} + \boldsymbol{b_2}) \tag{14}$$

where $\{\boldsymbol{W_1}, \boldsymbol{W_2}\} \in \mathbb{R}^{(d+d')\times|c|}$ are the weight parameters and $\{\boldsymbol{b_1}, \boldsymbol{b_2}\} \in \mathbb{R}^{|c|}$ are the bias terms. $\hat{\boldsymbol{y}}_{\boldsymbol{i}}'$ and $\hat{\boldsymbol{y}}_{\boldsymbol{i}}$ are the predicted probability distributions of p_i over two classes.

Then we use the cross-entropy loss as the optimization objective which consists of two parts. One is the standard loss $\mathcal{L}_1$ based on the original adjacency matrix $\boldsymbol{A}$. The other loss $\mathcal{L}_2$ is based on the camouflaged adjacency matrix $\boldsymbol{A'}$. Therefore, the overall loss $\mathcal{L}$ is

$$\mathcal{L} = \underbrace{\sum_{i=1}^{N}\sum_{c\in\{0,1\}} -\boldsymbol{y_i} log\hat{\boldsymbol{y}}_{\boldsymbol{i}}}_{\mathcal{L}_1(\theta_d)} + \beta \underbrace{\sum_{i=1}^{N}\sum_{c\in\{0,1\}} -\boldsymbol{y_i} log\hat{\boldsymbol{y}}_{\boldsymbol{i}}'}_{\mathcal{L}_2(\theta_s,\theta_d)} \tag{15}$$

where $\boldsymbol{y_i}$ is $[1, 0]$ when p_i is a rumor post and $[0, 1]$ when it is a non-rumor post. N is the number of training data. θ_s denotes parameters in the camouflaged adjacency matrix $\boldsymbol{A'}$ except the node embedding matrix $\boldsymbol{B}$. The remaining parameters are denoted as θ_d. β is a hyperparameter to control the scale of $\mathcal{L}_2$.

Adversarial Learning

Given the optimization object $\mathcal{L}$ in Eq. 15, we adopt an adversarial learning method to infer the parameters. It can be viewed as playing a minimax game as follows

$$\min_{\theta_d}\max_{\theta_s} \mathcal{L}(\theta_s; \theta_d) \tag{16}$$

The overall training process is shown in Algorithm 1. In each iteration, we first optimize the parameters θ_s in Eq. 2 to generate a camouflaged graph $\boldsymbol{A'}$ that can cause the largest classification loss $\mathcal{L}$ (Lines 7-9). We then optimize the other parameters θ_d to minimize $\mathcal{L}$ (Lines 10-12), encouraging the model to make the correct classification.

Algorithm 1 Model training with a minimax game

Input: Graph $G = (V, E, A)$, a set of post contents T, learning rate ϵ
Parameter: θ_s and θ_d

1: **for** epoch from 1 to maxEpoch **do**
2: **for** $G = (V, E, A)$ and a batch of T **do**
3: Compute the camouflage adjacency matrix A' using Eq. 1;
4: Extract structure features using Eq. 6;
5: Generate text representations using Eq. 12;
6: Compute CrossEntropy loss $\mathcal{L}$ using Eq. 15;
7: /* Maximize $\mathcal{L}$ w.r.t.θ_s */
8: Compute gradient $\nabla(\theta_s)$;
9: Update: $\theta_s \leftarrow \theta_s + \epsilon\nabla(\theta_s)$;
10: /* Minimize $\mathcal{L}$ w.r.t. θ_d */
11: Compute gradient $\nabla(\theta_d)$;
12: Update: $\theta_d \leftarrow \theta_d - \epsilon\nabla(\theta_d)$;
13: **end for**
14: **end for**

5 Experiments

5.1 Datasets

We evaluate our model on two real-word datasets: Weibo [Song *et al.*, 2018] and Twitter [Zubiaga *et al.*, 2016], which were collected from the most influential social media in China and the U.S respectively. For a specific post, both of the datasets contain two labels, i.e., rumor (R) or non-rumor (N), and also contain rich information such as post texts, comments, and user profiles. Table 2 shows the statistics of the datasets.

Statistic	Source Tweets	Rumors	Non-rumors	Users	Comments
Weibo	3,387	1,838	1,849	1,067,410	1,275,180
Twitter	5,802	1,972	3,830	49,345	97,410

Table 2: Statistic of Datasets.

5.2 Baselines

We compare the following baseline models with our model:

HAN [Yang *et al.*, 2016] applies a hierarchical attention network to the post contents at the word-level and sentence-level.

Text-CNN [Kim, 2014] uses convolutional neural networks to capture text semantics for classification tasks.

GRU [Ma *et al.*, 2016] adopts recurrent neural networks for learning hidden representations to capture the variation of contextual information of relevant posts over time.

RvNN [Ma *et al.*, 2018] adopts two recursive neural models based on a bottom-up and a top-down tree-structured neural networks, which are denoted as RvNN$_{BU}$ and RvNN$_{TD}$.

EANN [Wang *et al.*, 2018] is a GAN-based model that can extract event-invariant features and thus benefit the detection of newly arrived events. Note that different from the original EANN, we don't consider pictures as input due to the lack of pictures in our dataset.

GAN-GRU [Ma *et al.*, 2019] is a GAN-based model where a generator will produce conflicting voices to force the discriminator to learn stronger rumor indicative representations.

GLAN [Yuan *et al.*, 2019] adopts a global-local attention network for rumor detection, which jointly encodes the local semantic and the global structural information.

Method	Weibo					Twitter				
	Class	Acc.	Prec.	Rec.	F_1	Class	Acc.	Prec.	Rec.	F_1
GRU	R	0.839	0.757	0.789	0.773	R	0.852	0.773	0.784	0.779
	N		0.885	0.865	0.875	N		0.892	0.886	0.889
Text-CNN	R	0.807	0.802	0.748	0.774	R	0.839	0.757	0.789	0.773
	N		0.811	0.853	0.831	N		0.885	0.865	0.875
HAN	R	0.833	0.924	0.704	0.799	R	0.851	0.757	0.789	0.773
	N		0.782	0.949	0.857	N		0.885	0.865	0.875
RvNN$_{BU}$	R	0.903	0.909	0.859	0.884	R	0.789	0.782	0.793	0.788
	N		0.894	0.935	0.916	N		0.795	0.784	0.790
RvNN$_{TD}$	R	0.847	0.810	0.843	0.826	R	0.824	0.829	0.817	0.823
	N		0.877	0.851	0.864	N		0.818	0.830	0.824
EANN	R	0.866	0.872	0.808	0.838	R	0.794	0.811	0.735	0.771
	N		0.863	0.911	0.886	N		0.782	0.847	0.813
GAN-GRU	R	0.867	0.854	0.859	0.856	R	0.783	0.761	0.825	0.792
	N		0.878	0.874	0.876	N		0.809	0.741	0.773
GLAN	R	0.902	0.917	0.871	0.893	R	0.853	**0.871**	0.654	0.747
	N		0.891	0.931	0.910	N		0.847	**0.952**	0.896
CGAT	R	**0.940**	**0.959**	**0.906**	**0.932**	R	**0.892**	0.823	**0.871**	**0.846**
	N		**0.925**	**0.968**	**0.946**	N		**0.931**	0.903	**0.917**

Table 3: The performance results of the comparison methods on two datasets.

Method		CGAT-TEXT	CGAT-ADJ	CGAT-GP	CGAT-M	CGAT
Weibo	Acc.	0.882	0.910	0.912	0.913	**0.940**
	F_1	0.882	0.909	0.917	0.912	**0.939**
Twitter	Acc.	0.854	0.869	0.872	0.871	**0.892**
	F_1	0.823	0.846	0.854	0.845	**0.882**

Table 4: Experimental results of the variations of CGAT.

CGAT is our proposed model, which is short for simulating **C**amouflages on **G**raph with **A**dversarial **T**raining

Among the above baselines, Text-CNN, GRU, HAN, and RvNN are deep learning methods. EANN and GAN-GRU deal with rumor contents based on the idea of the generative adversarial network. GLAN combines text contents with social network structures to detect rumors.

We split the datasets for training, developing, and testing with a ratio of 7:1:2. We adopt the accuracy, precision, recall, and F1 score as the evaluation metrics. We use the Adam algorithm [Kingma and Ba, 2015] to optimize our objective function with $\beta_1 = 0.9$ and $\beta_2 = 0.999$. The learning rate used in the training process is 0.002.

5.3 Results and Discussion

Table 3 shows the performance of the comparison methods. It can be observed that CGAT significantly outperforms all the baselines which confirms that considering the camouflages on the graph would benefit the rumor detection task.

Most baseline models cannot perform well on both of the two datasets. The possible reason is these models are based on text representations, and it is hard to design a good text model that can capture the semantics across two very different languages. On the contrary, GLAN and CGAT also consider social structures which are independent of languages, and thus achieve consistent performance improvements.

Besides our model and GLAN, GAN-GRU and RvNN$_{BU}$ perform better than the other baselines on Weibo dataset. The reason is that, in addition to post contents, they also involve the user comments for detection. However, they perform not very well on Twitter dataset. The possible reason is that according to Table 2, the number of comments on Twitter is less than that in Weibo, limiting the benefits of the comments.

5.4 Performance of the Variations

To show the effectiveness of different components in CGAT, we compare it with the following variations:

CGAT-TEXT only uses rumor texts. As all the other variations utilize texts, we omit it in the remaining for simplicity.

CGAT-ADJ only uses the structure features from the original graph, without involving the camouflaged graph.

CGAT-GP uses another graph adversarial attack method [Bojchevski and Günnemann, 2019] to perturb the graph structure $\boldsymbol{A}$, and the resulting graph takes the role of $\boldsymbol{A'}$ in CGAT.

CGAT-M doesn't use the action mask matrix $\boldsymbol{M}$ and thus enables edges are added or removed between any two nodes.

The comparison results are shown in Table 4. It can be observed that: (i) all the other methods outperform CGAT-TEXT, indicating the structure features are important for rumor detection; (ii) models with the perturbed graph structure outperform CGAT-ADJ, which shows the benefits of the adversarial attacks on the graph; (iii) CGAT-M and CGAT-GP are only a little better than CGAT-ADJ, but much worse than CGAT, indicating that attacking the graph structures without domain constraints would introduce unnecessary noises in the model and thus limit the performance improvements.

6 Conclusion

In this paper, we propose a graph-based rumor detection method which takes into account the camouflages of rumors from an adversarial perspective. By dynamically generating the perturbations on the heterogeneous social graph with domain constrains, we learn to extract more distinctive structure features, which cooperates with text representations to improve the performance of the model. Evaluations with both English and Chinese rumor datasets demonstrate our model can outperform the state-of-the-art baselines.

Acknowledgments

This work was supported by the National Key Research and Development Program of China (No.2017YFB0803301) and Natural Science Foundation of China (No.61976026, No.U1836215) and 111 Project (B18008).

References

[Allcott and Gentzkow, 2017] Hunt Allcott and Matthew Gentzkow. Social media and fake news in the 2016 election. *Journal of economic perspectives*, 31(2):211–36, 2017.

[Bojchevski and Günnemann, 2019] Aleksandar Bojchevski and Stephan Günnemann. Adversarial attacks on node embeddings via graph poisoning. In *ICML*, pages 695–704, 2019.

[Castillo *et al.*, 2011] Carlos Castillo, Marcelo Mendoza, and Barbara Poblete. Information credibility on twitter. In *WWW*, pages 675–684, 2011.

[Dai *et al.*, 2018] Hanjun Dai, Hui Li, Tian Tian, Xin Huang, Lin Wang, Jun Zhu, and Le Song. Adversarial attack on graph structured data. In *ICML*, pages 1123–1132, 2018.

[Dai *et al.*, 2019] Quanyu Dai, Xiao Shen, Liang Zhang, Qiang Li, and Dan Wang. Adversarial training methods for network embedding. In *WWW*, pages 329–339, 2019.

[Goodfellow *et al.*, 2014] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron C. Courville, and Yoshua Bengio. Generative adversarial nets. In *NIPS*, pages 2672–2680, 2014.

[Khattar *et al.*, 2019] Dhruv Khattar, Jaipal Singh Goud, Manish Gupta, and Vasudeva Varma. MVAE: multimodal variational autoencoder for fake news detection. In *WWW*, pages 2915–2921, 2019.

[Kim, 2014] Yoon Kim. Convolutional neural networks for sentence classification. In *EMNLP*, pages 1746–1751, 2014.

[Kingma and Ba, 2015] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015.

[Kipf and Welling, 2017] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *ICLR*, 2017.

[Kwon *et al.*, 2017] Sejeong Kwon, Meeyoung Cha, and Kyomin Jung. Rumor detection over varying time windows. *PloS one*, 12(1), 2017.

[Liu and Wu, 2018] Yang Liu and Yi-fang Brook Wu. Early detection of fake news on social media through propagation path classification with recurrent and convolutional networks. In *AAAI*, pages 354–361, 2018.

[Ma *et al.*, 2015] Jing Ma, Wei Gao, Zhongyu Wei, Yueming Lu, and Kam-Fai Wong. Detect rumors using time series of social context information on microblogging websites. In *CIKM*, pages 1751–1754, 2015.

[Ma *et al.*, 2016] Jing Ma, Wei Gao, Prasenjit Mitra, Sejeong Kwon, Bernard J. Jansen, Kam-Fai Wong, and Meeyoung Cha. Detecting rumors from microblogs with recurrent neural networks. In *IJCAI*, pages 3818–3824, 2016.

[Ma *et al.*, 2017] Jing Ma, Wei Gao, and Kam-Fai Wong. Detect rumors in microblog posts using propagation structure via kernel learning. In *ACL*, pages 708–717, 2017.

[Ma *et al.*, 2018] Jing Ma, Wei Gao, and Kam-Fai Wong. Rumor detection on twitter with tree-structured recursive neural networks. In *ACL*, pages 1980–1989, 2018.

[Ma *et al.*, 2019] Jing Ma, Wei Gao, and Kam-Fai Wong. Detect rumors on twitter by promoting information campaigns with generative adversarial learning. In *WWW*, pages 3049–3055, 2019.

[Song *et al.*, 2018] Changhe Song, Cunchao Tu, Cheng Yang, Zhiyuan Liu, and Maosong Sun. CED: credible early detection of social media rumors. *CoRR*, abs/1811.04175, 2018.

[Vaswani *et al.*, 2017] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NIPS*, pages 5998–6008, 2017.

[Veličković *et al.*, 2018] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph Attention Networks. *ICLR*, 2018.

[Wang *et al.*, 2018] Yaqing Wang, Fenglong Ma, Zhiwei Jin, Ye Yuan, Guangxu Xun, Kishlay Jha, Lu Su, and Jing Gao. EANN: event adversarial neural networks for multi-modal fake news detection. In *KDD*, pages 849–857, 2018.

[Wu *et al.*, 2019] Huijun Wu, Chen Wang, Yuriy Tyshetskiy, Andrew Docherty, Kai Lu, and Liming Zhu. Adversarial examples for graph data: Deep insights into attack and defense. In *IJCAI*, pages 4816–4823, 2019.

[Xu *et al.*, 2019] Kaidi Xu, Hongge Chen, Sijia Liu, Pin-Yu Chen, Tsui-Wei Weng, Mingyi Hong, and Xue Lin. Topology attack and defense for graph neural networks: An optimization perspective. In *IJCAI*, pages 3961–3967, 2019.

[Yang *et al.*, 2012] Fan Yang, Yang Liu, Xiaohui Yu, and Min Yang. Automatic detection of rumor on sina weibo. In *Proceedings of the ACM SIGKDD Workshop on Mining Data Semantics*, pages 1–7, 2012.

[Yang *et al.*, 2016] Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alexander J. Smola, and Eduard H. Hovy. Hierarchical attention networks for document classification. In *NAACL*, pages 1480–1489, 2016.

[Yuan *et al.*, 2019] Chunyuan Yuan, Qianwen Ma, Wei Zhou, Jizhong Han, and Songlin Hu. Jointly embedding the local and global relations of heterogeneous graph for rumor detection. In *ICDM*, pages 796–805, 2019.

[Zhang *et al.*, 2018] Jiawei Zhang, Limeng Cui, Yanjie Fu, and Fisher B. Gouza. Fake news detection with deep diffusive network model. *CoRR*, abs/1805.08751, 2018.

[Zhao *et al.*, 2015] Zhe Zhao, Paul Resnick, and Qiaozhu Mei. Enquiring minds: Early detection of rumors in social media from enquiry posts. In *WWW*, pages 1395–1405, 2015.

[Zubiaga *et al.*, 2016] Arkaitz Zubiaga, Maria Liakata, and Rob Procter. Learning reporting dynamics during breaking news for rumour detection in social media. *CoRR*, abs/1610.07363, 2016.