Since January 2020 Elsevier has created a COVID-19 resource centre with free information in English and Mandarin on the novel coronavirus COVID-19. The COVID-19 resource centre is hosted on Elsevier Connect, the company's public news and information website.

Elsevier hereby grants permission to make all its COVID-19-related research that is available on the COVID-19 resource centre - including this research content - immediately available in PubMed Central and other publicly funded repositories, such as the WHO COVID database with rights for unrestricted research re-use and analyses in any form or by any means with acknowledgement of the original source. These permissions are granted for free by Elsevier for as long as the COVID-19 resource centre remains active.
Rumor detection based on propagation graph neural network with attention mechanism

Zhiyuan Wu\textsuperscript{a}, Dechang Pi\textsuperscript{a}, Junfu Chen\textsuperscript{a}, Meng Xie\textsuperscript{a}, Jianjun Cao\textsuperscript{b,\textast}}

\textsuperscript{a}College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing, Jiangsu, China
\textsuperscript{b}The Sixty-third Research Institute, National University of Defence Technology, Nanjing, Jiangsu, China

**Abstract**

Rumors on social media have always been an important issue that seriously endangers social security. Researches on timely and effective detection of rumors have aroused lots of interest in both academia and industry. At present, most existing methods identify rumors based solely on the linguistic information without considering the temporal dynamics and propagation patterns. In this work, we aim to solve rumor detection task under the framework of representation learning. We first propose a novel way to construct the propagation graph by following the propagation structure (who replies to whom) of posts on Twitter. Then we propose a gated graph neural network based algorithm called PGNN, which can generate powerful representations for each node in the propagation graph. The proposed PGNN algorithm repeatedly updates node representations by exchanging information between the neighbor nodes via relation paths within a limited time steps. On this basis, we propose two models, namely GLO-PGNN (rumor detection model based on the \textit{global} embedding with \textit{propagation graph neural network}) and ENS-PGNN (rumor detection model based on the \textit{ensemble learning} with \textit{propagation graph neural network}). They respectively adopt different classification strategies for rumor detection task, and further improve the performance by including attention mechanism to dynamically adjust the weight of each node in the propagation graph. Experiments on a real-world Twitter dataset demonstrate that our proposed models achieve much better performance than state-of-the-art methods both on the rumor detection task and early detection task.

1. Introduction

The popularization of the mobile Internet has dramatically changed the way in which people access news. According to a report released by the Kantar Media in 2019, 40% of the world's people use social media (Kantar Media, 2019). Social media such as Twitter, Facebook, Weibo, etc., enable Internet users to access a wide variety of news at a lower cost, while also facilitate the widespread dissemination of fake news or rumors. A report issued by the Weibo official account shows that, the Weibo platform effectively handled 77,742 false information in 2019 (Weibo, 2019). The untrue information seriously misleads people's mind and even endanger public safety.

The term rumor commonly refers to the information that emerge and spread among people in the absence of factual support (Sharma et al., 2019). Rumor is very similar to fake news, the main difference is that, rumor is the unverified information, they are not necessarily false and may turn out to be true or false, or may remain unresolved, while fake news is the false information usually spread through news outlets. Since rumor and fake news are closely related in characteristics, the rumor detection is also very similar to fake news detection in technique. The rumor detection commonly seeks to distinguish between verified information and unverified information, sometimes it further classifies the unverified information. The fake news detection seeks to distinguish between true information and false information. The false rumor on social media could disturb people's minds, mislead public opinion, more seriously, undermine the government's credibility and cause social shocks. Therefore, the detection of rumors on social networks has become a social security issue that cannot be ignored.

Nevertheless, rumor detection is a challenging task due to the following three aspects: 1) The non-trivial work to process huge information. News about hot events is emerging endlessly on social networks, and the contents cover all areas. A large amount of information needs to be processed in order to identify different type of
rums; 2) The demand for real-time detection. Users on social networks are particularly active, which enables a variety of information to spread widely in a short time. It is essential to identify rumors as early as possible, since the adverse effect caused by false rumors increase over time; 3) The confusing nature of rumors. Some rumors are intentionally well-designed to imitate real news for various motivations, such as political astroturfing, conducting malicious marketing management. It is difficult for ordinary people or even domain experts to distinguish between true rumors and false rumors.

Aiming at identifying rumors on social networks, many studies have been done on rumor detection. Most proposed approaches copy with rumors rely on text or images (Mihalcea & Strapparava, 2009; Wang, 2017; Vishwakarma, Varshney, & Yadav, 2019). However, some rumors are deliberately crafted to imitate real stories, it is challenging to determine veracity based on the features extracted from text or images. As time goes on, rumors form its specific propagation structure after being commented and replied. Many authors have worked on rumor detection problem by exploiting the propagation characteristic (Ma, Gao, Wei, Lu, & Wong, 2015; Ma, Gao, & Wong, 2018b). However, these works study the propagation patterns based on the contextual information extracted from posts in chronological order, they only capture the temporal characteristic of rumors. We argue that, in addition to the temporal characteristic, the topological characteristic is also very important when exploiting the propagation characteristic of rumors. We propose a novel approach to tackle the problem of rumor detection. We first use a novel way to construct the propagation graph based on the non-sequential propagation structure of posts on social network, then we propose a representation learning algorithm called PGNN based on gated graph neural network, which can learn powerful representations for each node in the propagation graph. On the basis of PGNN, we propose two different models for rumor detection task: GLO-PGNN and ENS-PGNN. The former identifies rumors based on the global embedding of the propagation graph, while the latter first calculates the prediction probability for each node and then summarizes to obtain the final result.

We empirically evaluate our proposed models on a public dataset from Twitter. The experimental results demonstrate that our models are superior to the state-of-the-art rumor detection models on multiple evaluation indicators. In addition, our models also have extremely good early stop performance, which is essential in the application scenario of real-time rumor detection. Our contributions are summarized in the following three folds:

- We propose a novel way to explicitly construct the propagation graph by following the propagation structure of posts on social network. We propose a representation learning algorithm called PGNN (propagation graph neural network) based on gated graph neural network. The proposed algorithm can embed textual and structural features into the high-level representations by propagating information between neighbor nodes in the propagation graph.
- On the basis of PGNN, we propose two rumor detection models, GLO-PGNN (rumor detection model based on the global embedding with propagation graph neural network) and ENS-PGNN (rumor detection model based on the ensemble learning with propagation graph neural network), which mainly differ in classification approaches. We also include attention mechanism into our proposed models in order to achieve significant performance improvement.
- The experimental results on the public dataset demonstrate that our models are superior to state-of-the-art baselines on both rumor detection task and early detection task.

The organization of this paper is as follows: Section 1 is the introduction. Section 2 introduces the related works. Section 3 first presents some definitions and symbolic markers used in this paper, and then describes the way to construct the propagation graph. Section 4 explains the algorithms we proposed in detail. Section 5 shows and analyzes the experimental results in detail. Section 6 describes concluding remarks and future work.

2. Related work

Currently, most rumor detection methods and fake news detection methods are supervised. The most common type is the content-based methods. The content-based methods classify rumors or fake news depending on the veracity of text or images. These work assume that, the content in different types of rumors (or news) differ in some quantifiable way. (Fuller, Biros, & Wilson, 2009) previously proposed the concept of cue sets based on several semantic cues including percentage of first person singular and plural pronouns in the text, the average length of words in the text and imagery. (Zhao, Resnick, & Mei, 2015) tried a more elaborate set of cues to make it more suitable for fake news detection, and included several platform-specific features, such as counts of hashtags# and mentions@ in tweets. (Mihalcea & Strapparava, 2009) applied the n-gram approach into lie detection task and represented the text content as word frequency vectors, which can be used as inputs of some machine learning classifiers. (Sicilia, Giudice, Pei, Pechenizkiy, & Soda, 2018) used content-based features, together with some fine-grained features inspired by the graph theory and the social influence models to detect rumor in each post of a single topic domain related to health news. However, the cue and feature based methods mentioned above require a lot of time and energy to design non-trivial linguistic cue set, and the hand-crafted features rely heavily on specific social platform, making it hard to generalize across different domains, languages and topics. On social networks, news or rumors often contain pictures. (Vishwakarma et al., 2019) proposed a fake news authentication system aiming at analyzing the veracity of platform-independent information available in the form of images. The model identifies news in 4 steps, firstly it extracts text from images, secondly it recognizes entities from the text, and then it scrapes the web for related content according to the extracted entities, finally a processing unit is responsible for the classification. In order to alleviate the shortcomings of traditional content-based methods that requiring manual feature engineering, some researchers automatically extracted features by including deep learning methods. (Wang, 2017) processed the statement text and the speaker meta-data by two different embedding layers, so as to obtain continuous low dimensional feature representations. (Shu, Wang, & Liu, 2019) assumed that publisher-news relations and user-news interactions are inherently associated. They considered fake news detection task in a tri-relationship embedding framework. (Kalijar, Goswami, Narang, & Sinha, 2020) proposed a deep convolutional neural network (FNDNet) for content-based fake news detection. FNDNet is designed to automatically learn the discriminatory features for fake news classification through multiple hidden layers built in the deep neural network. However, some rumors or fake news are deliberately crafted to imitate real stories for a variety of motivations, it is challenging for machine learning algorithms to determine veracity by solely analyzing contents. Some methods sort a series of relevant posts in chronological order and then identify rumors by capturing the temporal dynamics of the time series. (Ma et al., 2015) divided the time series into fixed time intervals, and captured the temporal characteristics by comparing the difference of social context information between adjacent time intervals. (Ma, Gao, Mitra, Kwon, Jansen, Wong,
Yang et al., 2018) further enriched the features of each time intervals and obtained users’ credibility scores by matrix decomposition. However, these approaches typically focus on temporal dynamics alone, ignoring the internal topology among posts on social networks (who replies to whom and who comments on whom).

Information on social media with different contents differs in the temporal patterns and diffusion speeds (Foroozani & Ebrahimi, 2019). Rumor and non-rumor, corresponding to unverified information and verified information respectively, spread through social media in the form of shares and re-shares of the source and shared posts, resulting in a diffusion cascade or tree, with the source post at the root. To figure out how information is being misused and disseminated to fulfill spiteful motives, (Meel and Vishwakarma, 2019) systematically studied the prevalent technologies used to detect and contain malicious information, they further gave a taxonomy for classifying spiteful information as different stages. Some work have designed many hand-crafted features to exploit the propagation tree of rumors. (Wu, Yang, & Zhu, 2015) and (Ma, Gao, & Wong, 2017) calculated similarity between different propagation trees based on well-designed random walk graph kernels and tree kernels respectively. (Vosoughi, 2017) proposed a system to automatically predict the veracity of rumors, the system identifies rumors based on 3 aspects, namely, linguistic style, characteristics of people involved in propagating information, and network propagation dynamics. To reduce the time and cost of manually designing features, (Ma, Gao, & Wong, 2018b) proposed two recursive neural models for rumor detection. The models automatically extract features by recursively traversing the tree structure in either the top-down or bottom-up manner.

The aforementioned approaches belong to the scope of supervised learning. In addition, some scholars have worked on how to use unsupervised methods to identify the veracity of unconfirmed news or rumors. However, research on unsupervised methods has encountered extremely hard difficulties and very few research findings are achieved in this field. For example, (Yang et al., 2019) captured the conditional dependencies among the veracity of news, users' opinions and users' credibility based on a Bayesian network model, and they proposed an efficient sampling method, assisting model to determine the veracity of unlabeled news. Inspired by these ideas, we propose a novel approach to tackle the problem of rumor detection. In our proposed model, we use word embedding technique to obtain continuous low dimensional representations for text content, which is a common practice in deep learning-based rumor detection approaches. And unlike those work simply considering the temporal characteristics of rumors, we exploit the topological characteristics by taking into consideration the who-replies-to-whom relationship. Instead of designing features manually for the propagation tree of rumors, we construct the propagation graph of rumors and automatically extract features through graph neural network. To the best of our knowledge, our work is the first attempt to tackle the rumor detection problem by exploiting the propagation graph of rumors.

3. Preliminary

In this section, we first introduce some definitions and symbolic markers used in the paper. Then we discuss how to construct the propagation graph according to who-replies-to-whom relationship. At last, we formally give the problem statement of rumor detection task.

3.1. Related definitions

When someone posts a tweet on Twitter, other Twitter users can express their stance by sharing, replying, commenting or retweeting, so that the information about the source post can widely spread on the Internet. For a better understanding of the propagation structures of posts, some definitions and symbolic markers used in this paper are described as follows:

Definition 1 ((Source tweet)). Source tweet is an original tweet that is not the reply to any other tweets. We use $s^0$ to represent source tweet, superscript $j$ indicates the index.

Definition 2 ((Responsive tweet)). Responsive tweet refers to the tweets which are the replies to the source tweet or other responsive tweets. We use $x^{ij}$ to represent the $i$-th responsive tweet by time which is relevant to the source tweet $s^0$.

Definition 3 ((Post set)). Each post set $C^0$ consists of a source tweet and all its responsive tweets, for example, $C^0 = \{s^0, x^{10}, x^{20}, \ldots, x^{0j}\}$. Each post set can form a propagation tree structure based on who-replies-to-whom relationships (Wu et al. 2015, Ma et al., 2017). Fig. 1(a) shows the connection structure based on who-replies-to-whom relationships after the ACM official Twitter account posted the winner of the 2019 Turing Award. Fig. 1(b) shows the propagation tree structure derived from Fig. 1(a), root node 0 represents the source tweet, nodes with indices from 1 to 3 represent the responsive tweets, all the nodes in the propagation tree form a post set.

In this paper, we extend the original definition of the propagation tree to form a propagation graph structure. Each propagation graph consists of several nodes and two different types of relation paths. Node $v_i$ in propagation graph $G^0$ corresponds to tweet $x^{ij}$ in the post set $C^0$. Relation paths can be classified into two different types, namely explicit relation path and implicit relation path.

Definition 4 ((Explicit relation path)). For tweets $x^{ij}$ and $x^{i'j}(i < i')$, if there exists the reply relationship between them, then the propagation graph $G^0$ has an explicit relation path with the direction from node $v_i$ to node $v_{i'}$. For example, as shown in Fig. 1(c), there is an explicit relation path from node 2 to node 4 since user 4 replies to user 2.

Definition 5 ((Implicit relation path)). In order to form a graph structure, we define that each implicit relation path in propagation graph corresponds to an implicit relation path with the opposite direction. For example, as shown in Fig. 1(c), the solid arrow and dotted arrow represent explicit relation path and implicit relation path respectively.

Definition 6 ((propagation graph)). With the definition of relation path, we can extend the homogeneous tree structure into a heterogeneous graph structure $G^0 = (V^0, \Phi, \Psi, V^0)$ is the set of all nodes in the propagation graph, where each node $v_i$ corresponds to tweet $x^{ij}$ in the post set $C^0$. $E^0 \subseteq V^0 \times V^0$ is the set of all edges. $\Phi$ and $\Psi$ are mapping functions, they are generic in different propagation...
graphs instead of being specific to a certain one. Mapping function \( \Phi : V \rightarrow H \) can generate high-level representation for each node according to its corresponding tweet content, for example, \( \forall v^{(i)} \in V^{(i)}, \) its textual representation can be calculated as \( \Phi(v^{(i)}) \in \mathbb{R}^d, d \) is the dimension. Function \( \Psi : E \rightarrow R \) is a mapping from edges set \( E \) to the set of all relation paths \( R \). The set \( R = \{ r_1, r_2 \} \) consists of two elements, \( r_1 \) represents the explicit relation path and \( r_2 \) represents the implicit relation path.

According to the way the propagation graph is constructed, the Twitter rumor detection dataset can be defined as a set of propagation graphs \( G = \{ G_1, G_2, \ldots, G_N \} \).

### 3.2. Problem statement

Rumor detection commonly can be seen as a multi-classification task under the framework of supervised learning. Traditional methods simply classify source tweets as non-rumors and rumors by distinguishing between verified information and unverified information. Rumors belong to the unverified information category, in spite of that, they are not necessarily false and may turn out to be true or false, or may remain unresolved. In this paper, we consider a more fine-grained classification. We assume that the source tweets can be divided into four classes: 1) non-rumors, namely verified information, 2) true rumors, namely the unverified information that turns out to be true 3) false rumors, namely the unverified information that turns out to be false, 4) unconfirmed rumors, namely the unverified information that remains unresolved. These four categories can be respectively represented as \( N, T, F \) and \( U \) for short (Ma et al., 2017). Take as an example the tweets related to coronavirus COVID-19. The tweets that “person-to-person contact is the main method of transmission” is a non-rumor, since many authoritative news media have reported relevant news and WHO has acknowledged that. The tweet that “coronavirus is caused by 5G technology” is absolutely a false rumor, it not only lacks the factual support, but also deviates from the scientific principles.

Therefore, our goal is to learn a classifier from the labeled propagation graph set, that is \( f : G^{(i)} \rightarrow Y^{(i)} \), where \( Y^{(i)} \) takes one of four finer-grained classes: \( N, T, F \) and \( U \). Given the propagation graph \( G^{(i)} \), the classifier \( f \) can output the classification result for source tweet \( s^{(i)} \).

### 4. The proposed models

In this section, we elaborate each module of our proposed models. We first detail the text embedding algorithm adopted in our models in Section 4.1. Then we discuss how to aggregate feature information from neighborhood with the proposed PGNN algorithm in Section 4.2. At last, we discuss how to identify rumors based on the representations with two different strategies in Section 4.3.

#### 4.1. Text embedding

It is hard to deal with discrete words in tweets directly, so we want to embed the information of tweets content into low-dimensional space. Our goal is to learn a mapping function \( \Phi : V \rightarrow H \) in order to generate representation for each node according to its corresponding tweet content. A common practice is to represent each word as a vector by using word2vec algorithm, and then take the mean vector as the representation of the whole tweet (Mikolov, Chen, Corrado, & Dean, 2013). However, social platform generally has a limitation on the number of words in a single post, for example, 280 words on Twitter and 140 words on Weibo, and the posts on social media often consist of several sentences, or even a few words, the word2vec algorithm generally does not work well in such application scenarios (Le & Mikolov, 2014).

In this paper, we use the extension method of word2vec algorithm-Doc2Vec to generate the representations of tweets content (Le & Mikolov, 2014). Doc2vec is also known as paragraph2vec or sentence embedding, which can generate powerful representation for a short text or even a sentence.

#### 4.2. Node update

With the mapping function \( \Phi \) learned using Doc2vec algorithm, we can obtain an original representation \( h_v^{(i)} \) for each node in the propagation graph. However, such node representation \( h_v^{(i)} \) only considers the textual content, ignoring the information of neighborhood nodes and the topological structure in the propagation graph.

In this section, we discuss our proposed PGNN algorithm in a step-by-step manner. Our proposed algorithm is based on the framework of the Graph Neural Network (GNN). GNN can learn useful node representations by encoding local graph structures and node attributes (Kipf & Welling, 2017). Our proposed algorithm updates the node representation iteratively. In a single iteration step, neighbor nodes first exchange information via different types of relation paths, and then update their representations by aggregating the neighborhood information and their own information. As a result, the new obtained node representation contains the textual and contextual information simultaneously.
For each node $v$ in the propagation graph $G$, its representation $h_v$ is iteratively updated until convergence according to Eq. (1), here we omit the subscript and superscript of symbol $v$ and $G$ for simplicity since the update approach is common to different nodes in different propagation graphs.

$$h_v^{(t)} = \sum_{w \in IN(v)} \sum_{i=1}^{R} l(\Psi(e_{w,v}) = r_i)z_iW_e h_w^{(t-1)} + b_i \sum_{i=1}^{R} z_i = 1 \quad (1)$$

The symbol $h_v^{(t)}$ is the representation of node $v$ in the t-th update step. $IN(v)$ is the set of neighbors of node $v$, and $\forall u \in IN(v)$, there exists an edge with the direction from node $u$ to node $v$ in the propagation graph. $l(\cdot)$ is the indicator function. $e_{w,v}$ denotes the directed edge from node $u$ to node $v$. $\Psi : E \rightarrow \mathbb{R}$ is the edge type mapping function. $E$ is the set of edges in the propagation graph and $R$ is the set of relation paths. $[R]$ indicates the total number of types of relation paths. $r_i$ indicates the relation path with the i-th type and the weight of relation path $r_i$ is $z_i$. The sum of the weights of all types of relation paths is 1.

However, this way of iteratively updating node representations converges exponentially to the final result (Zhou et al., 2018). That is to say, node representations need to be updated many times following the recurrence in Eq. (1), which takes a lot of computation time. Inspired by (Li, Tarlow, Brockschmidt, & Zemel, 2016), we include the gated mechanism into the update process mentioned in Eq. (1). The benefit is that the constraint on parameters to ensure convergence can be eliminated, which solves the problem of slow convergence and reduces the time complexity.

The gated recurrent neural network has two common variants: 1) long short-term memory (LSTM) (Hochreiter & Schmidhuber, 1997), 2) gated recurrent units (GRU) (Cho, Van Merrienboer, Bahdanau, & Bengio, 2014). In this work, we use GRU as the hidden unit rather than LSTM for efficiency since GRU uses a simpler architecture and can achieve almost the same performance with fewer parameters.

The changed update process from step t-1 to step t is shown as follows:

$$\delta_v^{(t)} = \sum_{u \in IN(v)} \sum_{i=1}^{R} l(\Psi(e_{w,v}) = r_i)z_iW_e h_u^{(t-1)} + b_i \quad (2)$$

$$z_u^{(t)} = \sigma(W_e \delta_v^{(t)} + U h_v^{(t-1)}) \quad (3)$$

$$r_v^{(t)} = \sigma(W_e \delta_v^{(t)} + U h_v^{(t-1)}) \quad (4)$$

$$h_v^{(t)} = \tanh(W_e \delta_v^{(t)} + U (r_v^{(t)} \circ h_v^{(t-1)})) \quad (5)$$

$$h_v^{(t)} = (1 - z_u^{(t)}) \circ h_v^{(t-1)} + z_u^{(t)} \circ h_v^{(t)} \quad (6)$$

where $h_v^{(t-1)}$ and $h_v^{(t)}$ refers to the previous state at step $t - 1$ and the current state at step $t$ of node $v$ respectively. $\delta_v^{(t)}$ denotes the merged information from the neighbors of node $v$ via incoming edges with parameters dependent on the relation path, it is used as the input of GRU unit at step t. $h_v^{(t)}$ is the candidate state of the hidden state $h_v^{(t)}$. $W_e$, $U$, $W$, $W$, $U$ are the weights connections inside GRU unit. $\circ$ is the element-wise multiplication. The reset gate $r_v^{(t)}$ determines how much previous hidden state $h_v^{(t-1)}$ is included in the current candidate state $h_v^{(t)}$. The update gate $z_u^{(t)}$ determines how to obtain current hidden state $h_v^{(t)}$ based on candidate state $h_v^{(t)}$ and previous state $h_v^{(t-1)}$. GRU uses two different types of activation functions, namely $\sigma(x)$ and $\tanh(x)$, $\sigma(x)$ is the sigmoid function, its mathematical expression is $1/(1 + e^{-x})$. $\sigma(x)$ is used to limit the values of reset gate $r_v^{(t)}$ and update gate $z_u^{(t)}$ between 0 and 1. The activation function $\tanh(x) = (e^x - e^{-x})/(e^x + e^{-x})$ is used to limit the value of candidate state $h_v^{(t)}$ between 0 and 1. To obtain the current hidden state of node $v$, the GRU unit selectively adds the information aggregated from the neighbors of node $v$ and selectively forgets the previous hidden state of node $v$.

There are two benefits to include the gated mechanism: 1) the cumulative speed of information can be controlled, 2) the noise due to excessive iterations can be reduced.

Fig. 2 shows the update process of node $v$ (the black node) according to Eqs. (2)-(6), the direction and type of the relation path are not discussed here for the sake of simplicity. In the 0-th step, all nodes only contain the context information of their own. From the 0-th step to the first step, the node $v$ aggregates the information from its first-order neighbors and uses the GRU unit to update the hidden state, then the new obtained hidden state $h_v^{(1)}$ contains the information of the 1-hop neighborhood of node $v$, the same is true for other nodes. From the first step to the second step, node $v$ repeats the update process, since each node in $IN(v)$ (grey nodes) also contains information of its first-order neighbors, the node $v$ extends its receptive field from 1-hop to 2-hop. Through inductive analysis, we can conclude that the hidden state $h_v^{(t)}$ contains the information of t-order neighbors. The dash-dotted circle in Fig. 2 shows the receptive field of node $v$ at step $t$. The parameter $\alpha$ in Eq. (2) is used to balance the weights of different types of relation paths. However, we argue that the weights of relation paths with the same type should also be different. For example, after a tweet is posted, users will reply to it to express the orientation towards the veracity. Obviously, the comments with different stances should not be treated equally. We believe that the weight of the relation path is related to the hidden states of nodes at both ends of the path, so we include the attention mechanism to dynamically adjust the weights of relation paths.

The attention mechanism was originally proposed by (Bahdanau, Cho, & Bengio, 2014) to solve the problem of machine translation. Nowadays, it has been widely used in various deep learning models. Attention mechanism can force models to focus selectively on specific information while ignoring irrelevant information. Take machine translation task as an example, when translating a word, only certain words around the target word may be relevant, there is no need to pay attention to all the words in the paragraph.

We include attention mechanism to adjust the weights when aggregating the information from neighbors following Eq. (2), the modified Equation is shown as Eq. (7).

$$\delta_v^{(t)} = \sum_{u \in IN(v)} \sum_{i=1}^{R} l(\Psi(e_{w,v}) = r_i)z_iW_e h_u^{(t-1)} + b_i \quad (7)$$

where $\alpha_{u,t-1}$ is the attention score, it can be calculated as Eq. (8)

$$\alpha_{u,t-1} = \frac{\exp(h_u^{(t-1)} \cdot h_u^{(t-1)})}{\sum_{u \in IN(v)} \exp(h_u^{(t-1)} \cdot h_u^{(t-1)})}, u \in IN(v) \quad (8)$$

where $\exp(x) = e^x$, the symbol $\cdot$ denotes the dot product.

After updating T times according to the above process, the final representation of each node in the propagation graph can be obtained. But the representations have limited expression capability since they are learned by using graph neural network with single layer. (LeCun, Bottou, Bengio, & Haffner, 1998) tried to use a multi-layer convolutional neural network to extract more abstract information from images. Inspired by this idea, our proposed algorithm also uses a multi-layer architecture. The update procedures...
in each layer are the same and are in accordance with Eqs. (2)–(6), but the parameters in different layers may be totally different, which forces each layer to focus on different information when updating node representations. The relation of node representations in adjacent layers is as follows:

\[ \forall \nu \in V, h^{(0)}_{\nu}(m + 1) = h^{(m)}_{\nu}(m) \tag{9} \]

where \( T_m \) denotes the total update times at layer \( m \), \( h^{(m)}_{\nu}(m) \) denotes the hidden state of node \( \nu \) after being updated \( t \) times at layer \( m \). Eq. (9) demonstrates that the node representations obtained in layer \( m \) are used as the initial states in layer \( m + 1 \).

The above mentioned process is our proposed Propagation Graph Neural Network (PGNN) algorithm. The complete procedures of PGNN are shown in Table: Algorithm PGNN.

Algorithm: PGNN

| Input: |
| --- |
| \( V \): the set of nodes in the propagation graph; |
| \( E \): the set of textual embeddings; |
| \( M \): the total number of layers in propagation graph neural network |
| \( T = \{T_0, T_1, \ldots, T_M\} \): \( M \) is the number of layers, \( T_m \) represents the number of update times in layer \( m \), \( T_0 = 0 \); |
| Output: |
| \( H \): the set of all node representations after \( M \)-layer updates. |

\[ H = \{h^{(m)}_{\nu}(M) | \nu \in V \} \]

Begin

For \( \nu \in V \) do

\[ h^{(0)}_{\nu}(0) = E_{\nu} \]

End for

For \( m = 1, 2, \ldots, M \) do

For \( t = 0, 1, \ldots, T_m \) do

If \( t = 0 \) then

For \( \nu \in V \) do

End for

Use the representation of node \( \nu \) obtained in the previous layer as the initial state of node \( \nu \) in the current layer, the initial state

\[ h^{(0)}_{\nu}(0) \]

is calculated according to Eq. (9)

End for

Else

For \( \nu \in V \) do

End for

Use attention mechanism to adjust the weights of different incoming edges of node \( \nu \) with the same type of relation path, the attention score is calculated according to Eq. (8)

End for

Aggregate the information from the neighbors of node \( \nu \) according to Eq. (2). Use gated mechanism to update the hidden state of node \( \nu \) according to Eqs. (3)–(6)

End for

End if

End for

End for

End

4.3. Classification

In this section we discuss how to identify different types of rumors based on the representations calculated by PGNN algorithm. We propose two ideas for rumor classification:

1) Detecting rumors based on the propagation graph can be regarded as a graph classification task. The common practice to solve the problem is to represent the whole graph with a single embedding (Bacciu, Errica, & Micheli, 2019). At that point, a standard classifier can be applied to output a graph prediction. Based on this, we come up with the first idea to classify rumors. We first aggregate the representation of each single node from a global perspective to obtain a global graph embedding for the whole propagation graph, and then we use the global embedding as the input of a fully connected neural network to get the prediction result;

2) Our second idea comes from the ensemble learning (Friedman (2001)). The main idea of ensemble learning is to predict separately with multiple individual learners and then combine the results with some certain strategies. As we discussed in Section 4.2, the representation of each individual node contains the information of its neighbors, we can first calculate the prediction probability for each node separately, and then summarize to obtain the final classification result.

For the first idea, the simplest approach is to take the mean of all node representations as the global embedding for the entire propagation graph, which is as shown in Eq. (10).

\[ g = \tanh(\frac{1}{N} \sum_{\nu \in V} h_{\nu}(T)) W_g + b_g \]

where \( g \) denotes the global representation for the entire propagation graph, \( W_g \) and \( b_g \) are weights and bias respectively, and \( T \) is the symbol denoting transpose.

As shown in Eq. (11), the obtained global representation \( g \) is feed into a fully-connected layer with softmax activation function to generate the prediction for the source tweet

\[ \hat{y} = \text{softmax}(F(g)) \]

where \( F \) is a fully-connected layer, the dimension of its input and output are \( d \) and \( k \) respectively, \( d \) is determined by the dimension of the global graph embedding \( g \), and \( k \) is determined by the categories of rumors. In this paper, we assume that the source tweets can be divided into four classes: 1) non-rumors, 2) true rumors, 3) false rumors, 4) unconfirmed rumors, therefore the value of \( k \) is 4. If the source tweets are simply classified as rumors and non-rumors, the value of \( k \) is 2.\( \hat{y} \) denotes the prediction result.

For the second idea, a simple way is to calculate the individual prediction probability according to each node representation first, and then use the linear summation to obtain the final result. The second idea can be formalized as Eq. (12).

\[ \hat{y} = \text{softmax}(\sum_{\nu \in V} \sigma(h_{\nu}^T W_s + b_s)) \]

where \( \sigma(\cdot) \) is the sigmoid function, it is used to limit the output between 0 and 1, \( W_s \) and \( b_s \) are weight and bias respectively, \( W_s \in \mathbb{R}^{d \times k} \), \( b_s \in \mathbb{R}^k \).

There is room for improvement in above two methods. From the intuitive point of view, the nodes in the propagation graph should not be treated equally. For the first idea, Eq. (10) is on the premise that each node representation makes equal contribution to the graph embedding. The global graph embedding is obtained by aggregating all nodes linearly. However, sophisticated aggregation
strategy may improve the classification performance (Bacciu et al., 2019). In our work, we aggregate the node representations for graph embedding with attention mechanism. For the second idea, Eq. (12) simply averages the prediction results, which can be seen as the majority voting strategy. However, the weighted voting strategy in ensemble learning assigns each classifier a weight so that the prediction results can be treated differently. In our work, we use attention mechanism to adjust the contribution that each node makes to the classification result dynamically, so that the prediction result based on each individual node representation can be treated differently.

For the first idea, the weight of each node is calculated based on the difference between the single node representation and the mean of all node representations. We replace Eq. (9) with Eqs. (12) and (13).

$$c = \tanh \left( \frac{1}{N} \sum_{v \in V} H_v \right)^T W_c + b_c \tag{13}$$

$$g = \sum_{v \in V} \sigma( [H_v^T, c^T] W_{g1}) \odot (H_v^T W_{g2}) \tag{14}$$

where \( \tanh(x) = \sinh(x)/cosh(x) = (e^x - e^{-x})/(e^x + e^{-x}) \), it is an activation function, \( \odot \) is the element-wise multiplication, and \( W_{g1} \in \mathbb{R}^{d \times d}, W_{g2} \in \mathbb{R}^{d \times d} \). The global representation \( g \) is calculated based on all node representations with different weights.

For the second idea, we first calculate the individual prediction probability according to each node representation, and then compare the difference between the single node representation and the mean in order to determine the contribution each node makes to the final classification result. We modify Eqs. (11)–(14).

$$y = \text{softmax}( \sum_{v \in V} \sigma( [H_v^T, c^T] W_{e1}) \odot (H_v^T W_{e2}) ) \tag{15}$$

where \( W_{e1} \in \mathbb{R}^{d \times k}, W_{e2} \in \mathbb{R}^{d \times k} \).

In this paper, we propose two models for rumor detection. We name the first model GLO-PGNN. PGNN stands for the propagation graph neural network and GLO stands for global graph embedding. GLO-PGNN first obtains the node representations using the propagation graph neural network, and then identifies rumors based on the global graph embedding. We name the second model ENS-PGNN. PGNN also stands for the propagation graph neural network and ENS stands for ensemble learning. ENS-PGNN uses the propagation graph neural network to learn the node representations, and then classifies rumors with the weighted voting strategy which is commonly used in ensemble learning. Both of our proposed models can be trained in an end-to-end fashion. Experiments show that by including attention mechanism, the two models pay more attention on the nodes whose representations are very different from the mean value, which can significantly improve the classification performance.

4.4. Complexity analysis

In this section, we discuss the time complexity and space complexity of the two proposed models. For deep learning algorithm, it is more important to focus on the prediction time than the training time. Therefore, we only estimate how long it takes the proposed models to detect one rumor. The GLO-PGNN and ENS-PGNN algorithms differ only in the approach used for classification.

**Time complexity:** The time complexity can be measured by FLOPs. FLOPs is the abbreviation of floating-point operations per second. For the PGNN algorithm, when the information from neighbors is aggregated according to Eq. (2), the time complexity is related to the number of nodes in the propagation graph \( |V| \) and the average indegree \( \beta \). The time complexity of Eq. (2) is about \( O(\beta d^2) \), and the time complexity is about \( O(6d^3) \) when the hidden state of each node is updated using Eqs. (3)–(6). So the time complexity of updating all nodes is \( O(\beta + 6) \times |V| \times d^3 \). For the GLO-PGNN algorithm, the time complexity of the classification stage is approximately \( O(3\times |V| \times d^3) \). For the ENS-PGNN algorithm, the time complexity of the classification stage is approximately \( O(3k \times |V| \times d) \). Experiments show that the average number of nodes in each propagation graph is 21,536, and the average indegree of the nodes is 1,902.

**Space complexity:** We only care about the number of trainable parameters. In the PGNN algorithm, it can be known from Eq. (2) that the relation path corresponds to parameters \( W_i \in \mathbb{R}^{d \times d} \) and \( b_i \in \mathbb{R}^{d \times 1} \). There are two types of relation paths, the number of trainable parameters in Eq. (2) is \( 2d^2 + 2d \). As shown in Eqs. (3)–(6), the candidate state \( h \) has the same dimension as the input \( \delta \), and \( W_z, U_z, W_r, U_r, W, U \) are the weights inside GRU unit, they all have \( d^2 \) parameters. Therefore, the number of trainable parameters of PGNN algorithm is \( 8d^2 + 2d \). For the GLO-PGNN algorithm, the number of trainable parameters of the classification stage is \( 4d^2 + d \). For the ENS-PGNN algorithm, the number is \( 3dk \). Therefore, the total number of trainable parameters of the GLO-PGNN and ENS-PGNN algorithms are \( 12d^2 + 3d \) and \( 8d^2 + 3dk \) respectively.

4.5. Model training

The loss function of our proposed algorithm consists of two modules: 1) the cross entropy loss between the ground truth and
the probability distributions of the predictions; 2) the regularization term of the parameter. The loss function is calculated as shown in Eq. (16).

\[ L = - \sum_{n=1}^{N} \sum_{c=1}^{C} y_n(x_n) \log(\hat{y}_c(x_n)) + \lambda||\theta||^2 \]  

(16)

where \( N \) is the total number of training samples, \( C = \{N, T, F, U\} \) denotes the set of four different types of rumors. \( y_n(x_n) \) denotes the ground truth of the \( n \)-th example for the \( c \)-th class, if the example \( x_n \) belongs to \( c \)-th class, then \( y_n(x_n) = 1 \), otherwise 0. \( \hat{y}_c(x_n) \) denotes the prediction probability that the \( n \)-th example belongs to the \( c \)-th class. \( || \cdot ||^2 \) denotes the regularization term over all trainable parameters, \( \lambda \) is the trade-off coefficient.

In order to optimize the model, we need to iterate all the training examples in each epoch and continue until the loss value calculated following Eq. (16) converges or the specified maximum epoch number is reached. Throughout the training process, all trainable parameters are updated using back-propagation algorithm (Goller & Kuchler, 1996).

There are many optimizers for training neural network based models, such as SGD, AdaGrad, RMSProp and Adam (Yu, 2020; Kingma & Ba, 2014; Shao, Pi, & Shao, 2019). SGD and some of its variants, such as batch gradient descent and mini-batch gradient descent, are very popular perform optimization in neural network. However, they all suffer from some drawbacks: 1) it is hard to choose a proper learning rate at the beginning, 2) all parameters are updated with the same learning rate, sometimes we may not want to update all the parameters to the same extend, especially when the data is sparse and the features have very different frequencies, 3) all these algorithms all face the problem of getting trapped in the numerous suboptimal local minima and saddle points. To solve these the first problem and the second problem, some optimization algorithms that can adaptively adjust the learning rate have been proposed. Take the AdaGrad algorithm as an example, AdaGrad can adapt the learning rate to different parameters, it updates the frequently occurring features with small learning rate and updates the infrequently occurring features with large learning rate. To solve these the second problem, the momentum based optimization algorithms have been proposed, such as RMSProp. RMSProp adjusts the learning rate according to the exponential moving average of squared gradient, which can accelerate the convergence speed and reduce the oscillation at saddle point or local minimum point. To combine the advantages of AdaGrad and RMSProp, Adam algorithm has been proposed. It effectively solves the above drawbacks of SGD, considering the first moment estimation and the second moment estimation of the gradients synthetically. Adam can adjust the learning rate adaptively and to some extent avoid the suboptimal local minima, which is very suitable for our models. In our work, we use the Adam optimizer to speed up the training process.

5. Experiment and result

In order to verify the performance of the proposed models, we performed comparative analysis over publicly available twitter dataset. We evaluated the detection performance, prediction time and early stop performance of our models and compared with other models to illustrate the superiority of GLO-PGNN and ENS-PGNN. We also discussed how to select particular values for different parameters and how to select the optimal models.

5.1. Dataset

For experimental evaluation, we use the recently released Twitter dataset - PHEME (Kochkina, Liakata, & Zubiaga, 2018), which consists of tweet data and user data from the Twitter platform. The dataset contains 15 6425 post sets related to 9 events, each post set consists of a source tweet and several responsive tweets. Each post set is annotated as either rumor or non-rumor, and rumors in the dataset are further labeled as true, false and unverified by professional journalists. As shown in Table 1, the total number of tweets in the PHEME dataset is 105354. The number of non-rumors, false rumors, true rumors and unverified rumors are 4023, 638, 1067 and 697 respectively. The average number of tweets in each post set is 16.

In the original PHEME dataset, the number of non-rumors is much more than the rumors and some post sets contain very few tweets. Therefore, we filter out post sets containing less than 4 tweets, and then randomly sample 800, 400, 600, 500 post sets respectively from four finer-grained classes: non-rumor, false rumor, true rumor and unverified rumor. We use the sampled post sets to form a new dataset. During training and testing, we construct each post set in the new dataset into a propagation graph according to the way proposed in Section 3.1.

5.2. Experimental setup

In this section, we first detail the evaluation metrics and the algorithms selected for comparison, and then we discuss how to select particular values for different parameters and how to select the optimal models.

5.2.1. Evaluation metrics

Owing to the imbalanced class prevalence, it is unreasonable to evaluate methods based solely on accuracy, because those methods that tend to classify the unknown example into the majority class will be considered to have better performance. In this work, we use accuracy, Micro-F1 value and Macro-F1 value to evaluate the performance. F1 score can be calculated following Eqs. (17)–(19).

\[ \text{Precision} = \frac{TP}{TP + FP} \]  

(17)

\[ \text{Recall} = \frac{TP}{TP + FN} \]  

(18)

\[ F_1 = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \]  

(19)

5.2.2. Baseline methods

We have made a detailed comparison of our proposed methods and some of the state-of-the-art baselines on rumor classification task. The methods involved are as follows:

- **RPC**: (Kwon, Cha, Jung, Chen, & Wang, 2013) used the random forest classifier to classify rumors, with inputs being several temporal properties and handcrafted features based user information, tweet content and propagation characteristics.
- **SVM-BOW**: (Mihalcea & Strapparava, 2009) used Bag-of-Words model to represent text based on the frequency of words appearing in the content.
in the text. The obtained representations were then used as the inputs of a linear SVM model for lie detection task.

**SVM-TS:** (Ma et al., 2015) divided the time series into fixed time intervals, and captured the temporal characteristics by comparing the difference of social context information between adjacent time intervals. The temporal characteristics, together with several hand-craft features based on tweets content, user information and propagation structure, were used as inputs of a SVM model for rumor detection task.

**GRU-RNN:** The method proposed by (Ma et al. 2016) used the recurrent neural network with GRU unit to obtain the representation of the post set by modeling the sequential dynamics of responsive tweets. This method is also a simplified form of the MT-ES method proposed by (Ma et al., 2018a) (with the user stance classification module removed).

**BU-RvNN and TD-RvNN:** Two variants of the RvNN model proposed by (Ma, Gao, & Wong, 2018b). Both methods construct the propagation tree following the non-sequential propagation structure of tweets in each post set. BU-RvNN captures the structural properties by visiting all nodes recursively in a bottom-up manner, and then classify rumors based on the representation of the single root. TD-RvNN visits nodes in a top-down manner, and identifies rumors based on the representations of leaf nodes via max pooling technique.

**GLO-PGNN (basic) and ENS-PGNN (basic):** These two models are the simplification of GLO-PGNN and ENS-PGNN. They both use the PGNN algorithm to obtain the representations of all nodes in the propagation graph. GLO-PGNN (basic) calculates the global graph embedding by simply averaging the node representations and then uses the fully connected network for classification, which is shown in Eq. (10) and Eq. (11). ENS-PGNN calculates the prediction probability for each node based on their representations and then combines the prediction results with the majority vote strategy, which is shown in Eq. (12).

**GLO-PGNN and ENS-PGNN:** GLO-PGNN is an improvement on GLO-PGNN(basic). The difference is that GLO-PGNN aggregates the node representations with attention mechanism for the global graph embedding, which is shown in Eq. (13) and Eq. (14). ENS-PGNN is an improvement on ENS-PGNN(basic). The difference is that ENS-PGNN combines the prediction results with the weighted voting strategy, which is shown in Eq. (15).

### 5.2.3. Parameters settings

For the algorithms selected for comparison, we use the default values of the parameters given in the cited papers. If the values of the parameters are not given explicitly, we tune the parameters following the suggestions in the cited papers to find the optimal value. For the SVM-BOW method, when generating representations for text content, only the top 1000 words with the highest frequency in the vocabulary are considered. Therefore, the dimension of the word frequency vector is 1000. We use the RBF kernel as the kernel function of the SVM-BOW method. The penalty coefficient $C$ is set to 1.0 and the gamma value of the RBF kernel function is set to 0.1. For the RF method, we use 500 decision trees to construct the Random Forest classifier. The max depth of one single tree is set to 3. The minimum number of samples divided into each leaf node is set to 1. The minimum number of samples required for splitting is set to 2. For the SVM-TS method, we divide the entire time series into 10 time intervals, since the average number of tweets in each post set is 16.

For the sake of fairness, all neural network based methods use doc2vec algorithm to learn representations of tweets content. The doc2vec algorithm has two variants: 1) Distributed Memory Model of Paragraph Vectors (PV-DM), 2) Distributed Bag of Words version of Paragraph Vector (PV-DBOW); we use both variants to learn 20-dimensional representations respectively, and then concatenate to get the 40-dimensional representation for tweet content. It should be noted that there are some useless characters (i.e. hashtags"@" and mentions "#") and hyperlinks in tweets. We filter out these useless information in the data cleaning phase. The GRU-RNN, BU-RvNN, TD-RvNN and our proposed methods include the GRU network, we set the layer of the GRU network to 1 and the size of GRU unit to 40. The balance parameter $\lambda$ and the learning rate $\alpha$ of all neural network based methods are set to 0.01 and 0.05 respectively. The optimization techniques commonly used in deep learning methods, such as dropout (Srivastava, Hinton, Krizhevsky, Sutskever, & Salakhutdinov, 2014) and normalization, are also used to implement the algorithms.

When using PGNN to learn representations of nodes in the propagation graph, the number of layers $M$ and the number of updates per layer $T$ need to be set manually. We first fix the number of layers and set $M$ as 1. Fig. 3(a) and (b) demonstrate the F1 scores of the two methods proposed by us respectively as $T$ increases. The optimal value of $T$ is 2, since the micro-averaged and macro-averaged F1 scores of both methods reach the maximum when $T = 2$. Then we fix the number of updates per layer and set $T$ as 2, Fig. 4(a) and (b) show the F1 scores of our proposed methods respectively as $M$ increases. For the GLO-PGNN method, the optimal value of $M$ is 2, and for the ENS-PGNN method, the optimal value of $M$ is 3. We assume that the number of updates per layer is the same for the sake of simplicity. However, the number of updates can be different in different layers. We implement the doc2vec algorithm using genism.

We implement RFC, SVM-BOW and SVM-TS models using scikit-learn. We implement all neural network based models using Tensorflow.

### 5.2.4. Optimal model

For RFC, SVM-BOW and SVM-TS models, we hold out 90% and 10% of the examples in the dataset for training and testing respectively. We perform 10-fold cross-validation throughout all experiments. We randomly splits the training set into 10 distinct subsets called folds, then train and evaluate models 10 times, picking a different fold for evaluation every time and training on the other 9 folds. Each time we first train the model against the training set and select the optimal model against the validation set, then we test the optimal model against the testing set. We take the average of the F1 scores in 10 tests as the final result to measure the performance of each model.

For neural network based methods, we hold out 80%, 10% and 10% of the examples in the dataset for training, validation and testing respectively. We select the optimal model as follows: At the beginning of each epoch, the training set is randomly divided into a number of batches, which are used as inputs to the model successively. The parameters of the model are updated by using back-propagation to minimize the loss. At the end of each epoch, we evaluate the model against validation set. We continue the above epoch until the best performance is not improved in 10 consecutive epochs. When the training phase is over, we measure the performance of the optimal model against the testing set.

Fig. 5(a) and (b) show the accuracy and loss of our propose methods against training set and validation set as the epoch increases. The x-axis is the number of epoch, the y-axis on the left represents the loss value while the y-axis on the right represents the accuracy. The x-axis value corresponding to the dotted line indicates the number of training epochs when the model is optimal.

### 5.3. Rumor classification performance

We classify all the compared methods into three groups. The first group consists of traditional methods, including SVM-BOW,
RFC and SVM-TS. The second group contains some state-of-the-art baselines based on the neural network, including GRU-RNN, BU-RvNN and TD-RvNN. Our proposed methods are in the third group. As shown in Table 2, our proposed methods achieve much better performance than other methods.

It is observed that the 3 methods in the first group achieve poor performance, fluctuating around 0.5 in accuracy, which is far lower than the methods in the second and third groups. This upsetting result demonstrates that the traditional methods based on hand-crafted features fail to identify rumors effectively. RFC performs relatively better than SVM-BOW and SVM-TS, since RFC comprehensively considers the temporal dynamics of user information, linguistic property and propagation structure. SVM-TS considers the difference of social context information between adjacent intervals additionally, however, the average number of tweets per post set in the PHEME dataset is only 16, which hinders SVM-TS model from capturing the temporal dynamics of features. SVM-BOW performs slightly worse than RFC since it classifies rumors based solely on the linguistic information captured from tweets content.

In the second group, it is obvious that the neural network based methods achieve superior improvements of more than 25%, compared with the traditional methods in the first group. This significant improvement indicates that neural network based methods do have better generalization capabilities to learn discriminative features automatically. Among all 3 methods in the second group, GRU-RNN is inferior to the two recursive models. This is because GRU-NN only considers the temporal dynamics of tweets content while TD_RvNN and BU_RvNN additionally take into account the non-sequential propagation structure of tweets. As shown in Table 2, TD_RvNN is superior to BU_RvNN, which is consistent with the experimental results in the original paper (Ma. et al. 2018). From an intuitive perspective, BU_RvNN suffers from much larger information loss because BU_RvNN classify rumors based on the
representation of single root while TD_RvNN comprehensively considers the representations of all leaf nodes. Although TD_RvNN and BU_RvNN can achieve decent performance, they are still inferior to our proposed methods. In order to overcome the shortcoming that TD_RvNN and BU_RvNN can only disseminate information in one direction using either a top-down or bottom-up manner, our proposed methods extend the propagation tree to the propagation graph by adding relation paths and include graph neural network to learn more powerful representations.

In the third group, GLO-PGNN (basic) and ENS-PGNN (basic) both achieve better performance than TD_RvNN and BU_RvNN, suggesting that the propagation graph neural network effectively captures the temporal characteristic and the topological characteristic of rumors. The two methods only differ in the strategy used for classification. GLO-PGNN (basic) first calculates the global graph embedding of the whole propagation graph by averaging all the node representations, and then uses the fully-connected network for classification. ENS-PGNN (basic) first calculates the prediction probability for each node based on their representations, and then combines the prediction results with the majority vote strategy. GLO-PGNN and ENS-PGNN yield the first highest and second highest F1 scores, suggesting that our proposed methods are more effective on rumor detection task. GLO-PGNN is an improvement on GLO-PGNN (basic). The difference is that GLO-PGNN takes a weighted average of the node representations to obtain the global graph embedding. Compared with GLO-PGNN (basic), GLO-PGNN achieve 1.6% improvements in terms of Micro-F1 and 1.4% gains in term if Macro-F1. ENS-PGNN is an improvement on ENS-PGNN (basic). The difference is that ENS-PGNN combines the prediction results with a weighted voting strategy. Compared with ENS-PGNN (basic), ENS-PGNN achieve 1.7% improvements in terms of Micro-F1 and 1.8% gains in term if Macro-F1. Both GLO-PGNN and ENS-PGNN assign different to nodes in the propogation graph. The value of the weight is determined by the attention mechanism, as shown in Eqs. (13)–(15). The improvement in terms of both Macro-F1 and Micro-F1 scores justifies the rationality of including attention mechanism. We can conclude that using attention mechanism to dynamically adjusting the node weights contributes to improving the detection performance. Besides, it is observed that GLO-PGNN is superior to ENS-PGNN in terms of most evaluation metrics. The Micro-F1 score achieved by GLO-PGNN is about 1.1% over ENS-PGNN. The Macro-F1 score achieved by GLO-PGNN is about 1.5% over ENS-PGNN. However, ENS-PGNN has fewer trainable parameters, which makes ENS-PGNN require smaller storage spaces especially when the representation has a large dimension.

According to the analysis, we conclude that taking into consideration the temporal and topological characteristic of rumors contributes to improving the detection performance significantly. The effectiveness of attention mechanism indicates that rumors have different influences at different stages. In this work, we adjust the weight of each node with attention mechanism to quantify the influence, which improves the detection performance slightly. In addition, if some researches set a large value for the node representation or have limited computing resources, we recommend them to adopt ENS-PGNN. Compared to GLO-PGNN, ENS-PGNN has slightly inferior performance, but requires smaller storage spaces.

### 5.4. Prediction time

To demonstrate the efficiency of our proposed models, we compared the prediction time of all deep learning based models. We
randomly selected 400 examples from the testing data and recorded the running time taken by each model to identify these examples. As shown in Fig. 6, the X-axis represents name of compared models and the Y-axis represents the value of prediction time.

Compared with GRU-RNN, BU-RvNN and TD-RvNN take more time to identify all the examples, which can be inferred from their structures. The GRU units in GRU-RNN are organized serially while the GRU units in BU-RvNN and TD-RvNN are organized in the form of tree structure. BU-RvNN and TD-RvNN have more complex structures and therefore take more time to predict. It is observed that our proposed models have the lowest time complexity. In GRU-RNN, the input of GRU unit depends on its predecessor. In BU-RvNN and TD-RvNN, the input of GRU unit depends on its parent node or children node. The GRU cells are highly dependent on each other, so they cannot be parallelized. However, our models can update the node representations in parallel because the representation of each node is only related to its k-hop neighborhood. In each layer of the PGNN algorithm, all nodes can use GRU unit to update their representations simultaneously.

In summary, the prediction time of our proposed models is very fast due to parallelization, indicating that GLO-PGNN and ENS-PGNN are very efficient in the speed with which they identify rumors.

5.5. Early stop performance

Detecting rumors at an early stage of propagation is very important so that timely mitigation could be made to prevent the adverse impact caused by rumors from further expanding. We evaluate the early stop performance of different methods in terms of different time delays measured by either tweet count received or time elapsed since the source tweet is posted. For each method, we first obtain the optimal model following the training approaches mentioned in Section 5.2.4. Then we evaluate the performance of each optimal model by accuracy as we incrementally increase the tweet count or elapsed time.

Fig. 7 shows the statistics of all post sets in the testing set about tweets count and elapsed time. The X-axis in Fig. 7(a) represents
the tweet count and the Y-axis represents the percentage. The X-axis in Fig. 7(b) represents the elapsed time and the Y-axis represents the percentage. As shown in Fig. 7(a), more than 90% of the post sets contain less than 60 tweets. As shown in Fig. 7(b), around 90% of the post sets have a time span not exceeding 25 h. Therefore, we increase the tweets count from 2 to 50 and increase the

![Graphs showing early stop performance at different checkpoints in terms of elapsed time and tweets count.]

Fig. 8. Early stop performance at different checkpoints in terms of elapsed time and tweets count.
elapsed time from 0.02 h to 20 h when evaluating the early stop performance.

Fig. 8(a) and (b) show the early stop performance of several neural network based methods at different checkpoints in terms of elapsed time and tweets count. As shown in Fig. 8(b), the accuracy curves are hard to distinguish at a very early stage. To have a better observation, Fig. 8(c) takes 1 h as the cutoff point and shows the early stop performance of each model in two different time periods. We only consider the neural network based methods since they perform much better than traditional methods on rumor detection task. It is observed that our proposed models demonstrate superior early detection performance than other models.

As shown in Fig. 8(a), the accuracy of each model climbs rapidly as the tweets count incrementally increases from 1 to 10. When the number of tweets exceeds 15, the growth rate of accuracy gradually decreases. The accuracy of models considering the propagation structure tends to be stable when the tweets count exceeds 30, while the accuracy of GRU-RNN tends to be stable much earlier. As shown in Fig. 8(c), the performance of each model improves rapidly within the 0.3 h after the source tweet is posted and tends to be stable when the elapsed time exceeds 1 h. TD-RvNN, GLO-PGNN and ENS-PGNN only need around 10 tweets or about 0.4 h to achieve the best performance of GRU-RNN because they take into consideration the non-sequential propagation structure of tweets. TD-RvNN, GLO-PGNN and ENS-PGNN only need around 10 tweets or about 0.4 h to achieve the best performance of GRU-RNN because they take into consideration the non-sequential propagation structure of tweets. TD-RvNN, GLO-PGNN and ENS-PGNN only need around 10 tweets or about 0.4 h to achieve the best performance of GRU-RNN because they take into consideration the non-sequential propagation structure of tweets. TD-RvNN, GLO-PGNN and ENS-PGNN only need around 10 tweets or about 0.4 h to achieve the best performance of GRU-RNN because they take into consideration the non-sequential propagation structure of tweets. TD-RvNN, GLO-PGNN and ENS-PGNN only need around 10 tweets or about 0.4 h to achieve the best performance of GRU-RNN because they take into consideration the non-sequential propagation structure of tweets. TD-RvNN, GLO-PGNN and ENS-PGNN only need around 10 tweets or about 0.4 h to achieve the best performance of GRU-RNN because they take into consideration the non-sequential propagation structure of tweets. TD-RvNN, GLO-PGNN and ENS-PGNN only need around 10 tweets or about 0.4 h to achieve the best performance of GRU-RNN because they take into consideration the non-sequential propagation structure of tweets. TD-RvNN, GLO-PGNN and ENS-PGNN only need around 10 tweets or about 0.4 h to achieve the best performance of GRU-RNN because they take into consideration the non-sequential propagation structure of tweets. TD-RvNN, GLO-PGNN and ENS-PGNN only need around 10 tweets or about 0.4 h to achieve the best performance of GRU-RNN because they take into consideration the non-sequential propagation structure of tweets. TD-RvNN, GLO-PGNN and ENS-PGNN only need around 10 tweets or about 0.4 h to achieve the best performance of GRU-RNN because they take into consideration the non-sequential propagation structure of tweets. TD-RvNN, GLO-PGNN and ENS-PGNN only need around 10 tweets or about 0.4 h to achieve the best performance of GRU-RNN because they take into consideration the non-sequential propagation structure of tweets. TD-RvNN, GLO-PGNN and ENS-PGNN only need around 10 tweets or about 0.4 h to achieve the best performance of GRU-RNN because they take into consideration the non-sequential propagation structure of tweets. TD-RvNN, GLO-PGNN and ENS-PGNN only need around 10 tweets or about 0.4 h to achieve the best performance of GRU-RNN because they take into consideration the non-sequential propagation structure of tweets. TD-RvNN, GLO-PGNN and ENS-PGNN only need around 10 tweets or about 0.4 h to achieve the best performance of GRU-RNN because they take into consideration the non-sequential propagation structure of tweets. TD-RvNN, GLO-PGNN and ENS-PGNN only need around 10 tweets or about 0.4 h to achieve the best performance of GRU-RNN because they take into consideration the non-sequential propagation structure of tweets. TD-RvNN, GLO-PGNN and ENS-PGNN only need around 10 tweets or about 0.4 h to achieve the best performance of GRU-RNN because they take into consideration the non-sequential propagation structure of tweets. TD-RvNN, GLO-PGNN and ENS-PGNN only need around 10 tweets or about 0.4 h to achieve the best performance of GRU-RNN because they take into consideration the non-sequential propagation structure of tweets. TD-RvNN, GLO-PGNN and ENS-PGNN only need around 10 tweets or about 0.4 h to achieve the best performance of GRU-RNN because they take into consideration the non-sequential propagation structure of tweets. TD-RvNN, GLO-PGNN and ENS-PGNN only need around 10 tweets or about 0.4 h to achieve the best performance of GRU-RNN because they take into consideration the non-sequential propagation structure of tweets. TD-RvNN, GLO-PGNN and ENS-PGNN only need around 10 tweets or about 0.4 h to achieve the best performance of GRU-RNN because they take into consideration the non-sequential propagation structure of tweets. TD-RvNN, GLO-PGNN and ENS-PGNN only need around 10 tweets or about 0.4 h to achieve the best performance of GRU-RNN because they take into consideration the non-sequential propagation structure of tweets. TD-RvNN, GLO-PGNN and ENS-PGNN only need around 10 tweets or about 0.4 h to achieve the best performance of GRU-RNN because they take into consideration the non-sequential propagation structure of tweets.

In summary, our proposed models achieve the highest prediction accuracy even when the tweet count is very small or the elapsed time is very short, indicating that GLO-PGNN and ENS-PGNN possess excellent early detection performance.

5.6. Case study

In this section, a case is analyzed in detail to give an intuitive explanation of how our proposed models identify rumors.

Taking as an example the source tweet whose tweet id is 544307028815253504, the tweet is correctly classified into the false rumor by the models proposed by us. Fig. 9(a) shows the partial propagation structure of the source tweet and its responsive tweets. Fig. 9(b) shows the tweet content. Each node in Fig. 9(a) corresponds to the textual content with the same index in Fig. 9(b). Since the user information in the dataset is desensitized due to privacy issues, we use user 0, 1, ... to represent the user id. As shown in Fig. 9, user0 released a source tweet whose tweet id is 544307028815253504, user 1, 2, ..., 16 express different kinds of stances towards the source tweet which include supporting, denying, querying and commenting. We use S, D, Q and C as the abbreviations of different kinds of stances respectively. As shown in Fig. 9, we manually label the user stance by inferring directly from the tweets content for simplicity. However, a better approach is to classify users' sentiments automatically with the help of the Convolutional Neural Network (Yoo, Song, & Jeong, 2018), especially when analyzing the entire propagation graph.

We construct the propagation graph based on the propagation structure in Fig. 9. We use PGNN algorithm to generate representations for all nodes and calculate the similarity between each node representation and the mean. The similarity is measured by cosine distance to measure and the results are shown in Fig. 10. The X-axis in Fig. 10 represents the node index and the Y-axis represents the similarity. We find out that the representations of node 8 and node 14 have a large difference from the mean. As mentioned in Section 4.3, our models pay more attention on the node whose representation is very different from the mean value, so node 8 and node 14 ought to make more contribution than other nodes when identifying the type of the source tweet. It is apparent in Fig. 9 that the local contexts where node 8 and node 14 are located encompass many conflict views. Previous study finds that there are many conflict views in the local context of the tweet related to false rumors (Jin, Cao, Zhang, & Luo, 2016). As mentioned in Section 4.2, our proposed models exchange information between different nodes via relation paths, so that the learned representations contain discriminative features of the context information, which is beneficial to solving the downstream rumor detection task and early detection task.
better performance even when the data is very little. Besides, the accuracy of our proposed models climb rapidly as the tweet count or the elapsed time increases, suggesting that the proposed models possess strong learning ability.

We list three limitations of the methodology adopted in this work, other researchers who want to make a profound study can make an improvement on these shortcomings. First, we construct the propagation graph based on the who-replies-to-whom structure, however, the follower-followee relationships and the forward relationships are ignored. Second, the neural network has many training parameters, parameter tuning is tedious and requires certain skills. As we discussed in Section 5.2.3 and Section 5.2.4, it is non-trivial work to turn parameters and select the optimal model. Third, when we embed the information of tweets content into low-dimensional space, we merely consider the textual information, however, tweets sometimes include pictures, emoticons and videos in addition to text. Take into consideration various forms of information may improve the performance of the models.

It is worth mentioning that each module of our proposed model has good scalability and low coupling. Other researchers may easily change the methodology used in each single module to achieve almost the same objective in this work. In Section 4.1, we have discussed how to embed the textual information into low-dimensional space with Doc2vec algorithm. Some other word embedding methods, such as word2vec and Glove, can also be adopted to extract the linguistic features of the tweets content. In Section 4.2, we have proposed a GRU-based propagation graph neural network to exploit the propagation graph. Recently, more and more graph neural networks have been proposed, such as GAT and GCN (Zhou et al., 2018). These algorithms are powerful in modeling non-Euclidean space data and capturing the internal dependencies of the graph, therefore they are very suitable for extracting the features of the propagation graph. We believe that some of these graph neural networks can achieve the same objective as PGNN does.

In the future work, we will try to integrate more social network information of users, such as the follower-followee relationships, co-participation relationships in the community. The social relationship inherently constitutes a graph structure which can be used to further enrich the propagation graph. In addition, most of the rumor detection algorithms proposed so far are under the framework of supervised learning, but what is frustrating is that the data in real world is mostly unlabeled. Therefore, we plan to solve the rumor detection task with the help of transfer learning methods.

CRediT authorship contribution statement

Zhiyuan Wu: Conceptualization, Funding acquisition, Methodology, Visualization, Writing - original draft, Writing - review & editing. Dechang Pi: Funding acquisition, Project administration, Supervision. Junfu Chen: Investigation, Validation. Meng Xie: Investigation. Jianjun Cao: Validation.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

The research work is supported by National Natural Science Foundation of China (U1433116) and the Fundamental Research Funds for the Central Universities (NP2017208).
