Social Media Rumour Detection Through Graph Attention Networks

Xinpeng Zhang[§], Shuzhi Gong[§], Richard O. Sinnott
School of Computing and Information Systems
The University of Melbourne, Melbourne, Australia
rsinnott@unimelb.edu.au

2021 IEEE Asia-Pacific Conference on Computer Science and Data Engineering (CSDE) | 978-1-6654-9552-3/21/$31.00 ©2021 IEEE | DOI: 10.1109/CSDE53843.2021.9718466

***Abstract*—Rumours are unverified statements or news that spread quickly across the Internet. The global ubiquity of social media platforms provides the perfect conditions for the spread of rumours. Such rumours can have global consequences. Tools for detection of rumours are therefore needed. Diverse methods have been applied to discover rumours through approaches based on text mining, propagation patterns and user networks and their interactions. Such approaches treat user interactions in discussions equally. In this paper, we propose a model to extract information from user interactions based on Graph Attention Networks. In the propagation graph, the nodes represent the user text content and the edges represent the reply interactions. The attention mechanism is implemented to determine the edge weights between node pairs. We conduct experiments using Twitter15, Twitter16, and PHEME datasets and achieve state of the art results.**

***Index Terms*—Rumour Detection, Graph Attention Networks**

I. Introduction

The rapidly developing Internet and growth of social media platforms provides globally adopted ways for people to communicate and share news online. This news can be factually correct, misleading, unverified or indeed deliberately false information - so called fake news. Rumours are the manifestation of the spread of unverified statements. Social media users typically do not have the ability to identify the authenticity of information, hence rumours can spread widely and quickly, bringing huge economic and social negative impacts. For example, in 2013, $130 billion in stock value was wiped out in a matter of minutes following an Associated Press tweet about an 'explosion' that injured Barack Obama [1]. The outcome of the 2016 United States presidential election was also in part driven by the use of rumours and false information sent through social media platforms [2]. It is thus essential to detect rumours online as early as possible to minimise their subsequent impact. This is the focus of this paper.

Early rumour detection methods based on hand-crafted language/feature extraction. For an online post, the user information, text content and information propagation patterns were manually extracted and subsequently utilized to train (supervised) rumour classifiers. User posting and re-posting (reply) behaviors were embedded as features in decision trees [3], through Support Vector Machines [4] and through Random Forest based approaches [5]. These methods were inefficient however due to the need for complex manual feature engineering, which was time-consuming, labor-intensive and lacked higher level propagation representations [6].

In recent years, many deep learning based rumour detection methods have been put forward: recurrent neural networks (RNNs) [7] and convolutional neural networks (CNNs) [8] have been explored for their ability to make rumour predictions. Other have explored the 'self-correct' effect among social media users [9], i.e. where different tweets which reply to a given source post represent different opinions and offer the ability to correct the original source post To extract information from replies to a source post, a recurrent neural network was used by Ma et al [10] to traverse all nodes in the propagation tree to establish a recursive neural networks (RvNN). Ma also introduced some variants of RvNN, such as the attention-based RvNN [11] and the Tree Transformer [12]. The RvNN and its variant successes mainly come from information extracted from the propagation tree structure. They implemented a recursive mechanism to traverse nodes in the propagation tree one by one to learn the non-Euclidean relationship between them. Information in such recursive models can then be propagated either from parent to child or vice versa.

In addition to extracting reply relationships from tree structures, [1] flattened the tree structure and arranged all posted tweets in a chronological order. A self-attention mechanism was then implemented to allow all possible pairwise interaction between tweets in an event.

Recent studies have explored non-Euclidean relationships in tree structures through graph based approaches. To extract high-level interactions in propagation trees more efficiently, graph neural networks (GNN) and graph convolutional network (GCN) have been used to detect and debunk online rumours [6], [13], [14]. Although many models have extracted high-level information from such conversations, few of them learned the pairwise relationships from the data. Pairwise relationships can be regarded as the impact of the reply to a given source post. For example as seen in Fig. 1, User 0 posts an unverified news (in the green rectangle) and there are several replies associated with it. Most of these replies express similar views to the source post except for User 7's statement which doubted the source post's authenticity (in the red rectangle). In this example, User 7's reply forms a key source to determine whether the source post is a rumour. However, most existing methods neglect the different relationships between different

[§]Xinpeng and Shuzhi contracted equally.

post-reply pairs. In graph based methods for example [6], [13], [14], the edges in the propagation tree are treated equally, while different replies to source posts can convey different statements with opposite/contradictory opinions to the source post.

To learn the edges in the propagation graph from the text content, attention based graph technology and specifically Graph Attention Networks (GAT) [15] are utilised. In our model, the node features in a graph are the text content of posts, and there is an edge with variable weights between every two nodes (posts) within the reply relationship. The weight of that edge is determined via attention mechanisms based on the contents of the corresponding nodes. After all edge weights are computed in the graph, we update the node features. These updated node features can then be used for prediction about whether the source post is potentially a rumour as presented in Section III.

The contributions of this paper are:

- a novel method that implements graph attention networks to tackle rumour detection;
- a rigorous set of experiments that outperform existing benchmarks.

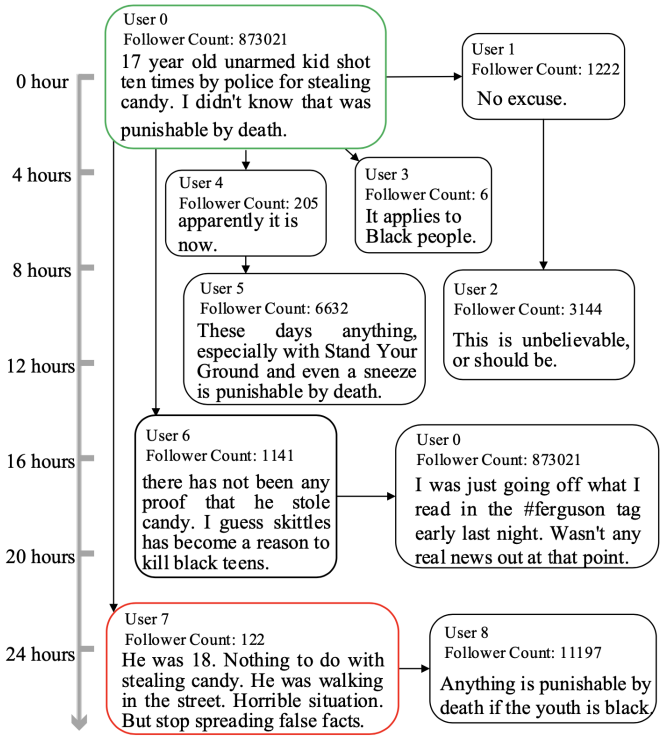

Fig. 1: A Twitter Post-Reply Example of a Rumour

II. Related Work

A. Rumour Detection

Early rumor prediction methods relied on hand-crafted feature engineering to extract features such as the text content, user information and indicative propagation patterns. These features were then utilizes to train supervised classifiers such as Decision Trees, Random Forests and Support Vector Machine [6]. With the rise of deep learning, deep structured models have been explored for rumor detection. To learn the continuous hidden representations in contextual information, a recurrent neural network (RNN) based model was proposed [7]. To capture the propagation patterns from replies, tree structured recursive neural models (RvNN) were introduced by [10]. Around the same time, a self-attention based transformer model was put forward to handle several natural language processing problems [16]. Recently, Graph Neural Networks (GNN) and Graph Convolutional Networks (GCN) [17] have been put forward. A convolutional based method following RvNN's two directional design for a Bi-Directional Graph Convolutional Networks (Bi-GCN) was introduced [6] that offered state-of-art performance for rumour detection based on several benchmarks.

B. Attention Mechanism

The attention mechanism stemmed from neural machine translation to solve sequence-to-sequence model issues. In the sequence-to-sequence model, a fixed length context vector is generated from an encoder via compressing the inputs, then the context vector is delivered to the decoder to emit the outputs. The issue of such sequence-to-sequence models however is that the fixed length context doesn't have enough capacity to deal with ad hoc or long sequences. To solve this [18] created shortcuts between the context vector and the entire source input, which is at the heart of the attention mechanism. Such a design allows the context vector to be based on the entire input sequence, and in so doing provides capacity for long-term memory challenges. The details of attention mechanism implemented in this paper are described in section III.

The attention mechanism in [18] bridged the gap between the source inputs and target outputs to provide a huge performance improvement in neural translation. The attention has also been adapted in other areas such as automatic image description generation in the field of computer vision [19]. There are also many attention mechanism variants that have been put forward [18]. These include content-based attention [20], location-based attention [21], general attention [21], dot-product attention [21] and scaled dot-product attention [16]. These variants implement different alignment score functions which are utilized to estimate the weights of shortcuts between contexts and inputs. In addition to relating contexts and inputs, self-attention mechanisms introduced in [22] allow to relate the different positions in a given sequence. For example, when processing a word in a sentence, it is possible to refer to other words based on the correlation extracted from the self-attention mechanism.

A full attention model transformer was built on a self-attention mechanism without recurrent network units [16]. The transformer introduced multi-head attention and achieved state-of-the-art performance in machine translation. Building on this, many transformer variants have been introduced and applied in the field of machine learning.

C. Graph Neural Networks

Convolutions are usually generalized in the graph neural network domain to aggregate neighborhood information. Graph Convolutional Network (GCN) was one of the earliest effective convolution graph models. It was designed with a "message-passing" architecture that depended on an adjacency matrix, hidden feature matrix and trainable parameters. There are many kinds of "message-passing" functions. A first-order approximation from [17] is shown below:

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

Where A is the normalized adjacency matrix, H is the hidden feature matrix, W is the trainable parameters, and σ is the activation function. GraphSAGE is another graph model, which provides a solution to learn node embeddings in an inductive way. Compared to GCN, GraphSAGE does not require an adjacency matrix, rather nodes are represented by the aggregation of neighborhoods. Thus, even if a new node is unseen during training appears in the graph, it can still be properly represented by its neighboring nodes. GAT [15] is a cutting-edge graph model. Within GAT, each layer expands the basic aggregation function of the GCN layer by assigning importance to each edge through attention coefficients.

III. Methodology

A. Notations

We define a rumour detection dataset as a set of events. Each event is composed of one source tweet (post) and all relevant responding tweets (replies) as shown in Fig. 1. In the Fig. 1 example, there is 1 source post with a green rectangle and 9 reply posts, which can be well represented as a tree or graph structure. For each event that includes the source and reply tweets, we develop a graph G(V,E) where V is a set of vertices representing the tweet texts and E is a set of paired vertices/edges representing the reply relationship to the event.

To define the direction of the graph's edges, we can follow the Top-down and Bottom-up design as proposed in [10]. The Top-down structure is based on the direction of information propagation. If u was posted earlier than v and v was the reply of u, then the information is propagated from u to v and the edge is from u to v. The Bottom-up structure is the reverse version of Top-down, i.e. for the same u, v as described above, the Bottom-up edge is from v to u.

The input of our model is a set of node features representing posts (text) content in an event based on a bag-of-word (BOW) form, where $\vec{h} = \left\{\vec{h_1}, \vec{h_2}, ..., \vec{h_N}\right\}$, $\vec{h} \in R^F$ where N is the number of nodes in a graph, and F is the dimension of node features. In rumour detection, F is usually the size of the BOW. In addition to the node features, we also use an adjacent matrix, $Adj = \{A_{TD}, A_{BU}\}$, where A_{TD} is the Top-down graph's adjacent matrix and A_{BU} is the Bottom-up graph's adjacent matrix.

B. Graph Attention Network Layer

To represent the non-Euclidean relationship between nodes in an event and hence to determine the edges' weight based on the content of the corresponding nodes, we implement graph attention networks. We follow a similar structure of many existing graph based rumour detection methods, where the texts of posts are represented by graph's node features and the reply relationships are described by the graph's edges. However, these methods all treat the edges equally, which means all neighbors of one node will have same influence on that node. Given the nature of rumour propagation, different replies can be quite different as shown in Fig. 1 where the reply with red rectangle expresses clear doubts about the content of the source tweets. Therefore, while determining whether the source tweet is a rumour, the reply with the red rectangle is likely to impact rumour prediction far more than others. Given this, to capture the high-level information in the rumour propagation graph mentioned, it is necessary to evaluate each node pairs edge weights based on their contents.

Following the design of the graph attention network, the actual attention mechanism needs to be implemented to determine the edge weights in the rumour propagation graph. In the original attention mechanism design, every node needs to be updated when one nodes features changes. However, since the rumour propagation graph has a tree-like connected characteristic, it is meaningless to refer to node pairs that are not connected directly with edges. Therefore, a masked attention mechanism is implemented, where the attention coefficients are only computed between two connected nodes.

Given a node pair (i, j), the coefficient of the edge is calculated by the attention mechanism based on:

$$\alpha_{ij} = \frac{exp(LeakyReLU(\overrightarrow{\boldsymbol{a}}^T[\boldsymbol{W}\overrightarrow{h_i}||\boldsymbol{W}\overrightarrow{h_j}]))}{\sum_{k\in N_i} exp(LeakyReLU(\overrightarrow{\boldsymbol{a}}^T[\boldsymbol{W}\overrightarrow{h_i}||\boldsymbol{W}\overrightarrow{h_k}]))}$$

where N_i is the neighbor set of node i, LeakyReLU is an activation function, a and W are learnable parameters, and h_i and h_j are features of the node pair (i, j).

Once the coefficients between all neighboring node pairs has been obtained, the node features can be updated. The normalized attention coefficients are used to compute a linear combination of the features corresponding to them. These are then used to serve as the final output features for every node based on:

$$\overrightarrow{h_i}' = \sigma(\sum_{j\in N_i} \alpha_{ij}^k \boldsymbol{W}\overrightarrow{h_j})$$

C. Model Structure

Using the graph attention layer described in Section III, it is possible to compute the edges in the propagation tree and update the node features through graph convolutional operations. In our work, we stacked two graph attention layers with residual connection in both the Top-down and Bottom-up directions and concatenate the outputs of two directions for use in final rumour predictions as shown in Fig. 2.

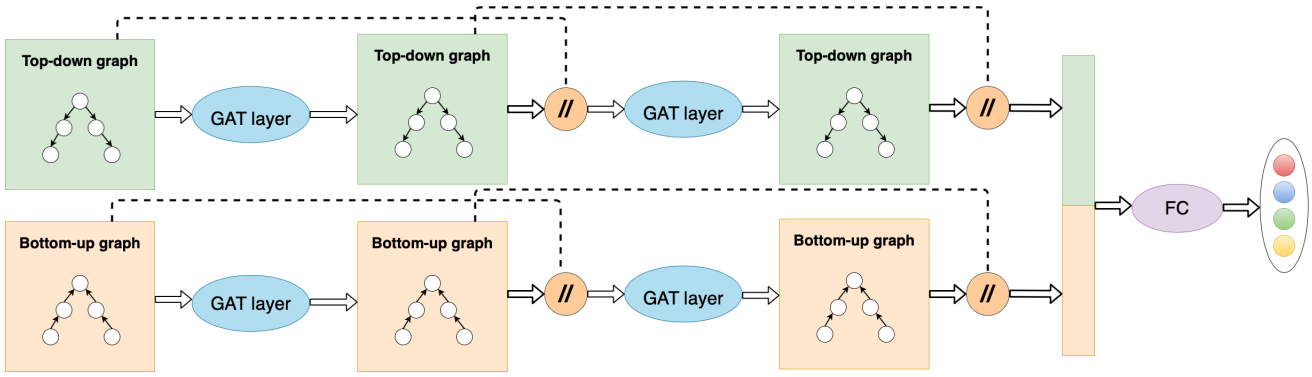

Fig. 2: Architecture of a Residual Graph Attention-based Network

In our model, graph attention network layers are used to extract information from top-down and bottom-up two information propagation directions. To avoid parameter vanishing and to training the model efficient, residual connections are added in the network between two graph attention network layers as shown in Fig. 2.

The inputs of the GAT layer comprise node features in one event that represent the text content of each post. Then the GAT layers computes the weights of edges and modulates the node features based on those edges. The residual skip connection used is based on [23] where it was used for image recognition.

The goal of training is to minimise the cross entropy loss function of the probability distribution of the prediction $\widehat{y}$ and the ground truth y. The loss function formula is given as:

$$Loss = -\sum_{i=1}^{N}\sum_{c=1}^{C} y_{i,c} \cdot log(\widehat{y}_{i,c}) + \lambda \left\|\Theta\right\|_2^2$$

where $y_{i,c}$ is the ground truth of the i_{th} sample in the cth class and this is equal to 1 if $y_{i,c}$ belongs to the c_{th} class and 0 otherwise, and $\widehat{y}_{i,c}$ is the probability of the prediction that the i_{th} sample belongs to the c_{th} class. Here N is the number of training data and C is the number of rumour classes, $\left\|\cdot\right\|_2$ is the L_2 regularization term for all parameters θ in the network, and λ is the trade-off coefficient.

IV. Experiments and Results

A. Datasets

For training and evaluation, several publicly available datasets were utilized in the experiments including Twitter15 [24], Twitter16 [24], and PHEME [25]. The Twitter15 and Twitter16 are four labelled datasets, where each source post is labelled in one of four categories: non-rumour, false rumours, true rumours and uncertain rumours. The PHEME dataset contains only two labels: rumour and non-rumour. The amount and type of data used in the work is shown in Table. I

TABLE I: Statistics of the Datasets

Statistic	Twitter15	Twitter16	PHEME
# of source tweets	1,490	818	5802
# of tree nodes	76,351	40,867	30,376
# of non-rumours	374	205	3,830
# of false rumours	370	205	1,972
# of true rumours	372	205	-
# of unverified rumours	374	203	-
Avg. time length / tree	444 Hours	196 Hours	18 Hours
Avg. # of posts / tree	52	50	6
Max # of posts / tree	814	757	228
Min # of posts / tree	1	1	3

B. Baseline Methods

The poor performances of hand-crafted feature engineering based methods have been verified in some studies [11], [12], therefore manual feature engineering methods are not compared here. Since our model utilises deep learning to detect rumours, state-of-the-art deep learning based rumour detection baselines which explore information from post contents and propagation patterns are compared in our experiments. These include:

- **RvNN**: Recursive Neural Network, a recurrent neural network (GRU) with top-down and bottom-up tree structure recursive exploration [10].
- **PLAN**: Post-level Attention Model, a self-attention based method that flattens the propagation tree and attends to every two posts in given events [1].
- **Bi-GCN**: Bidirectional Graph Convolutional Network, a bidirectional graph convolutional neural network that extracts information from the propagation tree [6].
- **Attention-based RvNN**: Attention-based Recursive Neural Network, a variant of RvNN in which self-attention mechanisms are implemented to detect evidential posts [11].
- **Tree Transformer**: Tree Transformer, a Transformer [16] encoder with a top-down and bottom-up tree structure recursive exploration similar to RvNN [10].

All baselines and our proposed model are implemented with Pytorch. To make a fair comparison, the Twitter15 and Twitter16 datasets are trained with 5-fold cross-validation. Some existing experiments such as [6] collect the best validation results as final results, which is unfair because the 5-fold cross-validation will split data randomly. To get robust results, we divide the Twitter15 and Twitter16 datasets into 5-fold many

TABLE II: Rumour Detection Performance on Twitter15 and Twitter16 Datasets. The accuracy and F1-Score for each individual class are reported. (NR: non-rumour; FR: false-rumour; TR: true-rumour; UR:unverified-rumour)

Method	Twitter15					Twitter16				
	Accuracy	NR	FR	TR	UR	Accuracy	NR	FR	TR	UR
BU-RvNN	0.708	0.695	0.728	0.759	0.653	0.718	0.723	0.712	0.779	0.659
TD-RvNN	0.723	0.682	0.758	0.821	0.654	0.737	0.662	0.743	0.835	0.708
PLAN	0.845	0.823	0.858	0.895	0.802	0.874	**0.853**	0.839	**0.917**	**0.888**
Bi-GCN	0.821	0.792	0.821	0.862	0.811	0.851	0.795	0.842	0.926	0.839
Attention-based RvNN	0.842	0.811	**0.875**	0.818	0.790	0.863	0.820	0.898	0.843	0.837
Tree-Transformer	0.816	0.759	0.794	0.826	0.783	0.823	0.835	0.781	0.796	0.801
GAT(ours)	**0.856**	**0.850**	0.830	**0.901**	**0.844**	**0.881**	0.835	**0.909**	0.909	0.864

TABLE III: F1 Performance on PHEME Datases (R: rumour; NR:non-rumour)

Statistic	Macro-F1	R-F1	NR-F1
BU-RvNN	0.742	0.622	0.862
TD-RvNN	0.741	0.607	0.874
PLAN	0.774	0.721	0.827
BiGCN	0.870	0.830	0.911
Attention-based RvNN	0.831	0.767	0.894
Tree-Transformer	0.805	0.730	0.881
GAT	**0.891**	**0.880**	**0.901**

times, and conduct cross-validation on the fixed split datasets. The results with the lowest evaluation error are then selected as the final results.

We set up all baselines with their original parameter settings. For our proposed model, the Bag-of-Word (BOW) representations of text contents comprise the initial node features in the graph. In the Twitter15 and Twitter16 datasets, the BOW length is 5,000, and in the PHEME dataset the BOW length is 6,500. The two graph attention layers' hidden states are set with size 1024 and 512 in both Top-down and Bottom-up directions respectively.

C. Rumour Detection Performance

The Twitter15 and Twitter16 are balanced datasets which means the amount of instances in each class are equal. However, the PHEME is an unbalanced dataset. Less than 30% of the instances are labeled as rumours. Therefore, if we assume all instances are non-rumours we still get an accuracy greater than 0.7. To make a meaningful evaluation, the F1-Score is selected as the metric. The performance of all compared models is shown in Tab. II and Tab. III.

Among all compared models, our Graph Attention Network based model performs best. We found that the Post-level Attention Model (PLAN) also performs well with the Twitter15 and Twitter16 datasets. From the conceptual design, our proposed model and PLAN have very similar ideas but different structure and implementation details. We both implement attention mechanism to attend to pair-wise relationships between posts. However, the PLAN flattens the propagation tree into a sequence, and attend to every two posts in the sequence. The design of PLAN is very similar to Transformer [16] based self-attention models. The difference is that tokens in Transformer are substituted with posts in PLAN - hence the name. In our GAT model, we only attend to post pairs with a reply relationship, and the attention mechanism is implemented to determine the weight of that relationship.

From the results of the PHEME dataset, the shortcomings of PLAN show up. With the PHEME dataset, many events contain one post with many retweets. Usually users post nothing in retweets, hence the PLAN will get less information when it attends to pairs containing retweets. That is, PLAN can only learn the propagation patterns of rumours that are spread when there is enough information in the reply posts. If we consider other models, we can find that the RvNN, tree Transformer and Bi-GCN only learn fixed propagation patterns because they don't adjust edge weights in the propagation graph. The Attention-based RvNN learns using variable edge weights, however its recursive design is equivalent to flattening the tree into several sequences, such that each sequence represents an information flow from the root to a tree leaf. Then the attention mechanism attends to every two neighbors in the sequence. Both its structure and attention mechanism implementation are less efficient than ours.

Tab. II and Tab. III show that our GAT model outperforms all baselines. From the design, our GAT model proposes a trade-off between attending to all possible post pairs and learning from fixed propagation patterns. Furthermore, our model extracts information from graphs globally instead of from information flows as is the case with RvNN, Attention-based RvNN and Tree Transformers, therefore it exhibits better performance.

V. Conclusions and Future Work

In this paper, we propose a GAT-based model for rumor detection through social media. In this model, we enhance every node embedding in the graph by concatenating it with source content higher-level representations. We use bottom-up and top-down GAT structures to represent the deep propagation and wide dispersion of rumours through use of a residual neural network. This approach outperforms state-of-art approaches by a considerable margin both in terms of accuracy and model efficiency.

There are various possible extensions to this work. One of the current shortcomings is that we haven't extracted information from the user characteristics. User information can represent additional relationships (e.g. follower relationships) in addition to the responsive relations mentioned in this paper. Such information is informative as discussed in [26]. Besides, since the online posts contain not only text content, it is also

worth exploring the role of non-textual media such as images or videos with regards to how they can be used to spread rumors and the unique challenges that this gives rise to.

REFERENCES

[1] L. M. S. Khoo, H. L. Chieu, Z. Qian, and J. Jiang, "Interpretable rumor detection in microblogs by attending to user interactions," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 05, 2020, pp. 8783–8790.

[2] R. Gunther, P. A. Beck, and E. C. Nisbet, "Fake news did have a significant impact on the vote in the 2016 election: Original full-length version with methodological appendix," *Unpublished manuscript, Ohio State University, Columbus, OH*, 2018.

[3] C. Castillo, M. Mendoza, and B. Poblete, "Information credibility on twitter," in *Proceedings of the 20th international conference on World wide web*, 2011, pp. 675–684.

[4] F. Yang, Y. Liu, X. Yu, and M. Yang, "Automatic detection of rumor on sina weibo," in *Proceedings of the ACM SIGKDD workshop on mining data semantics*, 2012, pp. 1–7.

[5] S. Kwon, M. Cha, K. Jung, W. Chen, and Y. Wang, "Prominent features of rumor propagation in online social media," in *2013 IEEE 13th international conference on data mining*. IEEE, 2013, pp. 1103–1108.

[6] T. Bian, X. Xiao, T. Xu, P. Zhao, W. Huang, Y. Rong, and J. Huang, "Rumor detection on social media with bi-directional graph convolutional networks," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 01, 2020, pp. 549–556.

[7] J. Ma, W. Gao, P. Mitra, S. Kwon, B. J. Jansen, K.-F. Wong, and M. Cha, "Detecting rumors from microblogs with recurrent neural networks," 2016.

[8] F. Yu, Q. Liu, S. Wu, L. Wang, T. Tan *et al.*, "A convolutional approach for misinformation identification." in *IJCAI*, 2017, pp. 3901–3907.

[9] A. Zubiaga, A. Aker, K. Bontcheva, M. Liakata, and R. Procter, "Detection and resolution of rumours in social media: A survey," *ACM Computing Surveys (CSUR)*, vol. 51, no. 2, pp. 1–36, 2018.

[10] J. Ma, W. Gao, and K.-F. Wong, "Rumor detection on twitter with tree-structured recursive neural networks." Association for Computational Linguistics, 2018.

[11] J. Ma, W. Gao, S. Joty, and K.-F. Wong, "An attention-based rumor detection model with tree-structured recursive neural networks," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 11, no. 4, pp. 1–28, 2020.

[12] J. Ma and W. Gao, "Debunking rumors on twitter with tree transformer." ACL, 2020.

[13] H. Lin, X. Zhang, and X. Fu, "A graph convolutional encoder and decoder model for rumor detection," 10 2020.

[14] L. Zhang, J. Li, B. Zhou, and Y. Jia, "Rumor detection based on sagnn: Simplified aggregation graph neural networks," *Machine Learning and Knowledge Extraction*, vol. 3, no. 1, pp. 84–94, 2021.

[15] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph attention networks," *arXiv preprint arXiv:1710.10903*, 2017.

[16] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in neural information processing systems*, 2017, pp. 5998–6008.

[17] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *arXiv preprint arXiv:1609.02907*, 2016.

[18] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," *arXiv preprint arXiv:1409.0473*, 2014.

[19] K. Xu, J. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhudinov, R. Zemel, and Y. Bengio, "Show, attend and tell: Neural image caption generation with visual attention," in *International conference on machine learning*. PMLR, 2015, pp. 2048–2057.

[20] A. Graves, G. Wayne, and I. Danihelka, "Neural turing machines," *arXiv preprint arXiv:1410.5401*, 2014.

[21] M.-T. Luong, H. Pham, and C. D. Manning, "Effective approaches to attention-based neural machine translation," *arXiv preprint arXiv:1508.04025*, 2015.

[22] J. Cheng, L. Dong, and M. Lapata, "Long short-term memory-networks for machine reading," *arXiv preprint arXiv:1601.06733*, 2016.

[23] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[24] J. Ma, W. Gao, and K.-F. Wong, "Detect rumors in microblog posts using propagation structure via kernel learning." Association for Computational Linguistics, 2017.

[25] E. Kochkina, M. Liakata, and A. Zubiaga, "Pheme dataset for rumour detection and veracity classification," Jun 2018.

[26] Q. Huang, C. Zhou, J. Wu, M. Wang, and B. Wang, "Deep structure learning for rumor detection on twitter," in *2019 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2019, pp. 1–8.