PESTO: A Post-User Fusion Network for Rumour Detection on Social Media

Anonymous ACL submission

Abstract

Rumour detection on social media is an important topic due to the challenges of misinformation propagation and slow verification of misleading information. Most previous work focus on the response posts on social media, ignoring the useful characteristics of involved users and their relations. In this paper, we propose a novel framework, Post-User Fusion Network (PESTO), which models the patterns of rumours from both post diffusion and user social networks. Specifically, we propose a novel Chronologically-masked Transformer architecture to model both temporal sequence and diffusion structure of rumours, and apply a Relational Graph Convolutional Network to model the social relations of involved users, with a fusion network based on self-attention mechanism to incorporate the two aspects. Additionally, two data augmentation techniques are leveraged to improve the robustness and accuracy of our models. Empirical results on several benchmarks show the superiority of the proposed method.

1 Introduction

Rumours, are unverified statements found in social media platforms, which can be damaging if they spread false information with social, economic and political impact (Del Vicario et al., 2016; Zubiaga et al., 2018). For instance: during the period of the U.S. 2016 presidential election, almost 529 different rumours about candidates were propagated on Facebook and Twitter which influenced voters' attitudes (Jin et al., 2017). To this end, it is important for social network platforms to develop effective strategies to combat against fake news and rumours. Recognising misinformation from social media is challenging due to different sources of information required to gather in order to conduct an extensive analysis and reasoning on these sources. Early efforts to tackle rumour detection and misinformation in social network platforms relied on manual verification from users or experts, however, this kind of approach is inefficient due to the substantial human effort and time to recognise a rumour after it has emerged. In recent years, automatic social context based rumour detection has attracted increasing attention. This area of research utilizes the collective wisdom of the social platforms by extracting signals from comments and/or replies towards a source claim (Ma et al., 2016, 2017, 2018; Han et al., 2019; Kochkina et al., 2018; Yuan et al., 2019; Bian et al., 2020a; Khoo et al., 2020; Kochkina and Liakata, 2020; Huang et al., 2019). The key idea behind these work is that users from social media would contribute opinions, clues and evidence for distinguishing between false and valid information for rumour detection. Therefore, the content of communication threads and the interaction between posts would be useful for rumour detection. However, apart from the threads of responses, the characteristics of the social network of users can also provide important clues for inferring news veracity. For example, eye-catching rumours usually attract mostly bot accounts to spread, who tend to follow many accounts but with few or no followers (Gilani et al., 2019), such implicit patterns can also support the veracity of a claim. Figure 1 is an example of rumour spreading, showing that both post diffusion patterns (e.g., stance in posts, responsive structures) and user patterns (e.g., user credibility, social network) provide crucial evidences for rumour verification. Our aim is to propose a method which can model the post diffusion and the user social network jointly to detect social rumours. In terms of post diffusion modeling, a typical line of methods have exploited the characteristics of diffusion structure, such as tree-structured RvNN (Ma et al., 2018), Bi-GCN (Bian et al., 2020b)s and DSL (Huang et al., 2019), but ignore the temporal information and the implicit connections between posts. Sequence-based models such Recurrent neural networks (RNNs) (Ma et al., 2016),

PLAN (Khoo et al., 2020) and DCM (Veyseh et al., 2019) flatten the tree structure and arrange posts in chronological order. They overcome some limitations of tree models but underexploit the diffusion structure. For this sake, in the paper, we propose a Chronologically-masked Transformer architecture, which integrates both temporal and structural information to effectively model the rumour diffusion patterns. In terms of user network modeling, many off-the-shelf graph neural networks such as Graph Convolutional Network (GCN) (Kipf and Welling, 2016), GraphSAGE (Hamilton et al., 2017), Graph Attention Network (GAT) (Velivcković et al., 2017), Relational Graph Convolutional Network (RGCN) (Schlichtkrull et al., 2018) can be leveraged. Considering that A-follow-B and A-followed-by-B are different relations, we adopt RGCN for user network representation. In order to fuse the information in two aspect, we propose to use a self-attention layer for final information aggregation. Since many existing rumour detection datasets are in small scale, we propose two data augmentation techniques: Connection dropping and Sub-conversation training to assist model training. We name the entire architecture as Post-User Fusion Network (PESTO). Our experimental evaluation shows PESTO improves performance over previous approaches. The contributions of our work are as follows:

- We propose a Chronologically-masked Transformer architecture to model the post diffusion patterns of rumours, with both temporal and structural information considered.
- We leverage a Relational Graph Convolutional Network to represent the user social network, and integrate it with the chronologically-masked Transformer via a Fusion network based on self-attention.
- We propose two data augmentation techniques: Connection dropping and Sub-conversation training, to reduce overfitting, making our model more robust and stable.

2 Related Work

Existing detection approaches of fake claims can be generally categories into three groups based on the information utilized: (i) the content of the claim, (ii) knowledge from trustworthy sources and (iii) social response to the claim. Our work in this paper falls into the last group, which exploits social

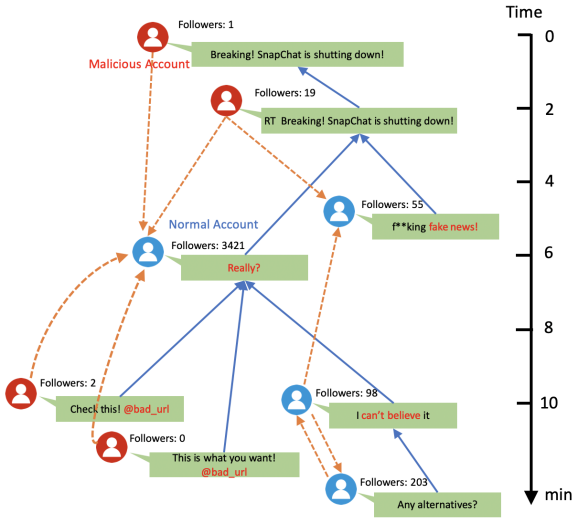


Figure 1: An example of a rumour spreading on Twitter platform. The blue lines denote responsive relations and the orange lines denote following relations. The malicious account is the initial spreader of the fake news. The normal respondents tend to query, oppose or comment on the news, but the malicious accounts/bots are likely to target the people with many followers and reply them with links of low-credibility content to get a lot of visibility. Besides, they tend to follow the spreaders but few users follow them. In summary, the diffusion patterns of rumours and the involved user patterns implies the veracity of claims.

replies and the involved user network to detect rumours. In this section, we briefly introduce each group of work.

Content-based Detection: This line of studies studied specific linguistic cues such as verb quantity, word classes, word length, pronouns, non-objectivity (Rubin and Lukoianova, 2015; Feng et al., 2012; Potthast et al., 2017). These features are useful to detect satires or onion news, but might be unique to domains or topics.

Knowledge-based Detection: Fact checking websites such as politifact.com and snope.com leverage manual verification to debunk fake news or rumours, but fail to match the rapid emergence rate of misinformation nowadays. Automated fact checking techniques rely on truthworthy sources such as Wikipedia, but they might not work for latest news without evidences.

Social Response-based Detection Social response information such as reply contents and propagation structures have been shown to be particularly useful for classifying rumours. Ma et al. (Ma et al., 2017) uses tree kernel to capture the similarity of propagation trees by counting their similar sub-structures in order to identify different types

of rumours on Twitter. Ma et al. (Ma et al., 2018) make use of tree-structured recursive neural network to model the propagation tree, and information from different nodes is aggregated recursively in either a bottom-up or a top-down manner. Bian et al. (Bian et al., 2020a) also propose a bi-directional graph model named Bi-GCN to explore both propagation and aggregation patterns by operating on both top-down and bottom-up propagation of rumours. However, the focus in these works is on using the static tree structure of Tweet propagation, ignoring the temporal order and implicit connections between posts. For this sake, Veyseh et al. (Veyseh et al., 2019) and Khoo et al. (Khoo et al., 2020) propose to apply self-attention mechanism (Vaswani et al., 2017) to model implicit connections, but their direct usage of self-attention does not consider the propagation and aggregation characteristic of news conversation and underexploit the explicit diffusion structure. All of previous work do not take user networks into consideration, which provides important evidences for detection (Yang et al., 2019; Shu et al., 2019).

3 Preliminaries

3.1 Problem Statement

We define rumour detection as predicting the label (e.g., Rumour or Non-rumour) of a source post on social media, given all its responding posts and the response relations between them. A rumour detection dataset is a set of threads: $\mathbf{T} = \{T_1, T_2, ...T_{|\mathbf{T}|}\}$, where $T_i = \{p_1^i, p_2^i, ...p_{M_i}^i, u_1^i, u_2^i, ...u_{N_i}^i, G_i^P, G_i^U, G_i^{UP}\}$ is the i-th event, where M_i and N_i denotes the number of posts and involved users in T_i respectively, p_j^i denotes the j-th post and u_k^i denotes the k-th user. p_1^i is the source post and others are corresponding retweeted posts or responsive posts in chronological order. G_i^P is the propagation structure of posts. Specifically, G_i^P is defined as a graph $\langle V_i^P, E_i^P \rangle$, where $V_i^P = \{p_1^i, p_2^i, ..., p_{M_i}^i\}$, and $E_i^P = \{e_{i(st)}^P | s, t = 1, ..., M_i\}$ that represents the set of edges from responsive posts to responded posts. Likewise, G_i^U is defined as a graph $\langle V_i^U, E_i^U \rangle$, where $V_i^U = \{u_1^i, u_2^i, ...u_{N_i}^i\}$. and $E_i^U = \{e_{i(st)}^U | s, t = 1, 2, ..., N_i\}$ represents the set of edges from users to the users they follow. $G_i^{UP} = \{V_i^U \cup V_i^P, E_i^{UP}\}$ is the user-publish-post graph, where $E_i^{UP} = \{e_{i(st)}^{UP} | s = 1, ..., N_i, t = 1, ..., M_i\}$ denotes the set of edges from users to the posts they published. Each event T_i is associated with a ground-truth label $y_i \in \{F, T\}$ (i.e., False Rumour or True Rumour). In certain cases, the dataset contains four fine-grained class $\{N, F, T, U\}$ (i.e., Non-rumour, False Rumour, True Rumour and Unverified Rumour). We formulate this task as a supervised classification problem, which aims at learning a classifier f from labeled events, that is $f : T_i \rightarrow y_i$.

3.2 Architecture of Transformer

The Transformer model (Vaswani et al., 2017) employs an encoder-decoder architecture, consisting of stacked encoder and decoder layers. Each encoder layer consists of two sub-layers: a self-attention layer and a position-wise feed-forward network. The self-attention layer employs h attention heads. Each attention head operates on the same input sequence $\mathbf{X} = (\mathbf{x}_1, ..., \mathbf{x}_n)$ of n elements where $\mathbf{x}_i \in \mathbb{R}^d$, and computes a new sequence $\mathbf{Z} = (\mathbf{z}_1, ..., \mathbf{z}_n)$ of the same length where $\mathbf{z}_i \in \mathbb{R}^{d_k}$. To be specific, each $\mathbf{x}_i$ is firstly linearly transformed into a query vector, a key vector and a value vector:

$$\mathbf{q}_i = \mathbf{W}^Q \mathbf{x}_i, \mathbf{k}_i = \mathbf{W}^K \mathbf{x}_i, \mathbf{v}_i = \mathbf{W}^V \mathbf{x}_i, \quad (1)$$

where $\mathbf{W}^K, \mathbf{W}^Q, \mathbf{W}^V \in \mathbb{R}^{d_k \times d}$ are layer-specific trainable parameter matrices. Then, each element $\mathbf{z}_i$ is computed as the weighted sum of $\mathbf{v}_j$:

$$\mathbf{z}_i = \sum_{j=1}^{n} \frac{\exp(e_{ij})}{\sum_{k=1}^{n} \exp(e_{ik})} \mathbf{v}_j \quad (2)$$

and e_{ij} is the unnormalized attention score computed via a compatibility function, e.g., Scaled dot product, that compares $\mathbf{q}_i$ and $\mathbf{k}_i$, using:

$$e_{ij} = \frac{\mathbf{q}_i^{\mathrm{T}} \mathbf{k}_j}{\sqrt{d_k}}. \quad (3)$$

Note that all these parameter matrices, $\mathbf{W}^Q, \mathbf{W}^K, \mathbf{W}^V$, are unique for each attention head. Then, the outputs of all the attention heads are concatenated. Finally, the concatenated vector is fed to a parameterized linear transformation to obtain the output of the self-attention sublayer:

$$\hat{\mathbf{z}}_i = \mathbf{W}^O \, \mathrm{Concat}(\mathbf{z}_i^1, ..., \mathbf{z}_i^h). \quad (4)$$

Finally, a position-wise feed-forward network is used to produce the output node embeddings $\tilde{\mathbf{z}}_i$:

$$\tilde{\mathbf{z}}_i = \mathrm{FFN}(\hat{z}_i) = \mathbf{W}_2 \sigma(\mathbf{W}_1 \hat{z}_i + \mathbf{b}_1) + \mathbf{b}_2, \quad (5)$$

where, $\mathbf{W}_1, \mathbf{W}_2, \mathbf{b}_1, \mathbf{b}_2$ are parameters, σ is the non-linear function.

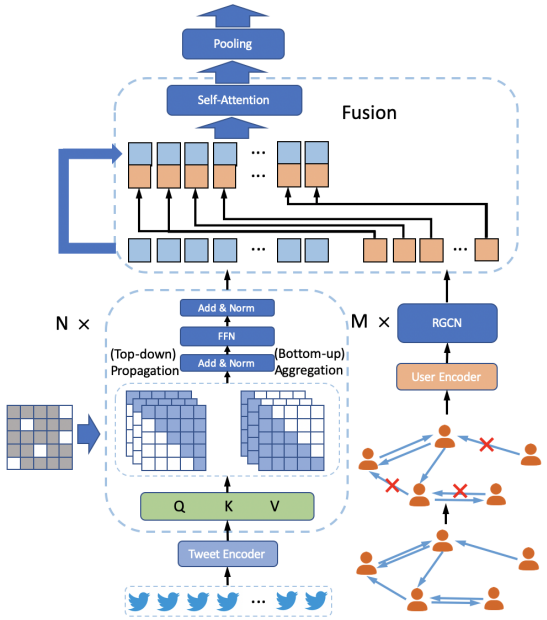


Figure 2: The architecture of PESTO. The left bottom part is the Chronologically-masked Transformer architecture, the right bottom part is the user network modeling architecture, with connection dropping mechanism applied to both parts. The upper part of the architecture is the fusion network for aggregation of the two views.

Table 1: Detailed meta features of post and user nodes

Type	Feature name	Example
Post	Post type	0/1/2*
	Retweet Count	10
	Reply Count	10
	Like Count	10
	Quote Count	10
	Created time	1501143981
	Sentiment Score	0.8
User	is_verified	1
	Following Count	100
	Followers Count	1000
	Tweet Count	1000
	List Count	10
	Account created time	1458483921
	Description length	20

* 0 denotes tweet, 1 denotes retweet, 2 denotes reply.

4 Methodology

4.1 Overview of Model Architecture

In this section, we introduce our proposed Post-User Fusion Network (PESTO). The core idea of PESTO is to learn discriminative representations for both post propagation tree and the user social network respectively, and then fuse them based on self-attention mechanism. The overall architecture of the proposed model is illustrated in Figure 2. Our model consists of four major parts: 1) Posts/User Feature Encoder, which encodes the text and meta features of a post/user into a dense vector. 2) Chronologically-masked Transformer, which learns the representation of the post tree. 3) Relational Graph Convolutional Network, which learns the representations of the user-follow network. 4) Fusion Network based on Self-Attention, which learns the global representation of post-user pairs.

4.2 Post/User Feature Encoder

Each post/user node contains two types of features: text features which are short sequences of words $\mathbf{x}$ and meta features $\mathbf{m}$ (e.g., follower count, following count, retweet count, etc). For each post, the text features are the post content, which contains distinctive patterns such as exaggerated expressions or negative stance, and for each user, the text features are the user description, which contains some bot-like flags or political stance that implies the credibility of users. We use the same encoder architecture to represent both post and user nodes. There are many methods to represent texts in rumour detection, such as TF-IDF (Aizawa, 2003), Convolutional Neural Network (CNN) (Kalchbrenner et al., 2014), LSTM (Hochreiter and Schmidhuber, 1997), Transformer (Vaswani et al., 2017) and BERT (Wolf et al., 2019). In our work, we apply word embeddings with CNN as our textual feature extractor, which shows the best performance and efficiency in our experiments. Specifically, we first embed each word in the text into a $k-$dimensional dense semantic representation using public pre-trained word vector Glove (Pennington et al., 2014). Then, a convolutional layer with window sizes of $2, 3, 4$ is applied, followed by a max-pooling layer to obtain the final text representation $\mathbf{h}_x$. After that, we concatenate $\mathbf{h}_x$ and $\mathbf{m}$ and use a linear layer to obtain the final representation of the node. For event T_i, we obtain the feature representation of all posts $\mathbf{P}^i = \{\mathbf{p}_1^i, \mathbf{p}_2^i, ...\mathbf{p}_{M_i}^i\}$, and representation of all users $\mathbf{U}^i = \{\mathbf{u}_1^i, \mathbf{u}_2^i, ..., \mathbf{u}_{N_i}^i\}$. We discard the superscript i in the following sections for simplicity.

4.3 Chronologically-masked Transformer for Representation of Post Diffusion Tree

Many post tree modeling methods such as tree-structured RvNN (Ma et al., 2018), Bi-GCN (Bian et al., 2020b)s and DSL (Huang et al., 2019) attempt to learn the representation of post diffusion

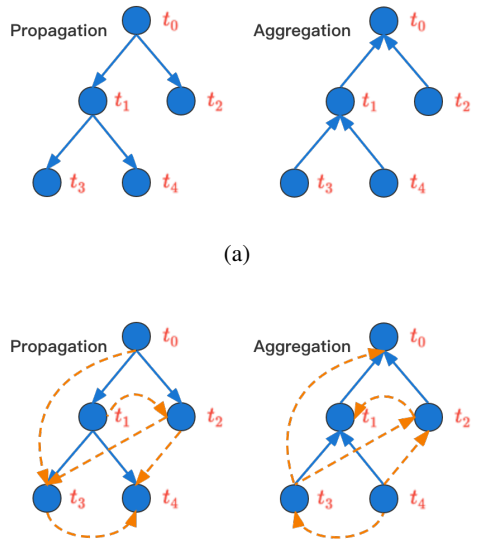


Figure 3: Illustration of diffusion trees. The blue lines denote responsive relations, and the orange lines denotes implicit relations

tree from two directions: Top-down (Propagation) and Bottom-up (Aggregation) as illustrated in Figure 3(a), to capture structural and semantic features. However, as illustrated in Figure 3(b), each user is often able to observe and respond to all existing posts at the time of writing a post in the conversation, while this lines of methods ignore the implicit interactions between unconnected posts, as well as the important temporal order. Therefore, we propose a Chronological-masked Transformer to model both temporal and structural characteristics of post diffusion. Specifically, we propose a chronologically-masked self-attention mechanism, which models the Top-down information spreading and Bottom-up aggregation separately in each layer based on the chronological order, and involves the diffusion tree structure into attention calculation via extra learnable position parameters. In the multi-head self-attention layers of standard Transformer (Vaswani et al., 2017), the state in i-th position can attend to any other position in the whole sequence, here we propose to adopt a chronologically-masking mechanism to inject the structure of both propagation and aggregation into multi-head self-attention mechanism (Vaswani et al., 2017). Specifically, As illustrated in the left bottom part of Figure 2, we first divide the heads in each self-attention layer into two groups: *propagation* heads and *aggregation* heads. For *propagation* heads, we restrict the head representation to only aggregate information from all position j with $(j \leqslant i)$ when calculating the output embedding at position i. Likewise, for *aggregation* heads, we mask the attention score from position j with $(j < i)$ for position i. The weighted sum of values at positions i for *propagation* heads and *aggregation* heads are computed as:

$$\mathbf{z}_i^p = \sum_{j=i}^{M_i} \frac{\exp e_{ij}}{\sum_{k=i}^{M_i} \exp e_{ik}} \mathbf{v}_j, \tag{6}$$

$$\mathbf{z}_i^a = \sum_{j=0}^{i} \frac{\exp e_{ij}}{\sum_{k=0}^{j} \exp e_{ik}} \mathbf{v}_j \tag{7}$$

, Furthermore, since the masking mechanism only utilizes the chronological information, in order to involve explicit spreading structure (i.e., the tree structure), we modify the calculation of attention score in Equation 3 to a structure-aware version as follows:

$$e_{ij} = \frac{\mathbf{q}_i^{\mathrm{T}} \mathbf{k}_j + \alpha_{\phi(i,j)}}{\sqrt{d_k}}, \tag{8}$$

where $\alpha_{\phi(i,j)}$ is a learnable scalar indexed by $\phi(i,j)$, and shared across all layers. $\phi(i,j)$ is the relative position between post i and post j:

$$\phi(i,j) = \begin{cases} d_i - d_j & p_i \text{ is the parent of } p_j \\ d_j - d_i + d_{\max} & p_i \text{ is the child of } p_j \\ 0 & i = j \\ 2d_{\max} & \text{in different branches} \end{cases} \tag{9}$$

, where d_i denotes the depth of post i in the spreading tree and d_{max} is the maximum depth. Through the learnable position parameters, the attention score can capture the meaningful structural information between post i and post j.

The final representation at position i before the FFN layer is the concatenation of all head presentation, denoted as:

$$\hat{\mathbf{z}}_i = \mathbf{W}^O \mathrm{Concat}(\mathbf{z}_{i,1}^p, ..., \mathbf{z}_{i,n_p}^p, \mathbf{z}_{i,1}^a, ..., \mathbf{z}_{i,n_a}^a) \tag{10}$$

, where n_p,n_a denote the number of propagation heads and aggregation heads, $\mathbf{W}^O$ is trainable parameters. Given input feature matrix of all posts $\mathbf{P}$, we obtain $\hat{\mathbf{P}} = \{\hat{\mathbf{p}}_1, \hat{\mathbf{p}}_2, ...\hat{\mathbf{p}}_M\}$ after the representation of the Chronologically-masked Transformer Network.

4.4 User Network Representation

We introduce our representation module for user social network in this section. Given the representation of all users $\mathbf{U} = \{\mathbf{u}_1, \mathbf{u}_2, ..., \mathbf{u}_N\}$ and the

adjacent matrix $\mathbf{A}^U$ of user-follow relation set E^U, we attempt to learn a structure-aware representation for each node in the following graph. Since the followers and followings describe two separate aspects of a user's characteristics, we consider neighbours of the two categories separately. Specifically, we generate the user-followed adjacent matrix $\mathbf{A}^{U\prime} = \mathbf{A}^{U\top}$. We also generate the symmetric friendship adjacent matrix $\mathbf{A}^{U\prime\prime} = \mathbf{A}^U \cdot \mathbf{A}^{U\prime}$. Given the three adjacent matrices and node features, we adopt RGCN (Schlichtkrull et al., 2018) to represent the graph. The feature update equation can be formulated as follows:

$$\mathbf{h}_i^{(t+1)} = \sigma\Big(\sum_{r\in\mathcal{R}}\sum_{j\in\mathcal{N}_i^r}\frac{1}{|\mathcal{N}_i^r|}\mathbf{W}_r^{(t)}\mathbf{h}_j^{(t)} + \mathbf{W}_{\text{root}}^{(t)}\mathbf{h}_i^{(t)}\Big) \tag{11}$$

where $\mathcal{N}_i^r$ denotes the set of neighbor indices of node i under relation $r \in R$, with corresponding adjacent matrix $\mathbf{A}^r \in \{\mathbf{A}^U, \mathbf{A}^{U\prime}, \mathbf{A}^{U\prime\prime}\}$, $\mathbf{W}_r^{(t)}$ is the parameter matrix of relation r in layer t, $\mathbf{W}_{\text{root}}^{(t)}$ is the parameter matrix of target node. After the transformation of multiple RGCN layers, we obtain the structure-aware representation of all users: $\hat{\mathbf{U}} = \{\hat{\mathbf{u}}_1, \hat{\mathbf{u}}_2, ...\hat{\mathbf{u}}_N\}$.

4.5 Post-User Fusion Network

Once we have obtain the representation of posts and users denoted as $\hat{\mathbf{P}}_i$ and $\hat{\mathbf{U}}_i$ for event T_i, we fuse the information of posts and users via a fusion network. According to user-publish-post graph G_i^{UP}, We first concatenate the hidden vectors of m-th post and n-th user, if n-th user is the author of m-th post. Note that a user can write multiple posts but a post only has one author. Therefore, we obtain the fused representation matrix $H_i = \{\mathbf{h}_1^i, \mathbf{h}_2^i, ..., \mathbf{h}_{M_i}^i\}$, where $\mathbf{h}_j^i = \text{Concat}(\hat{\mathbf{p}}_j^i, \hat{\mathbf{u}}_{u(j)}^i)$, where $u(j)$ denotes the index of user who is the author of j-th post. In order to capture the semantic relations between the fused post-user pairs, we further use a self-attention layer to obtain the final representation of all post-user pairs, denoted by $\hat{H} = \{\hat{\mathbf{h}}_1, \hat{\mathbf{h}}_2, ..., \hat{\mathbf{h}}_M\}$. Afterwards, a mean pooling layer is applied to obtain the aggregated representation $\mathbf{c}$, followed by several fully-connected layers and a Softmax layer to get the vector of probabilities for all classes. We train all the parameters in the Network by minizing the cross-entropy of the prediction and ground truth labels over the entire dataset T.

Table 2: Statistics of the datasets

Statistic	Twitter15	Twitter16	PolitiFact	GossipCop
# of posts	331,612	204,820	130872	880640
# of user	276,663	173,487	89238	568482
# of events	1490	818	574	6880
# of True rumors	374	205	\	\
# of False rumors	370	205	231	2313
# of Unverified rumors	374	203	\	\
# of Non-rumors	372	205	343	4567
Avg. # of posts / event	223	251	228	128
Max # of posts / event	1,768	2,765	3294	1038
Min # of posts / event	55	81	32	12

4.6 Data Augmentation Mechanism

Since existing datasets for rumour detection are mostly in small scale, overfitting is a serious issue in this domain. For this sake, we use two data augmentation mechanism to mitigate this problem: Connection dropping and Sub-conversation training.

- **Connection dropping:** We adopt two versions of Connection dropping operation for the user graph and post graph. For user graph, we apply the same strategy as (Bian et al., 2020a): supposing the total number of edges in the user following graph A^U is N_U and the dropping rate is p_u, then the adjacency matrix with edge dropping is $\widetilde{A}^U = A^U - A^U_{\text{drop}}$, where A^U_{drop} is the matrix constructed using $N_U \times p_u$ edges randomly sampled from A^U. The edge dropping operation is performed before input A^U into each RGCN layer, and the $A^{U\prime}, A^{U\prime\prime}$ are calculated based on $\widetilde{A}^U$. For post spreading tree, since we learn all implicit correlation between posts using self-attention, we propose to use an attention dropping mechanism, which randomly set the attention score before Softmax as $-inf$ with rate p_p.
- **Sub-conversation training:** In order to improve the robustness and early-detection capability of our model, we adopt a sub-conversation training technique. To be specific, we randomly set a time threshold t_{early}, with $t_{min} < t_{early} < t_{last}$ for each event during training, where t_{min} is the minimum detection time and t_{last} is the time of the last tweet in the event. The posts after the time is removed, so does the corresponding users. This technique enables models to learn invariant features during the whole life cycle of a event.

5 Experimental Results

In this section, we first compare the performance of our proposed PESTO method with several baseline models. Then, ablation studies are conducted to illustrate the impacts of each module. Afterwards, early detection performance is evaluated. Empirical results show the superiority of the proposed method.

5.1 Experimental Setup

We evaluate our proposed method on four publicly available Twitter datasets: Twitter15 and Twitter16 (Ma et al., 2017), PolitiFact and GossipCop (Shu et al., 2020). The statistics are listed in Table 2. Since in the original datasets, each instance only contains the tweet propagation tree, we use Twitter academic API[1] to search the corresponding user of each tweet and the following relations between users. Each source tweet is annotated with one of the four class labels, i.e., Non-rumour (N), False rumor (F), True rumor (T), and Unverified rumor (U). We compare our method with several baselines:

- DTC (Castillo et al., 2011): A Decision Tree classifier based on various handcrafted features to obtain information credibility.
- SVM-TS (Ma et al., 2017): A linear SVM classifier that utilizes handcraft features to construct time-series model.
- SVM-TK (Ma et al., 2017): A SVM classifier with a tree kernel based on the propagation structure of rumours.
- RvNN (Ma et al., 2018): A tree-structured recursive neural network with GRU units that learn the propagation structure
- PPC_RNN+CNN (Liu and Wu, 2018): A model combining RNN and CNN, which learns the rumour representations through the characteristics of users in the rumour propagation path.
- Bi-GCN (Bian et al., 2020a): A GCN-based rumour detection model using bi-directional propagation structure.
- DCM (Veyseh et al., 2019): A rumour detection model based on post-level self-attention mechanisom.
- PESTO-U: A variant of PESTO, with the user network modeling part removed.

[1]https://developer.twitter.com/en/products/twitter-api/academic-research

Table 3: Overall results on Twitter15 and Twitter16

Twitter15					
Method	ACC	N	F	T	U
DTC	0.779	0.415	0.355	0.733	0.317
SVM-TS	0.544	0.796	0.472	0.404	0.483
SVM-TK	0.750	0.804	0.698	0.765	0.733
RvNN	0.723	0.682	0.758	0.821	0.654
PPC RNN+CNN	0.477	0.359	0.507	0.300	0.640
Bi-GCN	0.886	0.891	0.860	0.930	0.864
DCM	0.770	0.814	0.764	0.775	0.743
PUFM-U	0.895	0.897	0.896	0.888	0.900
PESTO	**0.915**	**0.912**	**0.922**	**0.921**	**0.904**
Twitter16					
Method	ACC	N	F	T	U
DTC	0.473	0.254	0.080	0.190	0.482
SVM-TS	0.574	0.755	0.420	0.571	0.526
SVM-TK	0.732	0.740	0.709	0.836	0.686
RvNN	0.737	0.662	0.743	0.835	0.708
PPC RNN+CNN	0.564	0.591	0.543	0.394	0.674
Bi-GCN	0.880	0.847	0.869	0.937	0.865
DCM	0.768	0.825	0.751	0.768	0.789
PESTO-U	0.891	0.906	0.891	0.890	0.875
PESTO	**0.908**	**0.902**	**0.914**	**0.915**	**0.901**

Table 4: Overall results on PolitiFact and GossipCop

Dataset	PolitiFact		GossipCop	
Method	ACC	F1	ACC	F1
DTC	0.753	0.749	0.772	0.769
SVM-TS	0.757	0.759	0.789	0.783
SVM-TK	0.731	0.721	0.753	0.745
RvNN	0.790	0.778	0.798	0.796
PPC RNN+CNN	0.744	0.760	0.776	0.776
Bi-GCN	0.821	0.819	0.811	0.802
DCM	0.812	0.810	0.810	0.809
PUFM-U	0.832	0.821	0.821	0.816
PESTO	**0.845**	**0.836**	**0.834**	**0.831**

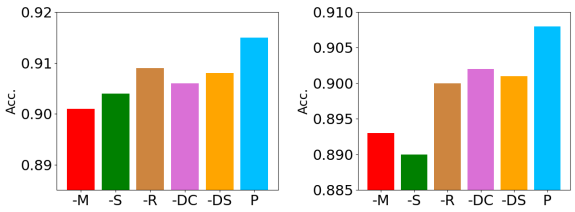
- PESTO: Our proposed PESTO, with all modules included.

5.2 Experimental Setup

In all experiments, we used the Glove 100d embeddings (Pennington et al., 2014) to represent each token in a tweet or user profile. For the chronologically-masked Transformer, the hidden size is 128, the layer number is 4, the head number is 8. For the RGCN Network, the layer number is 2, the hidden size is 128. The dropout rate of both networks is 0.2, and the edge dropping rate is also 0.2. We use the Adam optimizer with 6000 warm start-up steps. For all datasets, we evaluate the Accuracy (Acc.) over all categories and F1 measure (F_1) on each class.

5.3 Overall Performance

Table 3 shows the performance of the proposed method and all the baselines on Twitter15 and Twitter16, respectively. First, it is apparent that all the deep learning methods outperform those using handcrafted features significantly, showing that deep neural networks are able to learn better representations of rumours. Second, the proposed method and its variants outperform other deep learning methods in terms of all metrics, which indicates the superiority of PESTO. As for RvNN, it only uses the hidden feature vector of all the leaf nodes, which implies that it is heavily influenced by the information of latest posts. As for Bi-GCN, it only relies on the explicit responsive path, ignoring the implicit relations between posts. As for DCM, it simply use the self-attention layer without modification, ignoring the propagation and aggregation characteristics of rumours. PESTO-U outperforms previous methods, demonstrating the effectiveness of the proposed chronologically-masked self-attention architecture. PESTO has better performance compared with PESTO-U, indicating the user following network contains valuable information for detection.

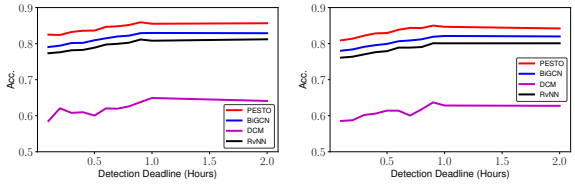


(a) Twitter15 (b) Twitter16

Figure 4: The performance of the PESTO and its variants.

5.4 Ablation study

To demonstrate the effectiveness of each module of PESTO, we conduct ablation analysis on Twitter15 and Twitter16 in this section. We compare **P**ESTO with its variants -**M**,-**S**,-**R**,-**DC**,-**DS** which represent our model (1) without chronological **M**asking for post Transformer,(2) without **S**tructure-aware attention for post Transformer,(3) with **R**GCN replaced by GCN,(4) without **C**onnection dropping and (5) **S**ub-conversation training. As illustrated in Table 4, each parts contribute to PESTO. The impacts of **M** and **S** show that involving intrinsic characteristic of the spreading tree improves the performance. RGCN is better than GCN for user network modeling, indicating that treating user-following network as directed graph retrains more valuable information. The contribution of **DC** and **DS** shows the importance of robust training.



(a) Twitter15 (b) Twitter16

Figure 5: The performance of early detection.

5.5 Early Rumour Detection

Detecting rumours at the early stage of propagation is crucial to reduce the negative effects of rumours. For the early detection task, we select a series of detection deadlines and only utilize the posts released before the deadlines and the corresponding induced user network to evaluate the performance in terms of accuracy. Figure 5 shows the performances of RvNN, Bi-GCN, DCM and our PESTO model at various deadlines on Twitter15 and Twitter16 datasets. We can find that the performance of PESTO is stably superior to other models.

6 Conclusion

In this paper, we address the task of rumour detection with social contexts. A novel Post-User Fusion Network (PESTO) is proposed to learn both post propagation patterns and user network patterns in a rumour event. To be specific, we model the post diffusion patterns using a novel chronologically-masked Transformer, and use RGCN to represent the user social network, then a fusion module based on self-attention is applied to integrate the two aspects. Experiments show that PESTO outperforms state-of-the-art baselines significantly.

References

Akiko Aizawa. 2003. An information-theoretic perspective of tf–idf measures. *Information Processing & Management*, 39(1):45–65.

Tian Bian, Xi Xiao, Tingyang Xu, Peilin Zhao, Wenbing Huang, Yu Rong, and Junzhou Huang. 2020a. Rumor detection on social media with bi-directional graph convolutional networks. In *Proceedings of*

the AAAI Conference on Artificial Intelligence, volume 34, pages 549–556.

Tian Bian, Xi Xiao, Tingyang Xu, Peilin Zhao, Wenbing Huang, Yu Rong, and Junzhou Huang. 2020b. Rumor detection on social media with bi-directional graph convolutional networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 549–556.

Carlos Castillo, Marcelo Mendoza, and Barbara Poblete. 2011. Information credibility on twitter. In *Proceedings of the 20th international conference on World wide web*, pages 675–684.

Michela Del Vicario, Alessandro Bessi, Fabiana Zollo, Fabio Petroni, Antonio Scala, Guido Caldarelli, H Eugene Stanley, and Walter Quattrociocchi. 2016. The spreading of misinformation online. *Proceedings of the National Academy of Sciences*, 113(3):554–559.

Song Feng, Ritwik Banerjee, and Yejin Choi. 2012. Syntactic stylometry for deception detection. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 171–175.

Zafar Gilani, Reza Farahbakhsh, Gareth Tyson, and Jon Crowcroft. 2019. A large-scale behavioural analysis of bots and humans on twitter. *ACM Transactions on the Web (TWEB)*, 13(1):1–23.

William L Hamilton, Rex Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. *arXiv preprint arXiv:1706.02216*.

Sooji Han, Jie Gao, and Fabio Ciravegna. 2019. Data augmentation for rumor detection using context-sensitive neural language model with large-scale credibility corpus.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Qi Huang, Chuan Zhou, Jia Wu, Mingwen Wang, and Bin Wang. 2019. Deep structure learning for rumor detection on twitter. In *2019 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE.

Zhiwei Jin, Juan Cao, Han Guo, Yongdong Zhang, Yu Wang, and Jiebo Luo. 2017. Detection and analysis of 2016 us presidential election related rumors on twitter. In *International conference on social computing, behavioral-cultural modeling and prediction and behavior representation in modeling and simulation*, pages 14–24. Springer.

Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A convolutional neural network for modelling sentences. *arXiv preprint arXiv:1404.2188*.

Ling Min Serena Khoo, Hai Leong Chieu, Zhong Qian, and Jing Jiang. 2020. Interpretable rumor detection in microblogs by attending to user interactions. *arXiv preprint arXiv:2001.10667*.

Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*.

Elena Kochkina and Maria Liakata. 2020. Estimating predictive uncertainty for rumour verification models. *arXiv preprint arXiv:2005.07174*.

Elena Kochkina, Maria Liakata, and Arkaitz Zubiaga. 2018. All-in-one: Multi-task learning for rumour verification. *arXiv preprint arXiv:1806.03713*.

Yang Liu and Yi-Fang Brook Wu. 2018. Early detection of fake news on social media through propagation path classification with recurrent and convolutional networks. In *Thirty-second AAAI conference on artificial intelligence*.

Jing Ma, Wei Gao, Prasenjit Mitra, Sejeong Kwon, Bernard J Jansen, Kam-Fai Wong, and Meeyoung Cha. 2016. Detecting rumors from microblogs with recurrent neural networks.

Jing Ma, Wei Gao, and Kam-Fai Wong. 2017. Detect rumors in microblog posts using propagation structure via kernel learning. Association for Computational Linguistics.

Jing Ma, Wei Gao, and Kam-Fai Wong. 2018. Rumor detection on twitter with tree-structured recursive neural networks. Association for Computational Linguistics.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Martin Potthast, Johannes Kiesel, Kevin Reinartz, Janek Bevendorff, and Benno Stein. 2017. A stylometric inquiry into hyperpartisan and fake news. *arXiv preprint arXiv:1702.05638*.

Victoria L Rubin and Tatiana Lukoianova. 2015. Truth and deception at the rhetorical structure level. *Journal of the Association for Information Science and Technology*, 66(5):905–917.

Michael Schlichtkrull, Thomas N Kipf, Peter Bloem, Rianne Van Den Berg, Ivan Titov, and Max Welling. 2018. Modeling relational data with graph convolutional networks. In *European semantic web conference*, pages 593–607. Springer.

Kai Shu, Deepak Mahudeswaran, Suhang Wang, Dongwon Lee, and Huan Liu. 2020. Fakenewsnet: A data repository with news content, social context, and spatiotemporal information for studying fake news on social media. *Big data*, 8(3):171–188.

Kai Shu, Suhang Wang, and Huan Liu. 2019. Beyond news contents: The role of social context for fake news detection. In *Proceedings of the twelfth ACM international conference on web search and data mining*, pages 312–320.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Petar Velivcković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2017. Graph attention networks. *arXiv preprint arXiv:1710.10903*.

Amir Pouran Ben Veyseh, My T Thai, Thien Huu Nguyen, and Dejing Dou. 2019. Rumor detection in social networks via deep contextual modeling. In *Proceedings of the 2019 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*, pages 113–120.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, R'emi Louf, Morgan Funtowicz, and Jamie Brew. 2019. Huggingface's transformers: State-of-the-art natural language processing. *ArXiv*, abs/1910.03771.

Shuo Yang, Kai Shu, Suhang Wang, Renjie Gu, Fan Wu, and Huan Liu. 2019. Unsupervised fake news detection on social media: A generative approach. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pages 5644–5651.

Chunyuan Yuan, Qianwen Ma, Wei Zhou, Jizhong Han, and Songlin Hu. 2019. Jointly embedding the local and global relations of heterogeneous graph for rumor detection. In *2019 IEEE International Conference on Data Mining (ICDM)*, pages 796–805. IEEE.

Arkaitz Zubiaga, Ahmet Aker, Kalina Bontcheva, Maria Liakata, and Rob Procter. 2018. Detection and resolution of rumours in social media: A survey. *ACM Computing Surveys (CSUR)*, 51(2):1–36.