A State-independent and Time-evolving Network for Early Rumor Detection in Social Media

Rui Xia[*], **Kaizhou Xuan**[*] **and Jianfei Yu**[†]
School of Computer Science and Engineering,
Nanjing University of Science and Technology, China
rxia@njust.edu.cn, kaizhouxuan@gmail.com, jfyu@njust.edu.cn

Abstract

In this paper, we study automatic rumor detection for in social media at the event level where an event consists of a sequence of posts organized according to the posting time. It is common that the state of an event is dynamically evolving. However, most of the existing methods to this task ignored this problem, and established a global representation based on all the posts in the event's life cycle. Such coarse-grained methods failed to capture the event's unique features in different states. To address this limitation, we propose a state-independent and time-evolving Network (STN) for rumor detection based on fine-grained event state detection and segmentation. Given an event composed of a sequence of posts, STN first predicts the corresponding sequence of states and segments the event into several state-independent sub-events. For each sub-event, STN independently trains an encoder to learn the feature representation for that sub-event and incrementally fuses the representation of the current sub-event with previous ones for rumor prediction. This framework can more accurately learn the representation of an event in the initial stage and enable early rumor detection. Experiments on two benchmark datasets show that STN can significantly improve the rumor detection accuracy in comparison with some strong baseline systems. We also design a new evaluation metric to measure the performance of early rumor detection, under which STN shows a higher advantage in comparison.

1 Introduction

Rumor is defined as an unverified statement, which may be unintentionally created or deliberately fabricated (DiFonzo and Bordia, 2007). False rumors are damaging as they may cause public panic and social unrest. Social media platforms have been ideal places for spreading rumors. It is important to automatically detect the rumors and debunk them before they are widely spread.

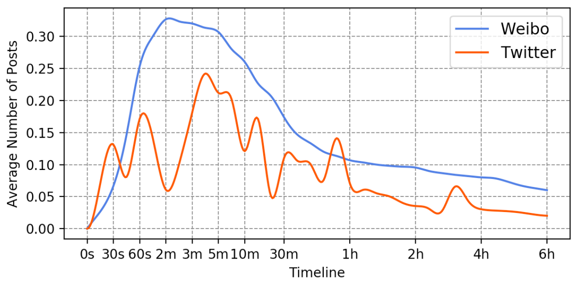

Figure 1: An illustration of the spread of rumors, which displays the average number of posts on Twitter and Weibo datasets (Ma et al., 2016) over the propagation timeline.

In recent years, the rumor detection task has attracted continuous attention from many researchers in the NLP community. We denote a statement in social media as an event consisting of a source post and its following posts such as comments or reposts (collectively called posts). Given an event, the rumor detection task is typically defined as a text classification problem (Zubiaga et al., 2018). The former aims to detect whether an event is a rumor or not.

In the literature, the typical method was to first obtain a global representation of the event based on all posts in the event's life cycle, and then employ a machine learning algorithm, such as Random Forest (RF, Kwon et al. 2013), Support Vector Machine (SVM, Ma et al. 2015), Convolution Neural Network (CNN, Yu et al. 2017) and Recurrent Neural Network (RNN, Ma et al. 2016) to learn the connection between the representation and the class labels.

On the one hand, events in social media evolve dynamically. According to communication stud-

[*] Equal contribution.
[†] Corresponding author.

ies, the dissemination of an event can be roughly divided into an evolution period, a high-tide period and an extinction period (Li et al., 2014; Han et al., 2014). As shown in Figure 1, similar curves can be observed in two real-world social media rumor datasets (i.e., Twitter and Weibo). Each state of the event has different posting density and data distribution. However, most of the aforementioned coarse-grained methods ignored the dynamics in the text data stream, and failed to capture the unique features in different states. Although part of these methods have considered temporal features or modeled the sequential dynamics with RNN, they still failed to establish fine-grained representations for different states.

On the other hand, the early detection of rumors is of great importance. According to our observation on the two rumor datasets, most events reach the high-tide period in less than five minutes. Although some of the previous work segmented the timeline by equal time span or equal number of posts for early rumor detection (Ma et al., 2016; Guo et al., 2018; Chen et al., 2018), they potentially ignored the vital features of early states and failed to train targeted models for early detection.

To address the limitations mentioned above, we propose a new State-independent and Time-evolving Network (STN) for rumor detection based on propagation state detection and segmentation, and apply it to early rumor detection in this paper. Specifically, since an event in social media is actually a sequence of posts sorted according to the posting time, it can be viewed as a time-series text data stream. To learn the propagation states in the text data stream, we first employ the Kleinberg algorithm (Kleinberg, 2003) to segment an event into several sub-events based on the state transition, each of which represents a continuous and identical state. Subsequently, we train an encoder to fit each sub-event separately. We furthermore propose a time-evolving fusion (He et al., 2018) mechanism to merge the current sub-event representation with previous ones, and combine them together for incremental prediction. STN no longer outputs one predictive label for one event, but outputs a sequence of labels for each state-independent sub-event, which enables early detection of rumors. Moreover, we further present a new evaluation metric, called Time-series Smoothing Accuracy (TS-Acc), for measuring the performance of early rumor detection.

Experimental results on two real-world rumor detection datasets released by Ma et al. (2016) demonstrate the effectiveness of our STN model. It not only achieves significant improvements for rumor detection in comparison with several strong baseline systems, but also greatly improves the early rumor detection performance.

2 Related Work

In recent years, rumor classification system has developed rapidly. Based on the definition in (Zubiaga et al., 2018), a complete rumor classification system consists of four components: i. rumor detection; ii. rumor tracking; iii. stance classification; iv. rumor verification. Among the four sub-tasks, rumor verification resembles rumor detection closely. For rumor detection, the goal is to detect whether a statement is a rumor or not (i.e, the class labels are rumor and non-rumor); for rumor verification, the goal is to determine whether a rumor is true, false or unconfirmed. Some following work have also combined the class labels together and consider it as a four-class classification problem (non rumor, true rumor, false rumor, unverified rumor) (Ma et al., 2017)[1].

During the prophase study of rumor detection, researchers focused on extracting various obvious features of microblog events on social media platforms, and combined the features with traditional machine learning classifiers to detect rumors or identify information credibility (Castillo et al., 2011; Yang et al., 2012; Kwon et al., 2013; Liu et al., 2015; Ma et al., 2015; Wu et al., 2015; Zhao et al., 2015; Wang and Terano, 2015; Vosoughi, 2015). These manually-designed features can be roughly categorized into three groups, including text content, user portraits and propagation states. However, it is hard for these traditional approaches to capture the dynamic characteristics during the spread of an event and the relationship between the posts.

To address this issue, Kwon et al. (2013) constructed a massage propagation model to find the diversity of the amount of related posts between rumor and nor-rumor. Ma et al. (2015) first proposed to divide the event timeline into equal-span periods and utilized the dynamic changes of fea-

[1]It should be noted our proposed framework is compatible with both rumor detection and rumor verification, although they have different space of class labels. Therefore, we make no distinction between the two and use the terminology of "rumor detection" instead of both, for simplicity.

tures in adjacent periods. Based on this, Ma et al. (2016) further introduced RNN models to encode the time periods, which verified the effectiveness of RNN models on encoding sequential posts. Zubiaga et al. (2017) utilized a sequential approach based on Linear-chain Conditional Random Fields (CRF) to learn the dynamic relations between posts, which relies on the content of a source microblog and its related posts. Kwon et al. (2017) employed different sets of features to keep the properties of the propagation structure and temporal relations among posts. Moreover, Guo et al. (2018) incorporated the attention mechanism into stacked RNNs to model the temporal propagation of an event.

In addition, there is another line of researches focusing on modeling the post sequences with tree structures, which aims to useful relations among the responsive posts (Nadamoto et al., 2013; Wu et al., 2015; Ma et al., 2017, 2018; Kumar and Carley, 2019). Among them, the representative studies are Ma et al. (2018) and Kumar and Carley (2019), which respectively proposed a recursive neural network and a Tree-LSTM architecture to explicitly model the tree structure. Different from all the studies mentioned above, a recent study by Ma et al. (2019) proposed to leverage Generative Adversarial Networks (GAN) to improve the robustness of rumor detection, where a generative model is trained to confuse the rumor detection discriminator by generating pseudo real examples.

Although much work has been done for rumor detection, only a few previous studies focused on the early detection of rumors (EDR). Zhao et al. (2015) argued that rumors are more likely to arouse users' suspicion, and proposed to aggregate related posts with specific phrases, followed by performing EDR with cluster-based classifiers. However, this work inevitably involved much human effort. To alleviate the reliance of feature engineering, Nguyen et al. (2017) utilized deep neural networks to automatically capture features at the post level. Although it achieves better early detection performance, it is difficult to be applied to large events. Liu and Wu (2018) believed that early posts are easy to be manipulated by the source microblog, while user characteristics are relatively stable. They integrated RNN and CNN models to capture user characteristics in the propagation process of an event. However, only using user features makes their model unable to achieve continuous performance improvement as time goes by. More recently, Song et al. (2019) introduced the concept of credible detection points, and proposed to gather every ten posts along the timeline as one time-step of RNN and made prediction at each step. But tens of thousands of posts make the number of time steps large, which may reduce the reliability of long-distance dependence.

3 Approach

3.1 Task Definition

Suppose $\mathcal{D} = \{(\mathbb{E}^{(1)}, y^{(1)}), \ldots, (\mathbb{E}^{(|\mathcal{D}|)}, y^{(|\mathcal{D}|)})\}$ is a rumor detection dataset, where $\mathbb{E}$ denotes one event, and y denotes its class label. Each event $\mathbb{E}$ consists of a large amount of posts:

$$\mathbb{E} = \{\mathbf{c}_0, \mathbf{c}_1, \ldots, \mathbf{c}_{|\mathbb{E}|}\}, \tag{1}$$

where $|\mathbb{E}|$ is the number of posts in it. The first post $\mathbf{c_0}$ in $\mathbb{E}$ is regarded as the source post published at time t_0. Each of the following posts $\mathbf{c}_i$ has an arrival time t_i and $\mathbf{c}_i$ denotes the feature representation of each post. After sorting all the posts in event $\mathbb{E}$ according to the arrival time, $\mathbb{E}$ can be considered as a time-series text data steam $\mathbb{E} = [(\mathbf{c}_0, t_0), (\mathbf{c}_1, t_1), \ldots, (\mathbf{c}_{|\mathbb{E}|}, t_{|\mathbb{E}|})]$.

We train a rumor detection model based on $\mathcal{D}$, and use it to predict the class labels y on an unseen event $\mathbb{E}$.

3.2 Event State Detection and Segmentation

The Kleinberg algorithm (Kleinberg, 2003) was originally used to detect burst incidents on news or e-mails. In this paper, we employ it to detect the state for each post in an event. Based on the hidden Markov model, the Kleinberg algorithm can identify the hidden state sequence corresponding to a post sequence.

For an event consisting of multiple posts $\mathbb{E} = [(\mathbf{c}_0, t_0), (\mathbf{c}_1, t_1), \ldots, (\mathbf{c}_{|\mathbb{E}|}, t_{|\mathbb{E}|})]$, we first build a sequence of arrival time intervals

$$X = [x_1, x_2, \ldots, x_{|\mathbb{E}|}], \tag{2}$$

where

$$x_i = t_i - t_{i-1}, i = 1, 2, \ldots, |\mathbb{E}|. \tag{3}$$

Our goal is to obtain the corresponding state sequence Q for the interval sequence X:

$$Q = [q_1, q_2, \ldots, q_{|\mathbb{E}|}], \tag{4}$$

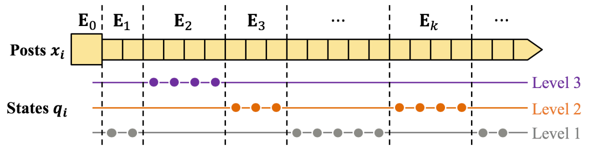

Figure 2: An example of the state sequence given the post sequence where the number of state levels $N = 3$. Each post is assigned with a hidden state $q_i \in \{1, 2, 3\}$. For example, $q_1 = 1$, $q_2 = 1$ and $q_3 = 3$.

where $q_i \in \{1, 2, \ldots, N\}$ denotes the state of x_i, and N number of state levels in Q. Figure 2 illustrates a case of the state changes of part of the posts in an event when $N = 3$.

The Kleinberg algorithm assumes the arrival time interval has a memoryless exponential distribution:

$$p(x_i|q_i = j) = \alpha_j e^{-\alpha_j x_i}, \tag{5}$$

where α_j can be regarded as the arrival rate of posts. It can be derived that the expected value of x_i is α_j^{-1}.

For the basic state $q = 1$, we set its arrival rate as the reciprocal of the average intervals of all posts in $\mathbb{E}$:

$$\alpha_1 = \frac{|\mathbb{E}|}{t_{|\mathbb{E}|} - t_0}. \tag{6}$$

The values of α_j corresponding to higher states $q_i = j$ are then set as:

$$\alpha_j = s^{(j-1)} \cdot \alpha_1, \tag{7}$$

where $s > 1$ is a preset scaling parameter.

For adjacent arrival time intervals x_i and x_{i+1} with the corresponding states $q_i = a$ and $q_{i+1} = b$, the loss of transition from state a to b is defined as:

$$\tau(a, b) = \begin{cases} (b - a)\gamma \ln n, b > a \\ 0, b \leq a \end{cases}, \tag{8}$$

where γ is a preset parameter to control the magnitude of transition loss.

The objective of the algorithm is to solve a state sequence Q, which minimizes the cost function $L(Q|X)$:

$$L(Q|X) = \sum_{i=1}^{|\mathbb{E}|-1} \tau(q_i, q_{i+1}) - \sum_{i=1}^{|\mathbb{E}|} \ln p(x_i|q_i). \tag{9}$$

where the first item is the loss of state transition, based on which we expect the frequency of transition to be as small as possible. The second term is the log-likelihood, based on which we want to maximize the density functions $p(X|Q)$ given the sequence of x_i and q_i pairs.

After obtaining the optimal state sequence Q, we then merge the continuous posts with the same state into a single sub-event, and finally represents an event $\mathbb{E}$ by a sequence of K state-independent sub-events:

$$\mathbb{E} = [\mathbf{E}_0, \mathbf{E}_1, \ldots, \mathbf{E}_{K-1}]. \tag{10}$$

Each sub-event $\mathbf{E}_k$ includes a series of continuous posts:

$$\mathbf{E}_k = [c_{k,1}, c_{k,2}, \ldots, c_{k,|\mathbf{E}_k|}]. \tag{11}$$

where $c_{k,l}$ denotes the l-th post in the k-th sub-event.

3.3 State-independent Sub-event Encoder

For each sub-event $\mathbf{E}_k$, we train a state-independent sub-event encoder e_k to get the sub-event representation.

Firstly, the mean pooling of the embedding of all words in a post is used as the post representation:

$$\mathbf{c}_i = \text{mean}(\mathbf{w}_1^{(c_i)}, \mathbf{w}_2^{(c_i)}, \ldots, \mathbf{w}_{|c_i|}^{(c_i)}), \tag{12}$$

where $\mathbf{w}_l^{(\mathbf{c}_i)}$ is the word embedding vector retrieved from a pre-trained word embedding matrix, and $\mathbf{c}_i$ denotes the representation of the i-th post. Based on $\mathbf{c}_i$, we can then get the input representation of the sub-event $\mathbf{E}_k$, denoted by $\mathbf{X}_k = [\mathbf{c}_1, \mathbf{c}_2, \ldots, \mathbf{c}_{|\mathbf{E}_k|}]$.

Secondly, we employ a basic encoder (e.g., CNN, LSTM) to get the sub-event representation $\mathbf{h}_k$ based on the input post representation $\mathbf{X}_k$. The encoding of the sub-event $\mathbf{E}_k$ can finally be expressed as follows:

$$\mathbf{h}_k = \text{Encoder}(\mathbf{X}_k). \tag{13}$$

Note that here the state-independent sub-event encoder is a general framework compatible with the widely used encoders of texts, e.g., CNN, LSTM, GRU, etc. In the experiments, in addition to CNN, we also report the results based on LSTM and GRU.

3.4 Time-evolving Representation and Classification

The social media event evolve dynamically, and the representations of pre-ordered sub-events may be helpful for current sub-event prediction. Therefore,

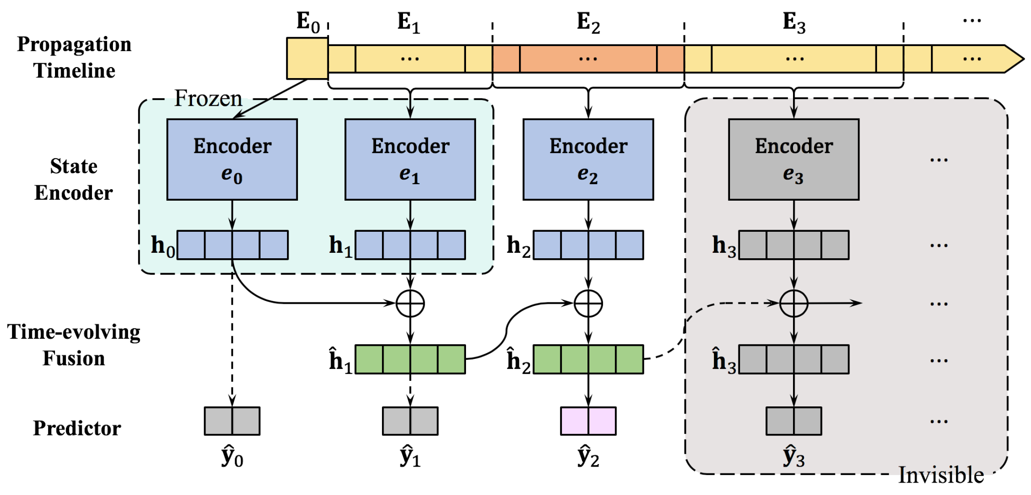

Figure 3: A schematic diagram of our model when learning for sub-event $\mathbf{E}_2$. The model outputs the prediction result $\hat{\mathbf{h}}_2$ and updates all the visible and unfrozen modules accordingly.

we add a time-evolving fusion module after each sub-event encoder to fuse the representation of the current sub-event with previous ones:

$$\begin{aligned}\hat{\mathbf{h}}_k &= \delta(\mathbf{W}_k(\mathbf{h}_k \oplus \hat{\mathbf{h}}_{k-1})) \\ &= \delta\big(\mathbf{W}_k\big(\mathbf{h}_k \oplus \delta(\mathbf{W}_{k-1}(\mathbf{h}_{k-1} \oplus \hat{\mathbf{h}}_{k-2}))\big)\big) \\ &= \ldots,\end{aligned} \tag{14}$$

where δ is a Sigmoid activation function and $\mathbf{W}_k$ is a weight matrix. Similarly, $\hat{\mathbf{h}}_k$ will be used to guide the encoding $\hat{\mathbf{h}}_{k+1}$ of next sub-event, forming a recursive encoding mode.

We independently predict the authenticity of sub-event $\mathbf{E}_k$ under each state. The encoding $\hat{\mathbf{h}}_k$ of $\mathbf{E}_k$ will be fed into a separate softmax classifier to get the prediction result $\hat{\mathbf{y}}_k$:

$$\hat{\mathbf{y}}_k = \mathrm{softmax}(\mathbf{V}_k \hat{\mathbf{h}}_k + \mathbf{b}_k), \tag{15}$$

where $\mathbf{V}_k$ and $\mathbf{b}_k$ are parameters representing weights and bias.

Given the sequence of sub-events $\mathbb{E} = [\mathbf{E}_0, \mathbf{E}_1, \ldots, \mathbf{E}_{K-1}]$, our model incrementally outputs a corresponding sequence of predictive probabilities:

$$Y = [\hat{y}_0, \hat{y}_1, \ldots, \hat{y}_{K-1}]. \tag{16}$$

The training objective of each sub-event is to minimize the cross-entropy loss between the predictive probability $\hat{\mathbf{y}}_k$ and the true class label $\mathbf{y}$:

$$L_{\mathbf{E}_k} = \mathbf{y} \cdot \log \hat{\mathbf{y}}_{\mathbf{k}}, \tag{17}$$

where $L_{\mathbf{E}_k}$ denotes the loss for sub-event $\mathbf{E}_k$.

It should be noted that our model is learned in an incremental training mechanism. In training for the current sub-event $\mathbf{E}_k$, the parameters of all previous encoders are frozen. That is, we only update the parameters in current encoder. But the fusion parameters (i.e., $\mathbf{W}_{k-1}, \mathbf{W}_{k-2}, \ldots$) should be fine-tuned synchronously. The incremental training process is illustrated in Figure 3.

3.5 Early Detection of Rumors

In this subsection, we further propose a new evaluation metric, named **Time-series Smoothing Accuracy (TS-Acc)**, to measure the performance of early rumor detection.

Since the earlier prediction results are more important for rumor detection, we first employ a smoothed exponential function to assign a weight to the accuracy for the predictions in each sub-event:

$$v(t) = e^{-\log_{\lambda}(t+\lambda)}, \tag{18}$$

where t is the arrival time of sub-events and and λ is a smoothing parameter defined as 60 in our experiments.

TS-Acc is then defined as a weighed sum of accuracies of already-appeared sub-events:

$$\text{TS-Acc} = \sum_{k} Acc^{(k)} \cdot \mathrm{Norm}(v(t^{(k)})). \tag{19}$$

where $\mathrm{Norm}(v(t^{(k)}))$ denotes the normalized weight among k already-appeared sub-events.

It is also worth noting that in case of discrete time-points, the area under accuracy-time curve

Table 1: Statistics of Twitter and Weibo datasets.

	Twitter	Weibo
Events	992	4,664
Non-rumor events	498	2,313
Rumor events	494	2,351
Posts	949,224	3,805,656
Avg. post number/event	957	816
Avg. time length/event	360.8Hours	1,808.7Hours

is equivalent to the sum of accuracies at all time-points. Since TS-Acc is a weighted sum of accuracies, it can also be regarded as a weighted version of area under accuracy-time curve.

4 Experiment

4.1 Datasets and Experimental Settings

Twitter and Weibo datasets were published by Ma et al. (2016), both of which provided a large number of relevant posts for each microblog event. Due to the protection policy of Twitter, we re-crawl all the posts in the Twitter dataset according to their ID numbers. However, since some of the tweets are no longer available, we discard those unresponsive source microblogs and finally obtain 90% of the original dataset for our experiments. For Weibo dataset, the events of misinformation are marked as rumor. According to the definition in (Zubiaga et al., 2018), it is more related to true rumor. But to be be consistent with (Ma et al., 2016), we still regard the event category as rumor and non-rumor.

The detailed statistics of both datasets are shown in Table 1. Following the same settings in the previous papers, we hold out 10% of the events in both datasets for model tuning, and the rest of the events are split with a ratio of 3:1 for training and test. To guarantee obtaining global states for different events, we have not performed the Kleinberg algorithm for each post. Instead, we combine all events in the dataset together and align the posts in them according to the posting time. The Kleinberg algorithm is then performed on the combined entire dataset. We use the Chinese word embeddings from Tencent AI Lab (Song et al., 2018) and the English word embeddings from Google News. When training the model, we use the Adam optimizer (Kingma and Ba, 2014).

4.2 Rumor Detection Performance

In this subsection, we compare our proposed STN model with the following rumor detection methods on the standard rumor detection task, i.e., evaluating the detection accuracy after the end of the event propagation:

- **DTR**: A ranking model based on a decision tree to identify trending rumors through searching for disputed claims (Zhao et al., 2015);
- **SVM**: A linear SVM model to identify rumors with the handcrafted features and feature change gradient (Ma et al., 2015);
- **GRU**: A GRU model with the text data extracted from the variable-length time series as the input (Ma et al., 2016);
- **PPC**: A time series classifier based on RNN and CNN, which captures the user characteristics along the propagation path (Liu and Wu, 2018);
- **AIM**: An attention-based classification model which can extract valid content and temporal features (Liu et al., 2018);
- **CED**: A continuous detection model which first obtains credible detection points for each repost sequence, followed by making reliable prediction based on the information before the credible detection point (Song et al., 2019).

Based on the results reported in Table 2, we can make a couple of observations. First, compared with the traditional model DTR and SVM, GRU achieves obvious improvements on both datasets, and AIM shows even better performance by using attention mechanism. Second, based on the credible detection point, CED further boosts the detection accuracy on Weibo to 94.6%. Finally, our model STN consistently achieves the best performance on both Twitter and Weibo datasets, which outperforms the state-of-the-art models by around two percentage points on both detection accuracy and $\mathbf{F_1}$ score.

4.3 Early Detection Performance

In this subsection, we compare the performance of all the models in early detection of rumors (EDR), i.e., predicting the credibility of microblog events based on the posts released before a detection time point.

(1) The curve of detection accuracy

In Figure 4, we show the detection accuracy of all the models as the time goes by. In particular, we illustrate more detection results within the first 6 hours.

First, we can see from Figure 4 that the accuracy of DTR and that of SVM grow slowly on both

Table 2: Results of conventional rumor detection on Twitter and Weibo datasets. Results are evaluated by accuracy, macro-precision, recall, and F_1-score. Part of the data are excerpted from published papers. (#CED only uses half of the Twitter dataset.)

Method	Twitter				Weibo			
	Accuracy	Precision	Recall	F_1	Accuracy	Precision	Recall	F_1
DTR	0.681	0.679	0.680	0.679	0.732	0.732	0.732	0.732
SVM	0.745	0.758	0.741	0.749	0.857	0.859	0.858	0.858
GRU	0.757	0.760	0.757	0.758	0.910	0.914	0.910	0.912
PPC	-	-	-	-	0.921	0.923	0.926	0.924
AIM	0.796	0.799	0.800	0.799	0.936	0.936	0.937	0.936
CED	0.744#	0.708#	0.791#	0.747#	0.946	0.946	0.944	0.945
Ours	**0.821**	**0.824**	**0.823**	**0.824**	**0.963**	**0.963**	**0.963**	**0.963**

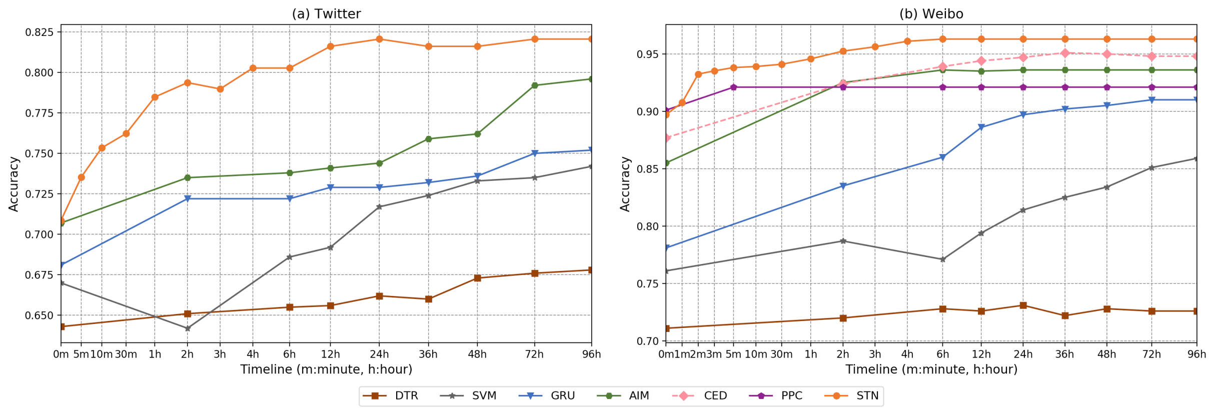

Figure 4: Early detection results on (a) Twitter and (b) Weibo. More time points within 6 hours are shown. (We reproduced the curve of CED, but it may be slightly different from the original.)

Twitter and Weibo datasets. In contrast, the GRU model has a faster and more stable rising curve on both datasets. Second, compared with the previous methods, we can find that AIM consistently improves the detection accuracy at each detection time point. Moreover, PPC can quickly improve the detection accuracy to over 92% in the first 5 minutes, but it cannot continue to perform better, whereas CED can continuously improve its detection accuracy to over 94% within the first 6 hours on the Weibo dataset. Finally, in comparison with all the methods mentioned above, STN shows a significant improvement within the first 6 hours. Specifically, it is easy to see that STN has achieved over 75% and 94% accuracy respectively at the 10th minute. In addition, as the time goes by, we can clearly see that STN can gradually improve its detection accuracy, and outperform all the state-of-the-art models at each detection time point.

(2) Time-series Smoothing Accuracy

As introduced in Section 3.5, we propose the Time-series Smoothing Accuracy (TS-Acc) to evaluate the efficiency of EDR. In Table 3, we report the results of using this evaluation metric to compare all the models. To be consistent with Figure 4, we respectively select 3 and 9 time points within the first 6 and 96 hours to re-evaluate the TS-Acc performance of all the models.

First, we can see that for each approach, the overall trend of the TS-Acc performance is similar to that of the accuracy performance in Figure 4. Second, it is worth noting that for the Weibo dataset, the TS-Acc of PPC in 96 hours is slightly lower than AIM and CED, whereas its TS-Acc in 6 hours is significantly higher than AIM and CED. This indicates that our evaluation metric TS-Acc primarily reflects the speed of improvement in the early stage. Finally, we can clearly observe that the TS-Acc of STN is significantly higher than that of all the state-of-the-art models on both datasets, which is consistent with the performance trend shown in Figure 4.

4.4 Discussion on State Detection and Event Segmentation

(1) Discussion on the Dynamics of Data Distribution

To show the state detection and event segmentation advantages of Kleinberg algorithm, we com-

Table 3: Evaluation results of early detection efficiency with our TS-Acc metric.

Model	6-hours' TS-Acc		96-hours' TS-Acc	
	Twitter	Weibo	Twitter	Weibo
DTR	0.6487	0.7183	0.6576	0.7225
SVM	0.6652	0.7719	0.6936	0.7994
GRU	0.7048	0.8189	0.7210	0.8603
PPC	-	0.9126	-	0.9169
AIM	0.7240	0.8985	0.7435	0.9175
CED	-	0.9082	-	0.9283
STN	**0.7603**	**0.9366**	**0.7897**	**0.9477**

Table 4: Mean values of intra-class distance on Weibo datasets when using different event segmentation methods.

Method	Sub-events/event	Intra-class distance
VTS	30	1.0570
VTS	50	1.2641
CPT	2000	0.9302
Kleinberg	33	**0.8177**

Table 5: Effects of different encoders of STN. (The number of detection points for TS-Acc are 12 and 33 respectively.)

s	1.5	1.2	1.1	1.1
γ	1	1	1	0.8
Max state level	5	10	19	19
State change times	247	89	84	104
Number of final sub-events	39	33	32	33

Table 6: Effects of different encoders of STN. Results are evaluated by regular accuracy and Time-series Smoothing Accuracy. (The number of detection points for TS-Acc are 3 and 9 respectively.)

Encoder	in 6-hours		in 96-hours	
	Accuracy	TS-Acc	Accuracy	TS-Acc
LR	0.9295	0.9149	0.9323	0.9197
LSTM	0.9476	0.9317	0.9486	0.9365
GRU	0.9505	0.9346	0.9514	0.9395
GRU-ATT	0.9524	0.9331	0.9533	0.9390
CNN	**0.9629**	**0.9366**	**0.9629**	**0.9477**

pare it with some representative state segmentation methods, such as Variable-length Time Series (VTS, Ma et al. 2016) and Constructing Post Series (CPS, Chen et al. 2018), which divide the event with equal time span and equal number of posts respectively.

Specifically, we calculate the intra-class distance of each divided sub-event, and obtain the mean value of all the distances of sub-events for each model. Note that the smaller the intra-distance is, the closer the post features in the sub-event are. In Table 4, it is easy to observe that Kleinberg algorithm can obtain the lowest intra-class distance, which demonstrates its better state segmentation ability, and it may reduce the dynamics of data distribution of sub-events and further enhance the feature extraction ability of our encoders.

(2) Effects of Parameters of Kleinberg

Kleinberg algorithm has two important preset parameters s and γ which are used to set the expected arrival rate of posts and the transfer loss between different state levels. As shown in Table 5, we explore the impact of several pairs of s and γ on the sub-event partition.

In order to ensure effective training of STN, we adjust the state division of the Kleinberg algorithm. If the number of events that have posts under a single state is less than 30% of the total number of events, we merge the current state with the sequential state, and if the duration of a single state exceeds two hours, we truncate it at the 2nd hour. Finally, we find that the reasonable changes of s and γ have little effect on the number and the segmented position of sub-events. Thus, we draw the conclusion that the performance of STN is not subject to the parameter adjustment of Kleinberg.

4.5 Discussion on the Compatibility of State-independent Encoders

As mentioned above, the state encoder of STN is a general framework compatible with traditional feature extraction and classification algorithms or deep neural networks. We do experiments with the following encoders on Weibo dataset: **LR** (Logistic Regression), **LSTM**, **GRU**, **GRU-ATT** (GRU with self-attention) and **CNN**.

Table 6 shows the detection accuracy and early detection efficiency TS-Acc of STN in the first 6 and 96 hours with different encoders. First, we can see that the traditional machine learning model LR can already achieve good performance. Second, among all the deep learning encoder, CNN obtains the best performance in both settings. Moreover, by comparing the results in Table 3 and Table 6, we can see that all the deep learning encoders can obtain better performance than all the state-of-the-art models, which indicates the effectiveness and the generalization ability of our STN model.

4.6 Discussion on the Incremental Training

Finally, to verify the effect of the time-evolving fusion module, we replace the module of STN with a standard GRU, and make STN degenerate into an integrated training model (GRU with Kleinberg, GK).

Table 7: Comparison between GK and other models. Results are evaluated by accuracy, macro-precision, recall, and $\mathbf{F_1}$-score.

Model	Accuracy	Precision	Recall	$\mathbf{F_1}$
GRU	0.910	0.914	0.910	0.912
GK	0.949	0.948	0.949	0.948
STN	**0.963**	**0.963**	**0.963**	**0.963**

As shown in Table 7, we can easily find that our incremental training method (i.e., STN) can consistently perform better than GRU and GK, which demonstrates the usefulness of our time-evolving fusion module. Moreover, in our experiments, we also find that since the prediction of the current state is dependent on the previous state in our time-evolving fusion module, events that are predicted correctly in the earlier states rarely change in the follow-up state, but most of the events that are wrongly predicted in the earlier states can be largely corrected in the follow-up state. This further proves the effectiveness of our STN model.

5 Conclusion and Future Work

In this paper, we first introduce the Kleinberg algorithm to identify the propagation states for an event composed of a sequence of posts and segment the sequence into several state-independent sub-events. On this basis, we propose a state-independent and time-evolving network (STN) for rumor detection as well as early rumor detection. We also present a new metric called time-series smoothing accuracy (TS-Acc) for measuring the efficiency of early rumor detection. The experimental results on two real-world microblog rumor datasets demonstrates the advantages of our STN approach in terms of both rumor detection accuracy and our proposed TS-Acc metric, in comparison with some strong rumor detection systems.

One disadvantage of this work is that the Kleinberg algorithm is performed on the combination of all events in the dataset to maintain global states. This way may fail to capture the individual state transition in single events. Secondly, it is a retrospective algorithm which depends on the condition all posts along the timeline should be provided in advance. Therefore, one direction for future work is to explore an online state detection algorithm and perform it for each event, but at the same time ensure that the state of each event is globally defined. It would be even better if the state detection and segmentation step can be integrated with subsequent state-independent feature extraction and rumor detection in an end-to-end framework.

Acknowledgments

We would like to thank the anonymous reviewers for their insightful comments and helpful suggestions. This work was supported by the Natural Science Foundation of China (No. 61672288, 62076133, and 62006117), and the Natural Science Foundation of Jiangsu Province for Young Scholars (SBK2020040749) and Distinguished Young Scholars (SBK2020010154).

References

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

Tong Chen, Xue Li, Hongzhi Yin, and Jun Zhang. 2018. Call attention to rumors: Deep attention based recurrent neural networks for early rumor detection. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD)*, pages 40–52.

Nicholas DiFonzo and Prashant Bordia. 2007. *Rumor psychology: Social and organizational approaches.* American Psychological Association.

Han Guo, Juan Cao, Yazi Zhang, Junbo Guo, and Jintao Li. 2018. Rumor detection with hierarchical social attention network. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management (CIKM)*, pages 943–951.

Shuo Han, Fuzhen Zhuang, Qing He, Zhongzhi Shi, and Xiang Ao. 2014. Energy model for rumor propagation on social networks. *Physica A: Statistical Mechanics and its Applications*, 394:99–109.

Yu He, Jianxin Li, Yangqiu Song, Mutian He, and Hao Peng. 2018. Time-evolving text classification with deep neural networks. In *International Joint Conferences on Artificial Intelligence (IJCAI)*, pages 2241–2247.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980.*

Jon Kleinberg. 2003. Bursty and hierarchical structure in streams. *Data Mining and Knowledge Discovery*, 7(4):373–397.

Sumeet Kumar and Kathleen Carley. 2019. Tree LSTMs with convolution units to predict stance and rumor veracity in social media conversations. In *Proceedings of ACL*.

Sejeong Kwon, Meeyoung Cha, and Kyomin Jung. 2017. Rumor detection over varying time windows. *PloS one*, 12(1):e0168344.

Sejeong Kwon, Meeyoung Cha, Kyomin Jung, Wei Chen, and Yajun Wang. 2013. Prominent features of rumor propagation in online social media. In *2013 IEEE 13th International Conference on Data Mining (ICDM)*, pages 1103–1108.

Mingde Li, Shengjun Meng, and Hongbang Zhang. 2014. Communication mode of microblog: a study based on the process analysis. *Journal of Intelligence*, (2):23.

Qiang Liu, Feng Yu, Shu Wu, and Liang Wang. 2018. Mining significant microblogs for misinformation identification: An attention-based approach. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 9(5):50.

Xiaomo Liu, Armineh Nourbakhsh, Quanzhi Li, Rui Fang, and Sameena Shah. 2015. Real-time rumor debunking on twitter. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management (CIKM)*, pages 1867–1870.

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 Association for the Advancement of Artificial Intelligence (AAAI)*.

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. In *International Joint Conferences on Artificial Intelligence (IJCAI)*, pages 3818–3824.

Jing Ma, Wei Gao, Zhongyu Wei, Yueming Lu, and Kam-Fai Wong. 2015. Detect rumors using time series of social context information on microblogging websites. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management (CIKM)*, pages 1751–1754.

Jing Ma, Wei Gao, and Kam-Fai Wong. 2017. Detect rumors in microblog posts using propagation structure via kernel learning. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL)*, volume 1, pages 708–717.

Jing Ma, Wei Gao, and Kam-Fai Wong. 2018. Rumor detection on twitter with tree-structured recursive neural networks. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (ACL)*, volume 1, pages 1980–1989.

Jing Ma, Wei Gao, and Kam-Fai Wong. 2019. Detect rumors on twitter by promoting information campaigns with generative adversarial learning. In *Proceedings of the 28th International Conference on World Wide Web (WWW)*, pages 3049–3055.

Akiyo Nadamoto, Mai Miyabe, and Eiji Aramaki. 2013. Analysis of microblog rumors and correction texts for disaster situations. In *Proceedings of International Conference on Information Integration and Web-based Applications & Services*, page 44.

Tu Ngoc Nguyen, Cheng Li, and Claudia Niederée. 2017. On early-stage debunking rumors on twitter: Leveraging the wisdom of weak learners. In *International Conference on Social Informatics*, pages 141–158.

Changhe Song, Cheng Yang, Huimin Chen, Cunchao Tu, Zhiyuan Liu, and Maosong Sun. 2019. Ced: credible early detection of social media rumors. *IEEE Transactions on Knowledge and Data Engineering*.

Yan Song, Shuming Shi, Jing Li, and Haisong Zhang. 2018. Directional skip-gram: Explicitly distinguishing left and right context for word embeddings. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*, pages 175–180.

Soroush Vosoughi. 2015. *Automatic detection and verification of rumors on Twitter*. Ph.D. thesis, Massachusetts Institute of Technology.

Shihan Wang and Takao Terano. 2015. Detecting rumor patterns in streaming social media. In *2015 IEEE International Conference on Big Data (Big Data)*, pages 2709–2715. IEEE.

Ke Wu, Song Yang, and Kenny Q Zhu. 2015. False rumors detection on sina weibo by propagation structures. In *2015 IEEE 31st International Conference on Data Engineering*, pages 651–662.

Fan Yang, Yang Liu, Xiaohui Yu, and Min Yang. 2012. Automatic detection of rumor on sina weibo. In *Proceedings of the ACM SIGKDD Workshop on Mining Data Semantics*, page 13.

Feng Yu, Qiang Liu, Shu Wu, Liang Wang, Tieniu Tan, et al. 2017. A convolutional approach for misinformation identification. In *International Joint Conferences on Artificial Intelligence (IJCAI)*, pages 3901–3907.

Zhe Zhao, Paul Resnick, and Qiaozhu Mei. 2015. Enquiring minds: Early detection of rumors in social media from enquiry posts. In *Proceedings of the 24th International Conference on World Wide Web (WWW)*, pages 1395–1405.

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):32.

Arkaitz Zubiaga, Maria Liakata, and Rob Procter. 2017. Exploiting context for rumour detection in social media. In *International Conference on Social Informatics*, pages 109–123.