Hindawi
Security and Communication Networks
Volume 2021, Article ID 5569064, 13 pages
https://doi.org/10.1155/2021/5569064

*Research Article*

# Early Rumor Detection Based on Deep Recurrent Q-Learning

**Wei Wang, Yuchen Qiu, Shichang Xuan, and Wu Yang**

*Information Security Research Center, College of Computer Science and Technology, Harbin Engineering University, Harbin 150001, China*

Correspondence should be addressed to Wu Yang; yangwu@hrbeu.edu.cn

Received 1 March 2021; Accepted 19 May 2021; Published 1 June 2021

Academic Editor: Lu Liu



Online social networks provide convenient conditions for the spread of rumors, and false rumors bring great harm to social life. Rumor dissemination is a process, and effective identification of rumors in the early stage of their appearance will reduce the negative impact of false rumors. This paper proposes a novel early rumor detection (ERD) model based on reinforcement learning. In the rumor detection part, a dual-engine rumor detection model based on deep learning is proposed to realize the differential feature extraction of original tweets and their replies. A double self-attention (DSA) mechanism is proposed, which can eliminate data redundancy in sentences and words at the same time. In the reinforcement learning part, an ERD model based on Deep Recurrent Q-Learning Network (DRQN) is proposed, which uses LSTM to learn the state sequence features, and the optimization strategy of the reward function is to take into account the timeliness and accuracy of rumor detection. Experiments show that, compared with existing methods, the ERD model proposed in this paper has a greater improvement in the timeliness and detection rate of rumor detection.

## 1. Introduction

With the rapid development of the Internet, social networks and people's lives have become increasingly close, and the participation and utilization rate of netizens has risen rapidly [1]. The global digital statistics report [2] released by "We Are Social" in 2019 shows that, by the end of 2018, there were 3.48 billion social network users in the world, accounting for 45% of the world's total population. Social network platforms represented by Twitter and Weibo provide netizens with functions to post information and express opinions. News media have gradually established official accounts on social networks for news reporting, so social networks have gradually become people's main sources of information.

5G and edge computing bring certain security issues to social networks [3, 4]. From the content point of view, the popularity of social networks improves life efficiency, but it has also become an environment for online rumors. An early study of social psychology defined rumors as "propositions spread without verification by relevant departments" [5]. Today, with the explosion of information, a huge amount of information is spread on social networks every day, including a lot of rumors. The harm of rumors to society cannot be ignored. For example, the 2011 Japanese earthquake triggered a tsunami that caused the Fukushima nuclear power plant to explode. The incident had little impact on China, but some illegal traders took the opportunity to drive up salt prices on the grounds that sea salt was contaminated by nuclear power. Salt prices in many places across the country have soared, and many people have been incited to rob salt in salt farms, which has seriously disrupted the order of social life. The purpose of the rumors is generally destructive, such as pranks and revenge on society. Because social networks have the characteristics of virtuality and anonymity, the cost of creating and spreading rumors is extremely low, and online rumors have become a trend of flooding. In the face of rumors rampant in the network environment, social network platforms have established rumor-defying accounts to manually refute rumors, but only relying on manual review to stop rumors is not only high in labor costs but also very inefficient, so artificial intelligence-based rumor detection technology has gradually become research hot spot.

The spreading process of rumors has obvious timeliness [6]. Specifically, rumors spread rapidly in the form of outbreaks when they appeared in the early days, but over time, their spread speed will be greatly reduced until they eventually die out. Figure 1 shows the spreading sequence diagram of a Twitter rumor. The red text represents the original tweet, the yellow text represents the reply message that questioned the original tweet, and the green text represents the reply message that opposes the original tweet.

The rumor was released after a shooting incident. The general content of the rumor was "According to the police, there were a large number of shooters in the shooting." Later, after investigation by relevant departments, there was only one shooter, and the police did not disclose the information. Within two hours after the rumor was released, there were a thousand reposts, and the rumor spread to tens of thousands of people. Analyzing the spreading process of this rumor, when there is obvious opposition and questioning information in the comment area, the information can be preliminarily judged as a rumor. If rumors can be accurately identified and their spreading behavior can be controlled when they appear early, the adverse effects of false rumors can be greatly reduced. Therefore, early rumors detection research on social networks is very important.

The main contributions of this paper are as follows. Aiming at the difference in content characteristics between original tweets and reply messages in Twitter, a dual-engine rumor detection model based on the self-attention mechanism is proposed, which improves the accuracy of rumor detection; in addition, we propose an early rumor detection (ERD) based on recurrent Q-learning, which can detect rumors earlier with higher accuracy.

The remainder of this paper is organized as follows: Section 2 briefly introduces the related works and research issues; Section 3 proposes an ERD model based on deep recurrent Q-learning and a dual-engine rumor detection model based on self-attention mechanism; Section 4 discusses and analyzes the experiment results. Section 5 draws the conclusion and proposes future research directions.

## 2. Related Works

*2.1. Rumor Detection.* The essence of the rumor detection problem is text classification. The current research on rumor detection is divided into two categories: methods based on traditional machine learning and methods based on deep learning. The former generally uses methods such as naive Bayes classification, decision trees, and support vector machines. Castillo et al. [8] used machine learning algorithms based on feature engineering to classify rumors and extracted a large number of text features based on the characteristics of rumors, including text length and the number of likes. On the basis of this method, many scholars [9–14] began to try to use different machine learning algorithms and richer features to study rumor detection.

Although the method based on machine learning can solve the problem of rumor detection to a certain extent, it is time-consuming, laborious, and inefficient in the feature engineering stage. The quality of the feature heavily relies on manual experience, which affects the quality of the rumor detection model. With the widespread application of deep learning in the field of natural language processing, researchers have also begun to use deep learning methods to solve the problem of rumor detection. The reply information of tweets has a great influence on the effect of rumor detection. Related researchers have proposed a multitask learning model for rumor detection and user stance detection. The most typical method is the multitask joint learning model proposed by Ma et al. [15], which is to define a shared layer as a bridge between the rumor detection deep learning model and the user stance detection deep learning model to exchange information. Li et al. [16] added user characteristics and attention mechanism on this basis to improve the performance of the model. In addition, with the emergence of the BERT (Bidirectional Encoder Representation from Transformers) language model, the rumor detection methods based on BERT have been proposed, such as the model of Yu et al. [14]. In addition, the rumor detection model based on the propagation tree has gradually attracted the attention of researchers. Its starting point is to abstract the tweets into a propagation tree according to the timeline and convert the rumors classification into tweet tree classification, such as Ma et al. [17] and Kumar's [18] research work.

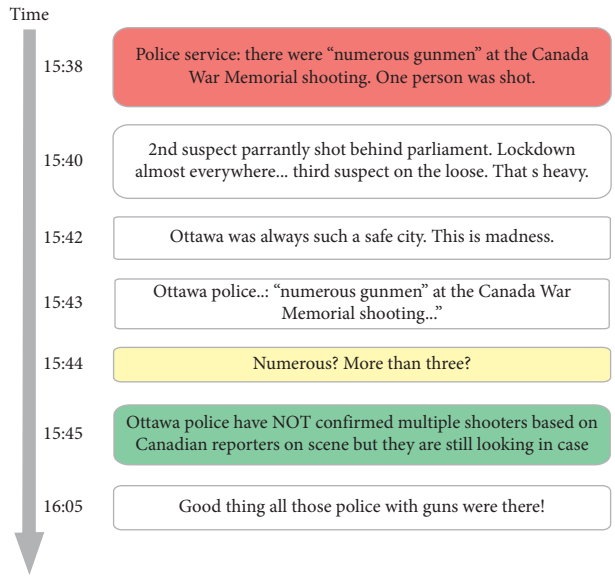



Figure 1: A rumor from PHEME dataset [7].

*2.2. ERD.* In the current research related to rumor detection, the research focus is mainly on improving the accuracy of rumor detection, and the early detection of rumors is less involved. The current ERD methods can be divided into three types:

(1) Real-time rumor detection, such as the model proposed by Castillo et al. [19]. This model uses a support vector machine to classify the original information without considering the reply information, so there is no delay in the detection time, so as

to achieve real-time detection. Although the real-time rumor detection method can ensure the detection of rumors in the early stage, it has a high rate of misjudgment and has little practical value.

(2) ERD based on static checkpoints. For example, Dungs et al. [20] proposed a detection method based on a hidden Markov model. This method uses a fixed number of replies as an interval when the model reads the reply information (the interval length used in the literature is 5). Set a static checkpoint; each checkpoint will consider whether to output the detection result. If the detection result outputs, the rumor detection process ends; otherwise, the reply information will continue to be read until a detection point appears and the result is output. Although this method can theoretically achieve early detection, it is not flexible enough to give play to the potential performance of the model.

(3) ERD based on reinforcement learning. For example, the model proposed by Zhou et al. [21] consists of two parts: a rumor detection module (RDM) and a checkpoint module (CKM). The CKM is implemented by a reinforcement learning model, which dynamically controls the number of input replies from the RDM enables ERD. The model can use the reinforcement learning method to constantly weigh the detection time and detection accuracy to achieve the best balance between the "early nature" and accuracy of rumor detection.

*2.3. Problems.* The original tweet and the reply message are two completely different messages. The original tweet is generally a complete description of an event, whose expression is more rigorous. Reply messages are towards the original tweet, sometimes even an emoticon or a punctuation mark. Existing models generally ignore the difference between the original tweets and their reply information, even though individual multitask joint learning models (such as Ma et al. [13]) use two independent networks to process the original tweet and the reply information, but these two independent networks are often two networks with the same structure. For these two kinds of information with large differences, it is not reasonable to use the same network for modeling, especially for the expression of chaotic word order such as comment information. If the recurrent neural network is only used for modeling, many potential features will be ignored. Therefore, it is necessary to separately model the characteristics of tweets and reply messages.

In terms of ERD, this paper is based on a reinforcement learning strategy to solve the problem of ERD, because the ERD model proposed by Zhou et al. [21] applies reinforcement learning to the ERD problem. The ERD model is composed of CKM and RDM. The CKM is implemented by DQN to control the number of reply messages input to RDM. The RDM is implemented by GRU. Through in-depth analysis of the model, it is found that the model has the following problems:

(1) Potential meaning of the state sequences are ignored

Reinforcement learning is generally based on the "Markov decision process," but in the case of ERD, it is more reasonable to regard the ERD process as a "partially observable Markov decision process" because the state sequence generated by RDM is potentially helpful for ERD.

Specifically, the state sequence refers to the coding sequence of known information formed by continuously inputting reply information to the RDM. For each state in the state sequence, the RDM can output the rumor classification result corresponding to the state. But for different states, even if the classification result output by the RDM is the same, the corresponding probabilities of the results are different. For this different probability sequence, the probability value may show a steady upward trend. For example, when RDM reads 2 reply messages, the probability of RDM outputting rumors and nonrumors is 0.55 and 0.45 (Softmax does the final classification; the sum of the two probabilities is 1), and the classification result is a rumor, but when 4 messages are read, the probability that RDM will output rumors and nonrumors is 0.85 and 0.15, and the classification result is still rumors; the difference is that the model will become more "certain." For the case where the classification probability changes steadily (the probability of one label increases; the probability of the other label will inevitably decrease because the probability sum is 1), the model can be allowed to output the detection results in advance, which can more effectively avoid the rumor detection process. By observing the partial change trend of the state to represent the actual state of the environment, this process is actually a partial Markov decision process. The CKM in ERD is implemented by DQN based on the Markov decision process, and the sequence features cannot be obtained, resulting in the model's poor performance in the timeliness of rumor detection.

(2) Incomplete reward function

The number of rumors in social networking platforms is much less than the number of nonrumors, so rumor detection is actually anomaly detection. In the field of anomaly detection, a model with a higher accuracy rate may be not the best one, and a model with a higher recall rate is often more practical [22]. The reward function used by ERD has the problem of uneven sample distribution, which leads to low model recall. In addition, there is

the problem of insufficient flexibility in the "early" detection strategy. The reward function of ERD is as follows:

$$r_i = \begin{cases} \log M & \text{terminal with correctprediction,} \\ -P & \text{terminal with wrong prediction,} \\ -\varepsilon & \text{continue.} \end{cases} \tag{1}$$

If the decision action is to terminate the reading, there will be two situations. If the prediction is correct, a reward of $\log M$ is obtained, where $M$ is the number of samples whose predictions are correct; if the prediction fails, the penalty is $-P$, where $P$ is a constant 100. If the action is to continue reading, it will be punished by $-\varepsilon$, where $\varepsilon$ is 0.01. This function specifically has the following problems:

(1) Predicting the correct reward of $\log M$ is intended to keep the model in a good state of performance in stages and to make the model converge as soon as possible, but the problem is that the model converges to the local optimal value, which reduces the generalization ability of the model.
(2) The prediction error is punished by $-P$; the purpose is to make the model make a "more cautious" judgment because once the prediction error is punished, the penalty is great. However, the model also has the problem of poor generalization ability, because when the number of rumors is small, if the misrecognition of rumors and misrecognition of nonrumors are treated equally, the recall rate of the model will decrease.
(3) Continuing to read the reply information is punished by $-\varepsilon$. The original intention is to make the model output the result as soon as possible, but the problem is that the model does not perceive the "urgency" of time. In other words, for a rumor message, if the model has read 2 replies and 50 replies, the penalty for continuing to read is the same. But in fact, the penalty should be greater after reading 50 replies, because the rumors may have spread over time and the results need to be reached as soon as possible.

## 3. Proposed Model and Algorithm

*3.1. Problem Description.* The goal of ERD is to achieve higher accuracy of rumor detection by collecting less information. This problem can be described as follows: set the input tweet as $X = \{x_0, x_1, \ldots, x_n\}$, where $x_0$represents the original tweet; others represent the reply information related to the original tweet and are sorted in chronological order. $x_i$ is composed of text information and metadata information. The classification result $y = \{$rumor, nonrumor$\}$, and $t \in [0, n]$ represents the number of reply messages used in the rumor detection process, so $t$ indirectly expresses the time-consuming detection process. Therefore, the purpose of ERD can be described as, for input $X$, it is necessary to accurately output tweet type $y$ when $t$ is as small as possible.

*3.2. Early Rumor Detection Model Based on DRQN.* In order to effectively solve the problems mentioned in Section 2, we propose an ERD model based on DRQN (Deep Recurrent Q-Learning Network). The basic model architecture is shown in Figure 2, which consists of a RDM and a control model.

*3.2.1. Control Model.* The control model is implemented by DRQN, which is a typical partially observable Markov decision algorithm. The recurrent neural network enables the model to have the memory function of the state sequence and then can learn the potential features in the state sequence. In the control module, this paper uses LSTM to realize the memory function of the state sequence. LSTM obtains the actions it considers reasonable by observing the state information and the last judgment.

The specific calculation process is shown in the following formula:

$$\begin{aligned} h_t &= \mathrm{LSTM}\left(\mathrm{state}_t, h_{t-1}\right), \quad t \in [1, n], \\ F &= W_f h_t + b_f, \\ p\left(\frac{y_i}{x_k}\right) &= \mathrm{sofmax}\left(F\right) = \frac{\exp\left(F_i\right)}{\sum_{j=1}^{m} \exp\left(F_j\right)}. \end{aligned} \tag{2}$$

Among them, in addition to receiving the current state information $\mathrm{state}_t$, the LSTM network also receives the LSTM neuron information $h_{t-1}$ at the previous moment. After outputting $h_t$, it passes through the fully connected layer to obtain a vector $F$ of length two, and finally, the action probability distribution is output through the softmax function.

It is worth noting that the input state of LSTM is the last vector used for classification in the RDM, and there are two output actions:

(1) Continue: It means that the current information is not enough to determine whether it is a rumor, and let the RDM read another reply message.
(2) Terminate: It indicates the end of the detection process and outputs the detection result. In other words, the RDM has sufficient information to judge whether the original tweet is a rumor and outputs the result in advance to achieve the purpose of early detection.

The reward function is the core of reinforcement learning. The quality of its design can directly determine the performance of the model. We design the following reward function:

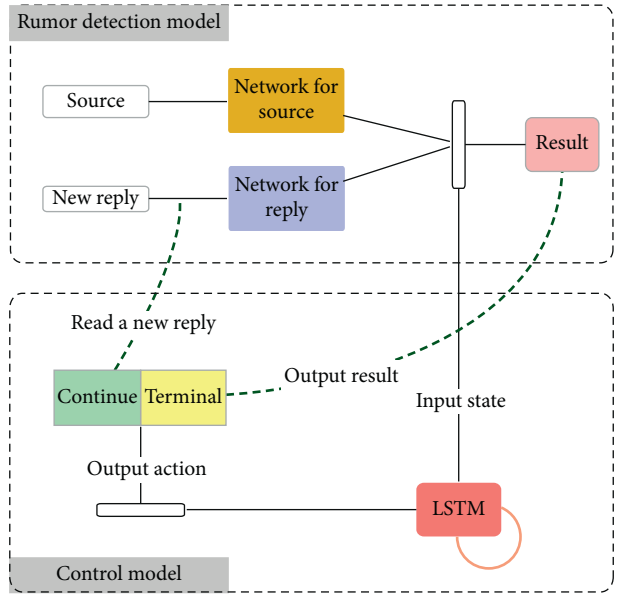



Figure 2: Architecture diagram of early rumor detection (ERD) model based on DRQN.

$$r_i = \begin{cases} R, & \text{terminal with correctprediction,} \\ -2P, & \text{terminal with wrongprediction, and label is rumor,} \\ -P, & \text{terminal with wrongprediction, and label is not rumor,} \\ -(\log n + \varepsilon), & \text{continue, } n = 1, 2, 3, \ldots. \end{cases} \tag{3}$$

Among them, when the model makes a stop-reading action, if the prediction is correct, it will directly get a reward of $R$ to avoid falling into the local optimum; if the prediction is wrong, there are two situations. When the actual label is a rumor, it will receive a $-2P$ punishment; when the actual label is nonrumor, it is punished by $-P$. The reasons for adopting this strategy include two aspects: (1) considering the fact that there are few rumor samples; (2) considering that the losses caused by the rumor detection system are different in the two cases of misjudgment, specifically comparing the effect of misjudging the information that was originally a rumor as not a rumor and the effect of identifying nonrumors as a rumor. Obviously, the former will have a greater impact, because the omission of the rumors by the model will spread the rumors to a greater extent, but for the latter, although the cost of misjudgment has been increased, no rumors have been missed. Therefore, if the information that was originally a rumor is judged to be not a rumor, the model will be punished twice.

When the model continues to read data, it will be punished by $-(\log n + \varepsilon)$, $n$ represents the number of reply messages read by the model, and $\varepsilon$ is a small value to avoid the situation where the penalty is 0 when reading the first reply message; the more the response information read, the greater the penalty for continuing to read.

*3.2.2. RDM.* In view of the difference between the original tweet and the reply information, this paper proposes a dual-engine RDM based on the self-attention mechanism. The specific model architecture is shown in Figure 3, which is mainly composed of the original tweet network and the reply information network.

In the network for source tweet, the text data passes through the word embedding layer, the GRU network, and the word-level self-attention mechanism in turn. The metadata feature extractor is used to extract the credibility characteristics of the tweet publisher and the basic information of the original tweet. The specific calculation process is as follows:

$$\begin{aligned} x_t &= \text{WordEmbedding}(w_t), \quad t \in [1, n], \\ \text{Output, } h_t &= GRU(x_t, h_{t-1}), \quad t \in [1, n], \\ \text{Output} &= [\text{output}_1, \text{output}_2, \ldots, \text{output}_n], \\ Q &= W_Q \,\text{Output} + b_Q, \\ K &= W_K \,\text{Output} + b_K, \\ V &= W_V \,\text{Output} + b_V, \\ f(Q, K_i) &= Q^T W_a K_i, \\ a_i &= \text{softmax}(f(Q, K_i)) = \frac{\exp(f(Q, K_i))}{\sum_j \exp(f(Q, K_i))}, \\ \text{Attention}(Q, K, V) &= \sum_i a_i V_i. \end{aligned} \tag{4}$$

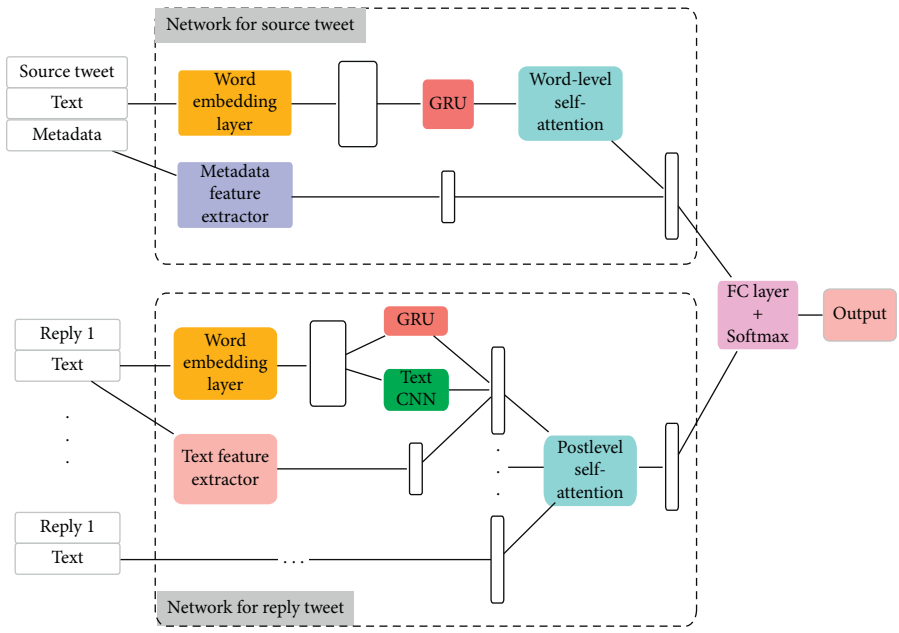



Figure 3: Architecture diagram of dual-engine rumor detection model (RDM) based on self-attention mechanism.

The features of metadata are shown in Table 1.

Since the reply information is usually expressed with strong emotional color, the expression is more casual, and there is an unstable word order; this paper considers using two-way GRU, Text-CNN and text feature extractor to extract the reply information features in parallel, and the final feature vector is constructed through vector splicing.

(1) Bidirectional GRU

The word order of reply messages is unstable. In order to extract more information, we use a bidirectional GRU network. The calculation details are shown in the following equation:

$$\begin{aligned} x_t &= \text{WordEmbedding}(w_t), \quad t \in [1, n], \\ \overrightarrow{h_{rt}} &= \overrightarrow{\text{GRU}}(x_t, h_{rt-1}), \quad t \in [1, n], \\ \overleftarrow{h_{lt}} &= \overleftarrow{\text{GRU}}(x_t, h_{rt+1}), \quad t \in [n, 1]. \end{aligned} \tag{5}$$

$\overrightarrow{\text{GRU}}$ represents the forward GRU, $\overrightarrow{h_{rt}}$ represents the vector representation of forward sequence of words $w_1 \sim w_n$ processed by $\overrightarrow{\text{GRU}}$, and similarly $\overleftarrow{h_{lt}}$ represents the vector representation of reverse sequence of words $w_n \sim w_1$ processed by GRU . At last, $\overrightarrow{h_{rt}}$ and $\overleftarrow{h_{lt}}$ are concatenated to obtain the final vector representation of the sentence, as shown in formula

$$h_t = \text{concat}\left(\overrightarrow{h_{rt}}, \overleftarrow{h_{lt}}\right). \tag{6}$$

(2) Text-CNN

Aiming at the random features of the way of replying information, this paper uses Text-CNN to extract the semantic features of abnormal word order. Text-CNN consists of convolutional layer and pooling layer.

The convolutional layer is used to extract text features. The process of extracting text features can be expressed by the following formula:

$$\begin{aligned} a_i &= f\left(W * M_{i:\, i+h-1} + b\right), \\ A &= \left[a_1, a_2, \ldots, a_{n-k+1}\right]. \end{aligned} \tag{7}$$

$M_{i:\, i+h-1}$ is the word vector from row $i$ to $i + h$ in the word vector matrix. After performing a linear transformation on $M_{i:\, i+h-1}$, the activation function $f$ is used to obtain $a_i$, which represents the $i$-th text feature extracted by the convolution kernel of length $h$. Finally, all the features extracted by the convolution kernel are spliced to obtain the vector $A$. The above process is the processing result of one convolution kernel, and the same steps are repeated for multiple convolution kernels.

In addition, the pooling function of this model adopts the maximum pooling function; that is, after the features extracted by the convolutional layer are obtained, one of the largest features is selected to represent all the features, which can be expressed by the formula

$$\tilde{a} = \max\left(a_1, a_2, \ldots, a_i\right). \tag{8}$$

(3) Text feature extractor

The text feature extractor can extract relatively intuitive text features, such as statistical negative words, whether there is an exclamation mark, and the similarity with the original tweet, as shown in Table 2.

Table 1: Features of metadata.

| Feature name | Description |
|---|---|
| has_url | Whether the tweet contains URL or hyperlink |
| urls | The number of URLs or hyperlinks contained in the content of the tweet |
| has_tag | Whether to use the keyword "#" to carry a topic in the content of the tweet |
| tags | How many topics are carried with the keyword "#" in the content of the tweet |
| favorite_count | The number of times the tweet was liked |
| retweet_count | The number of tweets reposted |
| verified | Is the user authenticated by real name on Twitter? |
| profile_use_background_image | Whether Twitter users set a background image on their homepage |
| default_profile_image | Whether the background image set by Twitter users on their homepage is the system default |
| geo_enabled | Whether users disclose their location information |
| is_translation_enabled | Whether the user has the translation permission |
| default_profile | Whether users modify the default personal information |
| friends_count | Number of users followed |
| followers_count | User attention |
| statuses_count | Number of tweets posted by users |
| description_len | User profile length |
| favourites_count | Cumulative number of likes of user tweets |
| listed_count | How many public channels are users involved |
| user_age | User registration period |
| coordinates | Does the tweet contain coordinate information |
| coordinates | Whether the tweet contains URL or hyperlink |

Table 2: Features of text.

| Feature name | Description |
|---|---|
| word_count | Number of words in the sentence |
| is_question | Describe whether a question mark appears in the sentence |
| is_exclamation | Describe whether an exclamation mark appears in the sentence |
| negation_count | Number of negative words, such as "no, nothing" |
| badword_count | Number of bad words, such as "fuck, bitch" |
| similar_to_src | Cosine similarity to the original tweet |

The calculation formula of cosine similarity is as follows:

$$\cos(\theta) = \frac{\sum_{i=1}^{n}(x_i \times y_i)}{\sqrt{\sum_{i=1}^{n} x_i^2} \times \sqrt{\sum_{i=1}^{n} y_i^2}}. \tag{9}$$

(4) Postlevel self-attention mechanism

The self-attention mechanism at the postlevel refers to the weighting of the self-attention mechanism after feature extraction of all response information so that the model can pay attention to the useful response information. The specific calculation process is shown in the following formula:

$$R = [\text{reply}_1, \text{reply}_2, \ldots, \text{reply}_n], \tag{10}$$

$$Q = W_Q R + b_Q, \tag{11}$$

$$K = W_K R + b_K, \tag{12}$$

$$V = W_V R + b_V. \tag{13}$$

In formula (10), $\text{reply}_n$ represents the encoding of a reply message by the reply message network. $R$ is the set of reply message codes. The self-attention mechanism requires three vectors $Q$, $K$, and $V$ to represent query, key, and value, respectively, which are obtained from the vector $R$ through three different linear transformations. Attention can be calculated based on these vectors. The specific calculation process is shown in the following formula:

$$f(Q, K_i) = Q^T W_a K_i, \tag{14}$$

$$a_i = \text{softmax}(f(Q, K_i)) = \frac{\exp(f(Q, K_i))}{\sum_j \exp(f(Q, K_i))}, \tag{15}$$

$$\text{Attention}(Q, K, V) = \sum_i a_i V_i. \tag{16}$$

In formula (14), $f(Q, K_i)$ is a general linear transformation to calculate the similarity between $Q$ and $K$. The weight $a_i$ that needs attention for each post is obtained through the sotfmax function, and the product of the weight and the vector $V$ is expressed as the response information processed by attention.

*3.3. Model Building.* In the process of building an ERD model, pretraining RDM and time difference method are used to train CTM (control model).

(1) Pretraining RDM

There must be a reliable RDM as a basis before training CTM. This pretrained RDM reads all the response information during the training process. This also means that the performance of the ERD finally obtained after joining the CTM will not be higher than the performance of the pretrained RDM. The significance of adding CTM is how to make RDM get the best performance with the least information, and its best performance is that of the pretrained RDM.
In the RDM training process, the batch size is 80, the dropout is 0.4, and the loss function is cross-entropy loss function. In addition, Adam optimizer is used in the model and the initial learning rate is 0.0005. In the training process, when the model's loss on the validation set does not decrease twice in a row, the learning rate is reduced by 10 times. When the model's loss on the validation set stabilizes, the training process stops.

(2) Training CTM

If CM is trained directly, its parameters will lack stability, which will make it difficult for the model to converge. Therefore, in order to speed up the convergence, we use a dual network structure and experience playback mechanism to train CTM.
Dual networks are two CTM networks with the same structure, which are called: current network (parameter $\theta$) and target network (parameter $\theta'$), respectively. The experience pool stores $n$ four-tuple records $(s_t, a_i, r_t, s_{t+1})$ about the environment, and a batch of samples are randomly taken from the experience pool for training each time. The training process is to train the current network first and update the target network with the parameters of the current network when a certain batch is reached.
Because the rumor detection problem is relatively special, traditional reinforcement learning is aimed at an environment, such as a game scene, but in the rumor detection problem, each rumor data is actually an environment, so it is necessary to consider multiple environments in the process of constructing the experience database. For environmental factors, the specific training process is shown in Algorithm 1.

## 4. Experiments Results and Analysis

*4.1. Experimental Environment.* The experimental environment is shown in Table 3.

*4.2. Dataset.* The experimental datasets include the public rumor dataset PHEME Dataset of Rumors and Non-Rumors (PHEME Rumor) [7], which is rumors and nonrumors data about five breaking news events collected on Twitter by ArkaitzZubiaga in 2016. The data distribution of each breaking news event is shown in Table 4.

The dataset is divided into training dataset, verification dataset, and test dataset with the ratio of 7 : 1 : 2. The validation set is used to observe the real-time training results. Based on the best training results, the best model is selected as the final model. Finally, the test set is used to test the performance of the model.

*4.3. Baselines.* This paper selects the following RDM as the baseline algorithm:

(1) CRF: the RDM proposed by Zubiaga et al. [23].
(2) GAN-GRU: the RDM proposed by Ma et al. [24].
(3) RDM: the RDM in the model proposed by Zhou et al. [23].
(4) LSTM: an LSTM network for rumor veracity proposed by Singh et al. [25].
(5) LSTM-Attention: attention-based LSTM network for rumor veracity proposed by Singh et al. [26].
(6) SA-SE: the model proposed in this paper uses only a single-engine model using a word-level attention mechanism (only the original information network).
(7) SA-DE: the model proposed in this paper only uses a dual-engine model using sentence-level attention mechanism.
(8) DSA-DE: the model proposed in this paper is based on the DSA dual-engine model.

In the current ERD research, only the ERD [21] uses reinforcement learning to solve the ERD problem, so we consider using ERD as a baseline method. The control variable method is used in the comparison link, and the submodels of the two models are cross-combined into multiple models for comparison experiments.

First, the reinforcement learning module in ERD is named RL1, and the RDM is named RDM1; the reinforcement learning module in the model proposed in this paper is named RL2, and the RDM is named RDM2. Finally, the experimental models to be compared are divided into the following three models:

(1) RDM1_RL1: ERD model
(2) RDM2_RL1: the model proposed in this paper only contains the RDM and the control module in the ERD
(3) RDM2_RL2: the ERD model proposed in this paper

## 5. Experimental Results and Analysis

*5.1. Rumor Detection Performance Evaluation.* Comparative experimental results of eight RDM are shown in Table 5.

It can be seen from Table 5 that the GAN-GRU performs well in terms of accuracy, but it has poor performance in terms of precision and recall; the model of Zhou et al. is

Input: Network $Q(s,a)$, Environment set $E$, Experience pool $P$
Output: $Q(s,a,\theta')$
(1) Initialize current network $Q(s,a,\theta)$, and target network $Q(s,a,\theta')$, $\theta' = \theta$
(2) **for** each epoch **do**
(3) Select an environment $e$ from $E$
(4) Initialize environment $e$, and get state $s_t$
(5) **while** true **do**
(6) According to $s_t$, use $\epsilon$-greedy strategy to select action $a_t$ from $Q(s,a,\theta)$
(7) Perform action $a_t$ in the environment to get the new state $s_{t+1}$ and reward $r_t$
(8) **if** $P$ is full **do**
(9) Delete the oldest experience record
(10) **end if**
(11) Insert $(s_t, a_t, r_t, s_{t+1})$ into $P$
(12) $s_t \leftarrow s_{t+1}$
(13) **if** $s_t$ is the last state **do**
(14) **break**
(15) **end if**
(16) **end while**
(17) **if** $P$ is full **do**
(18) Select a batch of records from $P$ randomly
(19) **for** each record **do**
(20) Use target network to get $y_t = r_t + \gamma \max_{a_{t+1}} Q(s_{t+1}, a_{t+1}, \theta')$
(21) Use loss function $(y_t - Q(s_t, a_t, \theta))^2$ to update current network $Q(s,a,\theta)$
(22) Update current network with target network every $n$ epochs
(23) **end for**
(24) **end if**
(25) **end for**

Algorithm 1: Training DRQN.

Table 3: Experimental environment.

| Name | Version |
|---|---|
| CPU | Intel(R) Xeon(R) Platinum 8160 CPU @ 2.10 GHz |
| GPU | NVIDIA Tesla P4 8 GB GPU |
| OS | Centos7.4 |
| Memory | 4 GB |
| PyTorch | 1.7.1 |

relatively stable; the LSTM-Attention achieved the highest score in precision; and the models proposed in this paper are SA-SE, SA-DE, and DSA-DE that have achieved good results in all four indicators. When the model adds the double self-attention (DSA) mechanism and the dual-engine network, the DSA-DE is compared to the sentence-level dual-engine network model SA-DE, and the single-level attention and single-engine network model SA is compared with SE; the accuracy rate is increased by 2.3% and 5%, respectively. It shows that the dual-engine network and DSA mechanism have improved the performance of the rumor detection task.

Figure 4(a) describes the change trend of the model SA-SE, SA-DE, DSA-DE loss value for the validation set during the training process. It can be seen from the figure that the DSA-DE model achieved the lowest loss. Although the initial loss of the SA-DE model that does not use the sentence-level attention mechanism decreases the fastest, the final training effect is not as good as the DSA-DE model that uses the DSA mechanism. It can be seen that both the DSA mechanism and the dual-engine network can improve the learning ability of the model.

Figure 4(b) shows the change trend of the model SA-SE, SA-DE, DSA-DE accuracy for the validation set during the training process. It can be seen that the DSA-DE model has the fastest learning ability. After 10 rounds of training, the accuracy of the model has stabilized at around 85%. Although the SA-DE model that does not use word-level attention has strong initial learning ability, the final learning result does not exceed the DSA-DE model. Therefore, both the DSA mechanism and the dual-engine network can improve the learning ability of the model.

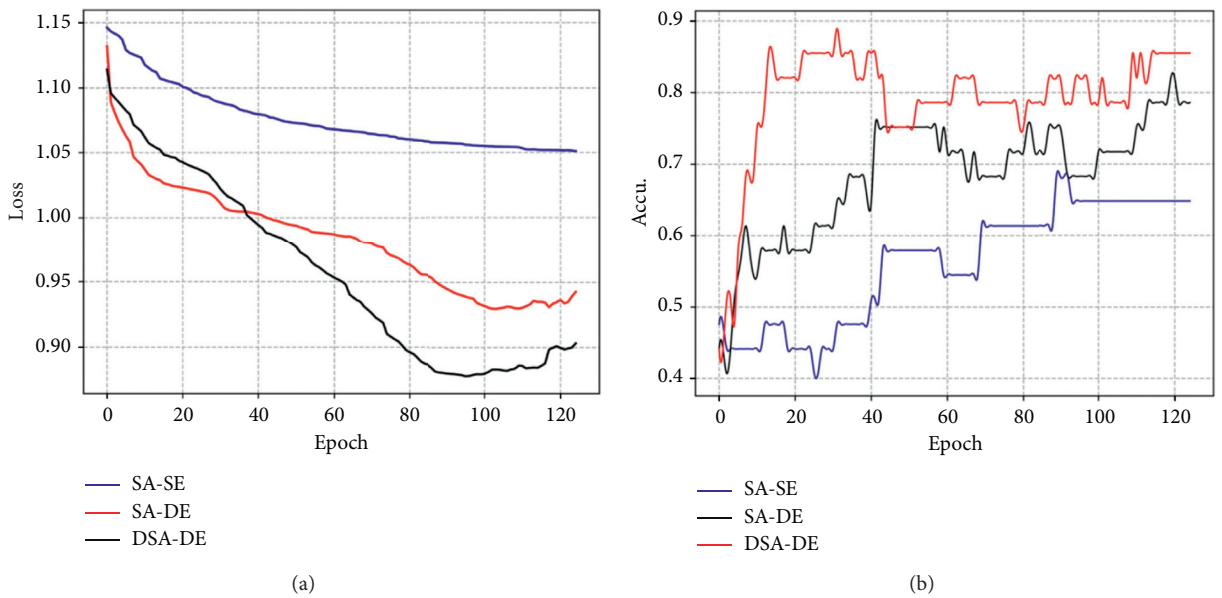
*5.2. ERD Efficiency Evaluation.* Since we utilize reinforcement learning theory to achieve the purpose of ERD by controlling the number of replies input. In order to evaluate the early nature of the model, firstly, we use the standard indicators such as accuracy, precision, recall, and $F_1$ score.

It can be seen from Table 6 that when the control module RL1 is added, the accuracy of RDM2_RL1 is 0.05 higher than that of RDM1_RL1, indicating that the performance of the dual-engine RDM based on the DSA mechanism proposed in this paper is better than the rumor in ERD Detection module. Comparing the control modules, when RDM2 is added to the control modules RL1 and RL2, the accuracy rates drop to 0.80 and 0.81, respectively, indicating that the DRQN-based control module proposed in this paper can maintain a high accuracy rate.

Secondly, we use the average number of responses for each sample as one of the evaluation indicators, which is recorded as mean posts used. In order to more intuitively

Table 4: Rumor dataset distribution.

| Event | Number of rumors (proportion) | Number of rumors (proportion) |
|---|---|---|
| Charlie Hebdo | 456 (22.0%) | 1621 (78.0%) |
| Ferguson | 284 (24.8%) | 859 (75.2%) |
| Germanwings Crash | 238 (50.7%) | 231 (49.3%) |
| Ottawa Shooting | 470 (52.8%) | 420 (47.2%) |
| Sydney Siege | 522 (42.8%) | 699 (57.2%) |
| Sum | 1970 (34.0%) | 3830 (66.0%) |




(a) (b)

Figure 4: Comparison of RDM during training. (a) Loss value trend analysis. (b) Accuracy analysis.

Table 5: Experimental results of RDM.

| Model name | Accuracy | Precision | Recall | $F_1$ score |
|---|---|---|---|---|
| CRF | 0.67 | 0.67 | 0.56 | 0.60 |
| GAN-GRU | 0.78 | 0.53 | 0.35 | 0.42 |
| RDM | 0.77 | 0.74 | 0.74 | 0.74 |
| LSTM | 0.81 | 0.77 | 0.69 | 0.72 |
| LSTM-Attention | 0.83 | **0.83** | 0.79 | 0.81 |
| SA-SE | 0.78 | 0.70 | 0.73 | 0.71 |
| SA-DE | 0.81 | 0.79 | 0.80 | 0.80 |
| DSA-DE | **0.84** | 0.81 | **0.82** | **0.82** |

reflect the performance of the model in terms of accuracy and timeliness, we propose an evaluation indicator called early detection rate:

$$\text{early detection rate} = \frac{F_1\_\text{Score} * 10}{\text{mean posts used}}. \quad (17)$$

Figure 5 shows the experimental results of the models RDM1_RL1, RDM2_RL1, and RDM2_RL2 with the control module added in the early detection. The average number of reply messages used and the early detection rate are shown in Figures 5(a) and 5(b), respectively. It can be seen that the model RDM2_RL2 proposed uses the least amount of information. On average, each piece of data uses only 1.004 reply messages and has an early detection rate of 8.058, indicating that the model proposed in this paper can find a better balance between accuracy and timeliness so that the model can identify the rumors in the early stage while ensuring the accuracy.

Figure 6 shows the change of the average reward value of the model RDM2_RL2 during the training process. It can be seen that the average reward value of the model is stable between 23 and 24 after 40 rounds of training, indicating that the DRQN-based control module proposed is effective.

Figure 7 shows the changes in the early detection rate of models RDM1_RL1, RDM2_RL1, and RDM2_RL2. It can be seen that, compared to the model RDM1_RL1, the model RDM2_RL1 has the same learning ability, but the final result

Table 6: Experimental results of ERD model.

| Model name | Accuracy | Precision | Recall | $F_1$ score |
|---|---|---|---|---|
| RDM1_RL1 (ERD) | 0.75 | 0.73 | 0.73 | 0.75 |
| RDM2_RL1 | 0.80 | 0.78 | 0.78 | 0.80 |
| RDM2_RL2 | **0.81** | **0.79** | **0.79** | **0.81** |

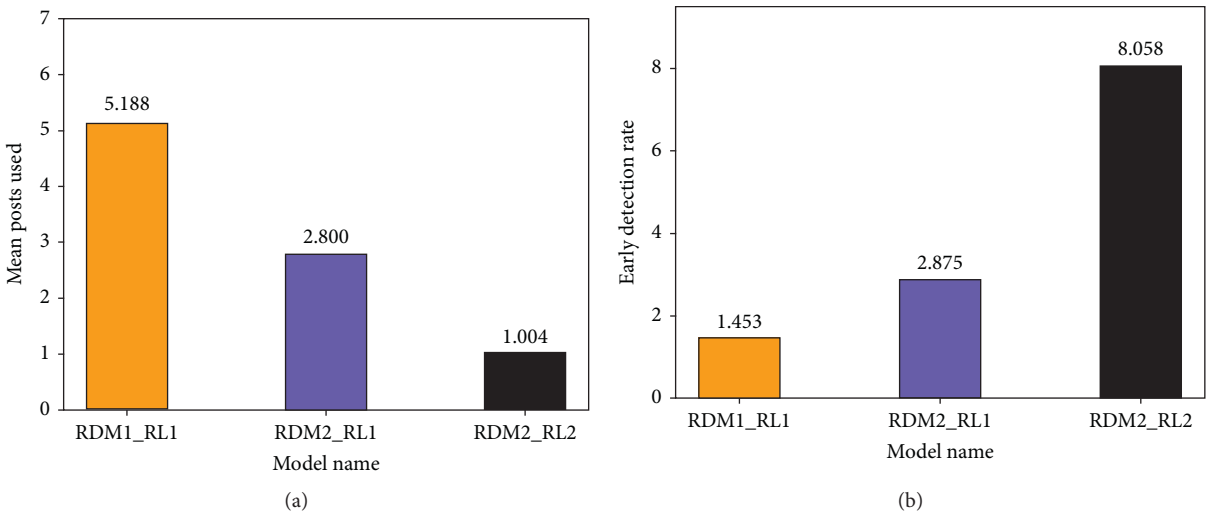



Figure 5: ERD comparison experiment results. (a) Mean posts used. (b) Early detection rate.

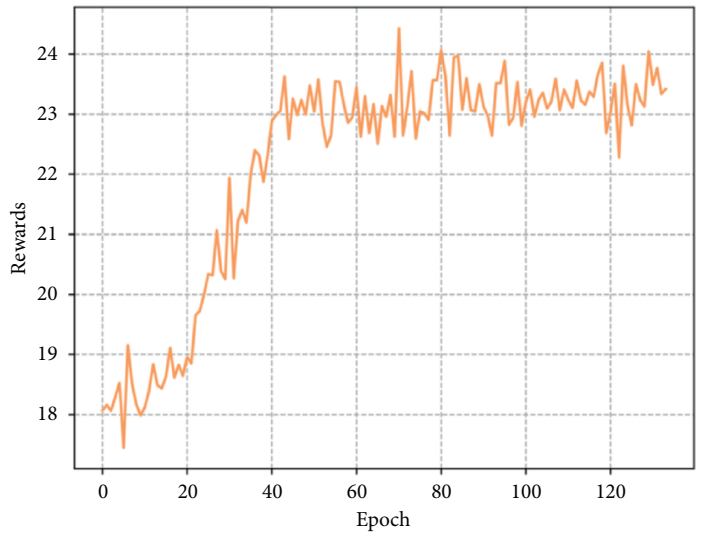



Figure 6: Reward value changing state during training.

of RDM2_RL1 training is better than RDM1_RL1. It shows that the RDM proposed is more helpful to detect rumors. The model RDM2_RL2 proposed has a stronger learning ability. It reaches the peak of early detection rate at about 50 rounds of training, and the early detection rate far exceeds RDM1_RL1 and RDM2_RL1. Therefore, the model proposed has good performance in the accuracy and timeliness of rumor detection.

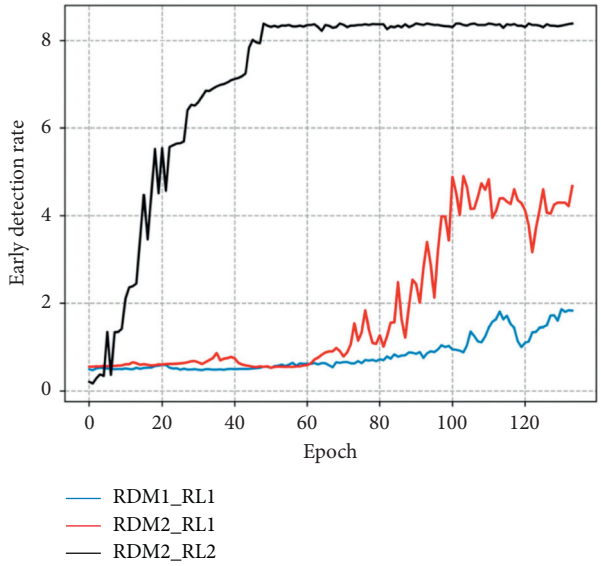



Figure 7: Early detection rate changing state during training.

## 6. Conclusions

In terms of rumor detection, this paper first analyzes the existing research on three problems: the inability to obtain the optimal representation of the reply information, the ignorance of the difference between the original tweet and the reply information in the tweet, and the inability to handle redundant data well. In response to the above problems, this paper uses the difference between the original tweet and the reply information in the Twitter data as an entry point and proposes a dual-engine RDM, which separately deals with the original tweet and the reply information; on the remaining problem, the DSA mechanism is proposed to solve the problem of data redundancy in the two dimensions of sentences and words. For the existing multitask model, there is a problem that the optimal representation of the reply information cannot be obtained. This paper uses a single-task learning model., Let the model itself learn to encode the reply message. The final experimental results show that the method proposed in this paper has a better detection effect.

In terms of ERD, this paper considers solving the problem of ERD from the perspective of reinforcement learning. First, the following problems are found through analysis of existing research: the potential meaning of the state sequence is ignored, the reward function is imperfect, and the performance of the RDM is poor. In response to the above problems, this paper proposes an ERD model based on DRQN and describes the model in detail. In order to analyze the experimental results more effectively, this paper proposes the evaluation index of ERD rate to evaluate the performance of the ERD model. Finally, this paper is verified on the rumor dataset. The experimental results show that this paper can detect rumors earlier under the premise of ensuring the accuracy of rumor detection.

Although the model proposed in this paper has achieved good results by comparing the baseline method, it can still be optimized from the following perspectives.

In natural language processing tasks, the data cleaning stage cannot be ignored. In future research, more fine-grained methods can be considered to clean information such as words, sentences, special symbols, and URLs.

In terms of rumor detection, the language model can be modeled using the relatively new BERT. BERT can learn specific expressions of words in specific language scenarios, and the model may achieve better performance. In addition, in terms of features, more meaningful features can be explored for experimentation.

In the ERD problem, you can use more powerful reinforcement learning algorithms such as A3C to model. The reward function and training method can also be further optimized. In addition, some potential features in time can be explored for modeling. Of course, the problem of ERD is not necessarily limited to reinforcement learning, and there may be more suitable methods for ERD.

## Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

## Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

## Acknowledgments

This work was supported by the NSFC-Xinjiang Joint Fund Key Program (Grant no. U2003206) and the National Natural Science Foundation of China (Grant no. 61972255).

## References


[1] G. Q. Sun, W. P. Shi, and L. Wang, "A review of group behavior in online social networks and future prospects," *Journal of Modern Information*, vol. 36, no. 2, pp. 38–42, 2016.

[2] We Are Social, Global Digital Report, 2019, https://wearesocial.com/global-digital-report-2019.

[3] T. Wang, Y. Lu, J. Wang, H.-N. Dai, X. Zheng, and W. Jia, "EIHDP: edge-intelligent hierarchical dynamic pricing based on cloud-edge-client collaboration for IOT systems," *IEEE Transactions on Computers*, p. 1, 2021.

[4] T. Wang, Y. X. Mei, X. X. Liu et al., "Edge-based auditing method for data security in resource-constrained Internet of Things," *Journal of Systems Architecture*, vol. 114, Article ID 101971, 2021.

[5] R. H. Knapp, "A psychology of rumor," *Public Opinion Quarterly*, vol. 8, no. 1, pp. 22–37, 1944.

[6] M. Al-Sarem, W. Boulila, M. Al-Harby et al., "Deep learning based rumor detection on microblogging platforms: a systematic review," *IEEE Access*, vol. 7, pp. 152788–152812, 2019.

[7] A. Zubiaga, M. Liakata, and R. Proctor, *PHEME Dataset of Rumours and Non-Rumours Dataset*, https://figshare.com/articles/PHEME_dataset_of_rumours_and_non-rumours/4010619, 2016.

[8] C. Castillo, M. Mendoza, and B. Poblete, "Information credibility on Twitter," in *Proceedings of the 20th International Conference on World Wide Web*, pp. 675–684, Hyderabad, India, 2011.

[9] S. Kwon, M. Cha, K. Jung et al., "Prominent features of rumor propagation in online social media," in *Proceedings of the IEEE 13th International Conference on Data Mining*, pp. 1103–1108, Dallas, TX, USA, 2013.

[10] S. Vosoughi, *Automatic Detection and Verification of Rumors on Twitter*, Massachusetts Institute of Technology, Cambridge, MA, USA, 2015.

[11] K. Wu, S. Yang, and K. Q. Zhu, "False rumors detection on sinaweibo by propagation structure," in *Proceedings of the 2015 IEEE 31st International Conference on Data Engineering*, pp. 651–662, Seoul, South Korea, April 2015.

[12] Y. Wu, H. Huang, Q. Wu, A. Liu, and T. Wang, "A risk defense method based on microscopic state prediction with partial information observations in social networks," *Journal of Parallel and Distributed Computing*, vol. 131, pp. 189–199, 2019.

[13] J. Ma, W. Gao, Z. Wei et al., "Detect rumors using time series of social context information on microblogging web-sites," in *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, pp. 1751–1754, Melbourne, Australia, October 2015.

[14] J. Yu, J. Jiang, L. Min et al., "Coupled hierarchical transformer for stance-aware rumor verification in social media conversations," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*, pp. 1392–1401, November 2020.

[15] J. Ma, W. Gao, and K. F. Wong, "Detect rumor and stance jointly by neural multi-task learning," in *Proceedings of the 2018 Companion: The 2018 Web Conference Companion*, pp. 585–593, Lyon, France, April 2018.

[16] Q. Li, Q. Zhang, and L. Si, "Rumor detection by exploiting user credibility information, attention and multi-task learning," in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 1173–1179, Florence, Italy, July 2019.

[17] J. Ma, W. Gao, and K. F. Wong, *Rumor Detection on Twitter with Tree-Structured Recursive Neural Networks*, Association for Computational Linguistics, Stroudsburg, PA, USA, 2018.

[18] S. Kumar and K. M. Carley, "Tree LSTMS with convolution units to predict stance and rumor veracity in social media conversation," in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 5047–5058, Florence, Italy, July 2019.

[19] A. Gupta, P. Kumaraguru, C. Castillo, and P. Meier, "TweetCred: real-time credibility assessment of content on twitter," in *Proceedings of the International Conference on Social Informatics*, pp. 228–243, Barcelona, Spain, November 2014.

[20] S. Dungs, A. Aker, N. Fuhr et al., "Can rumour stance alone predict veracity?" in *Proceedings of the 27th International Conference on Computational Linguistics*, pp. 3360–3370, Santa Fe, NM, USA, August 2018.

[21] K. Zhou, C. Shu, B. Li et al., "Early rumour detection," in *Proceedings of the 2019 Conference of the North*, Minneapolis, MN, USA, 2019.

[22] X. Wang, T. Zhang, and Y. G. Jin, "Overview of anomaly detection algorithms," *Modern Computer*, no. 30, pp. 21–26, 2020.

[23] A. Zubiaga, M. Liakata, and R. Procter, "Exploiting context for rumour detection in social media," in *Proceedings of the International Conference on Social Informatics*, pp. 109–123, Oxford, UK, September 2017.

[24] J. Ma, W. Gao, and K. F. Wong, "Detect rumors on Twitter by promoting information campaigns with generative adversarial learning," in *Proceedings of the World Wide Web Conference*, pp. 3049–3055, Geneva, Switzerland, October 2019.

[25] J. P. Singh, P. R. Nripendra, and Y. K. Dwivedi, "Rumour veracity estimation with deep learning for Twitter," in *Proceedings of the International Working Conference on Transfer and Diffusion of IT*, Accra, Ghana, 2019.

[26] J. P. Singh, A. Kumar, N. P. Rana et al., "Attention-based LSTM network for rumor veracity estimation of tweets," *Information Systems Frontiers*, pp. 1–16, 2020.