Received: 26 March 2020 | Revised: 18 September 2020 | Accepted: 2 November 2020

DOI: 10.1002/cpe.6083

WILEY

RESEARCH ARTICLE

Social rumor detection based on multilayer transformer encoding blocks

Lijun Lin[1] | **Zhiyun Chen**[2]

[1]School of Computer Science and Technology, East China Normal University, Shanghai, China

[2]School of Data Science and Engineering, East China Normal University, Shanghai, China

Correspondence
Zhiyun Chen, School of Data Science and Engineering, East China Normal University, Shanghai, China.
Email: chenzhy@cc.ecnu.edu.cn

Abstract

The propagation of rumors on social media has been identified as a critical problem in recent years that causes social panic or economic turmoil (to some extent), thereby giving rise to the need for faster identification. With the advancements in deep learning, researches based on neural networks become popular. Most of the existing methods extensively adopt recurrent neural networks (RNNs), such as gated recurrent unit and long short-term memory. This results in a significant degradation in the concurrency performance of the models, implying an increased consumption of time and resources. This study proposes a model with multilayer transformer encoding blocks for detecting rumors. The self-attention mechanism in the transformer encoding blocks provides better concurrency to the proposed model and improves its performance. The proposed model performs faster executions using less or no recurrences in comparison to other models based on RNNs. Experiments on two real social datasets verified that our model could achieve significantly better results than baseline methods. The accuracy rate increased by 1.1% and 2.2% on the Weibo and PHEME datasets, respectively. In comparison to the methods that use RNNs as a feature extractor, the training duration in the proposed model was reduced by 16% and 70% in Weibo and PHEME, respectively.

KEYWORDS

rumor detection, semantic feature extraction, transformer

1 | INTRODUCTION

At present, people spend more time reading the latest news and information from social networking platforms, such as Instagram, Weibo, Facebook, and Twitter. Netizens enjoy the convenience of social networking platforms. However, some criminals propagate rumors or fake news on social media to get public attention. While some rumors can be easily identified, others cannot in a timely manner, which may cause social panic and economic turmoil. Facebook recently released the list of 100 fake news stories for 2019. Despite the extensive efforts to combat rumors, social media platforms are still overflowing with fake news. In 2019, Facebook's top 100 fake news were viewed 150 million times, which continues to increase even now. Researches show that a year ahead from the 2020 U.S. presidential election, a flood of fake news has already consumed Facebook.

Some websites, such as snopes.com, and journalists play a significant role in tracking rumors. Figure 1 shows a news article from Facebook's top 10 rumors (2019), which has been clarified as a false rumor by snopes.com[1]. However, this clarification is done manually by journalists, which

[1]https://www.snopes.com/fact-check/obamacare-website-border-wall/

FIGURE 1 Example of a rumor clarified up by snope.com

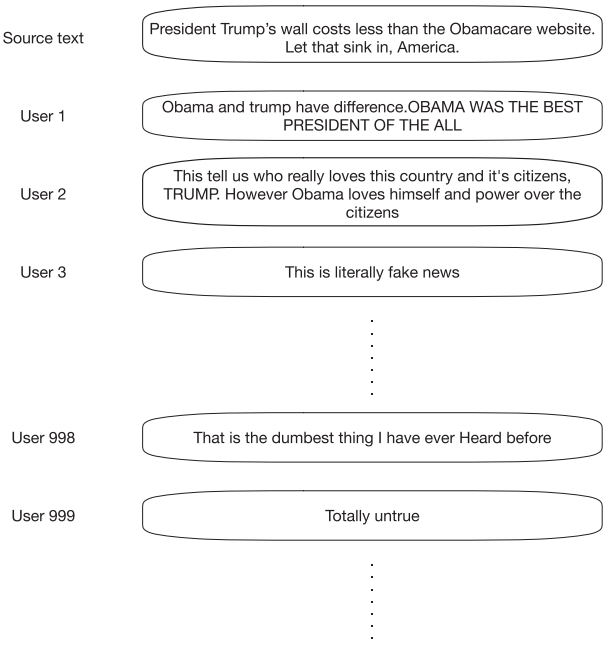


FIGURE 2 One of Top 10 Facebook Rumors 2019

requires a significant amount of time. Therefore, this method is inefficient, and there is an urgent need to determine an effective method to detect rumors.

Deep learning has become significantly popular in recent years because it helps in solving various problems. The concept of recurrent neural networks (RNNs)[1] with an attention mechanism as a feature extractor is one of the best approaches that has been proposed in the field of natural language processing (NLP), including rumor detection in social media. However, the low concurrency of multilayer RNNs will lead to long execution time both in the training and testing phases. The popularity of deep learning has warranted the need for research[2,3] using RNNs, which has significantly reduced the efficiency of the models.[2,3] A lack of concurrency indicates that more time and resources are required during the training phase. In addition, the use of RNNs and convolutional neural networks (CNNs) cannot effectively provide global semantic features.

Transformers were first proposed by Vaswani et al.,[4] which mainly consist of encoders and decoders. A transformer, as a new feature extractor for neural machine translation (NMT) tasks, is superior to RNNs in most cases; it is a method used for managing problems related to sequence models. Traditional NMT mostly uses RNNs or CNNs as the model basis for the encoders and decoders. However, the transformer is only based on attention, which has high parallelism and fast training speed to improve the performance of translation. Tang et al.[5] postulated that the efficient performance of transformers can be attributed to their ability to capture the semantic features and long-term dependencies. Figure 2 shows the source text and some comments mentioned on the news article presented in Figure 1. If the quantity of comments becomes huge, it becomes difficult for the RNN-based methods to extract the global features owing to the long distance. For example, in Figure 2, the comments made by User 1 and User 2 show the effects of the source text, that is, fake news. However, when the number of comments increased significantly, the comment made by User 999 may not be affected by the source text (fake news) or previously posted comments. The existing RNN-based methods can extract the local features; however, they exhibit poor performance while extracting the global features.

Rumors can be identified by professionals or people with experience in the respective domains. However, normal netizens can either believe or doubt such events. Netizens then share or retweet these events to their Twitter or Facebook accounts with their opinions of either supporting or contradicting them. Their attitudes are expressed through comments, which can be a key factor in rumor detection. Linguistic features can be directly extracted from the text of comments. This study focused on the impact of comments. All comments were considered because they can all be influencing factors in one event. We used multilayer transformer encoding blocks to capture the relationship between the source text and comments of an event to train a classifier to recognize an event is a rumor.

The main contributions of this study are as follows:

- We devised a model based on transformer encoding blocks for rumor detection that can extract the global semantic features.
- Our model exhibits better performance that the existing methods during the training and testing processes.
- Experiments demonstrated that the proposed model is more efficient and superior that the existing methods for rumor detection and early detection tasks.

The remainder of this article is organized as follows. Section 2 presents the research status of rumor detection. Section 3 elaborates on the theoretical basis of our research. In Section 4, the experimental results are presented. Finally, Section 5 presents a summary of this study and our future work.

2 | RELATED WORK

Rumor detection on social media aims to use the relevant information about an event to identify whether it is a rumor. Bondielli and Marcelloni[6] recently reviewed the methods for detecting fake news and rumors. Relevant information includes texts, propagation patterns, and personal information of users. According to the information or method used, the existing research consists of the following three categories:[7] (1) traditional machine learning methods based on artificial design features, (2) methods based on the social rumor propagation process, and (3) methods based on the original rumor and the comments on it.

Traditional rumor detection methods designed the classification features base on the relevant information of an event. The identified rumor and nonrumor data are used as the training data training the classifiers. Castillo et al.[8] extracted emotion scores, URLS, number of days since a user signed up, and other features; they employed a decision tree algorithm to detect rumors. Yang et al.[9] used the geographic locations involved in microblogs, user login information for retweeting, sentimental polarity of text symbols, and other features, and used an support vector machine (SVM) classifier to detect rumors. Kwon et al.[10] proposed a new periodic time series model, which proved that rumors may fluctuate over time. Subsequent researches extracted more potential features related to the content, users, propagation, and timing.[11-14] These researches extracted various shallow features from datasets; however, shallow features depend on the artificial design, that is, they cannot provide better robustness.

Methods related to the propagation patterns focus on the differences between the true and false information in the communication process, instead of the text or subscriber data. Based on graph kernels, Wu and Liu[13] proposed an SVM classifier, which can capture the semantic features, such as topic and sentimental data, as well as higher-order propagation patterns. Sampson et al.[15] proposed a method to discover the implicit link between rumors in a conversation. Ma et al.[16] proposed a kernel-based method that compared the spreading of rumors to a tree structure to distinguish between different types of higher order rumors. Wu et al.[17] utilized a long short-term memory (LSTM) RNN that could classify and represent the propagation routes of a message. Ma et al.[18] proposed a CNN with user-attention-based methods. Liu and Wu[19] implemented RNNs and CNNs to capture the global and local variations of user characteristics along propagation paths to detect fake news.

Social rumor detection based on the rumor texts and comments can be performed by NLP. Ma et al.[2] proposed a method based on RNNs. Subsequently, various deep learning-based rumor detection methods were introduced. A deep learning approach can learn from the simple inputs of contexts and changes in the content of hidden representations. For detecting rumors in Chinese microblogs, Lin et al.[3] proposed a sequence model with RNNs, wherein falsity and influence were the two key factors in each rumor. To better understand a rumor, their model captured the bi-directional sequence content. Ruchansky et al.[20] combined two features including the article text and user response to improve the accuracy of rumor detection. Chen et al.[21] presented a model with deep attention RNNs to acquire the temporal features of continuous comments. Guo et al.[22] combined social information into a novel hierarchical neural network. Yu et al.[23] proposed the use of paragraph embedding as input to CNNs to learn the representation of group posts in specific events. Ma et al.[24] and Li et al.[25] adopted a multitask learning model, where all tasks shared and LSTM layer along with some specific task layers. Jin et al.[26] and Wang et al.,[27] respectively, combined LSTM and VGG19 with text and image features, and proposed multimodal fusion methods. These methods focused on the text features and image. Lin et al.[28] introduced hierarchical attention mechanism into recurrent convolution neural networks (RCNN) for researching. Ma et al.[29] borrowed the concept of generative adversarial networks,

and compensated for the lack of data by generating rumors. Combing sentimental reactions, Wang and Guo[30] proposed a two-layer cascaded gated recurrent unit for detecting rumors. To obtain the textual information from fake news, Zhang et al.[31] introduced graph neural network to extract the features. Wang et al.[32p]ropose a knowledge-driven model by combining the textual information, knowledge concepts, and visual information. To classify the reliability of small messages on fake news, Schwarz et al.[33u]sed an end-to-end framework.

Different from the aforementioned methods that extensively employed RNNs, we propose models based on transformer encoding blocks with a multihead self-attention mechanism and no recurrences. We hypothesize that our model can capture the global semantic features from the context information and ignore the order in which comments are posted for rumor detection. We assume that our model can perform faster executions that other RNN-based rumor detecting models.

3 | MODEL

3.1 | Problem statement

Rumor detection involves bi-classification tasks. Given some events, each event contains a series of related posts, comments, and retweets. We combined all relative source posts, reposts, or comments with the original event text as an event and identified whether an event is a rumor. We established an event set $X = x_1, x_2, \ldots, x_n$ and label set $Y = y_1, y_2, \ldots, y_n$ as the data source to train the proposed model to detect whether an unlabeled event is a rumor.

3.2 | Model structure

The model proposed in this study is shown in Figure 3, wherein the bottom to top of each subfigure consists of embedding, feature extractor and feature fusion, and classifier. Figure 3(A) represents a model with one transformer encoding block, Figure 3(B) represents a model with three transformer encoding blocks, and Figure 3(C) represents a model with three transformer encoding blocks and one LSTM layer. We replaced the LSTM layer in Figure 3(C) with a fully connected layer (FCN) to obtain Figure 3(D). Details of the model are described below.

3.2.1 | Word embedding

Word2vec is a type of immutable word embedding method, implying that there is only one vector for one word, even in different contexts. For example, in Figure 4, the Chinese word "eat" appears twice in one sentence, but its word vectors are the same. It is not reasonable for a word to use the same word vector in different contexts, because it is not appropriate for obtaining contextual representations.

3.2.2 | Positional embedding

To provide a model with the sequence without complex recurrence or convolution, positional embedding is also utilized with word embedding in the first transformer encoding layer. To ensure that positional embedding can be summed with word embedding, its dimensions are the same as those of word embedding. The sine and cosine functions of different frequencies can be formulated as follows:

$$\begin{cases} \text{PositionEcoding}_{(\text{pos},2i)} = Sin\left(\frac{\text{pos}}{10{,}000^{\frac{2i}{d_{\text{modal}}}}}\right) \\ \text{PositionEcoding}_{(\text{pos},2i+1)} = Cos\left(\frac{\text{pos}}{10{,}000^{\frac{2i+1}{d_{\text{modal}}}}}\right) \end{cases}. \tag{1}$$

In each event, let *pos* be the desired position in a rumor or nonrumor event. The dimension of Word2vec is d_{modal} for positional embedding; i is one dimension of positional embedding d_{modal}. The even i of the positional embedding of each word is injected with a sine variable, and the odd i is injected with a cosine variable to fill the positional embedding of each word. After the implementation of Formula 1, the same word in a sentence can obtain different positional embeddings. As shown in Figure 4, instances of the Chinese word "eat" have different positional embeddings; therefore, the results are different after adding with the same word encoding. Next, we add the positional and word embeddings of each word for each event for the input of the transformer encoding block. The input of the first transformer encoding block can be formulated as:

$$\text{Input} = \text{Word Embedding} + \text{Positional Embedding}. \tag{2}$$

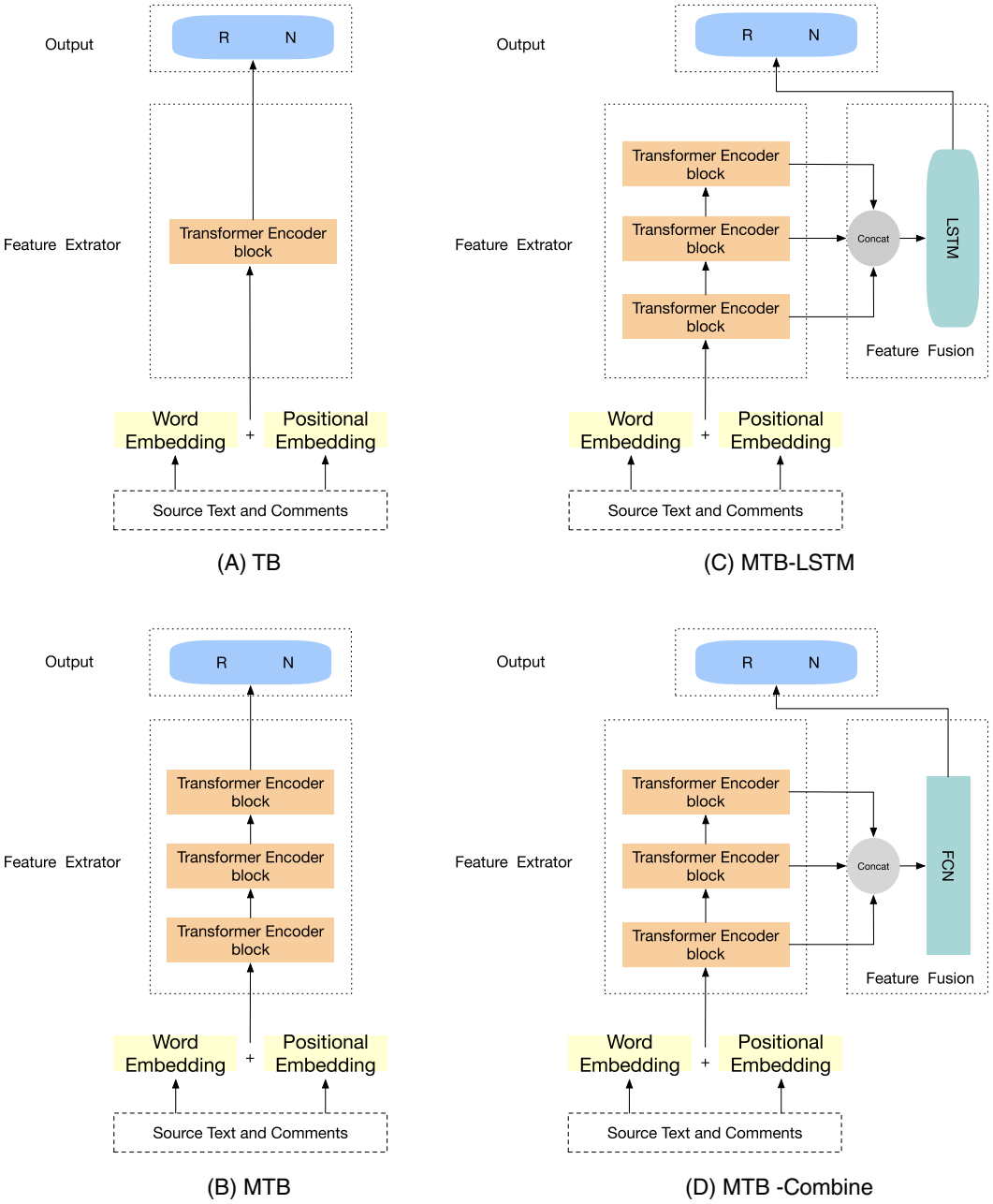


FIGURE 3 (A) MTB with one transformer encoding block. (B) MTB with three transformer encoding blocks. (C) MTB-LSTM with three transformer encoding blocks and one-layer LSTM. (D) MTB-Combine with three transformer encoding blocks and one FCN. FCN, fully connected layer; LSTM, long short-term memory

3.2.3 | Feature extractor

To capture the global representation of all comments and posts without considering the feature order, we adopted transformer encoding blocks as a feature extractor. The structure of a transformer encoding block is shown in Figure 5. A transformer encoding block is mainly composed of a multihead self-attention mechanism, normalization, feed-forward network (FFNs), followed by normalization, again. The number of blocks is a hyperparameter of the model. Different transformer encoding blocks can extract different features. For instance, the transformer encoding blocks in Figure 3(A) with only one layer can only extract the information related to semantic features from the shallow layer, while those in Figure 3(B) with three layers can extract the semantic features of the deep layer. Figure 3(C,D) obtain features from the shallow and deep layers. The multi-head self-attention mechanism shown in Figure 6 is the core module of each transformer encoding block. Several heads perform the self-attention operation in parallel according to different weight matrices.

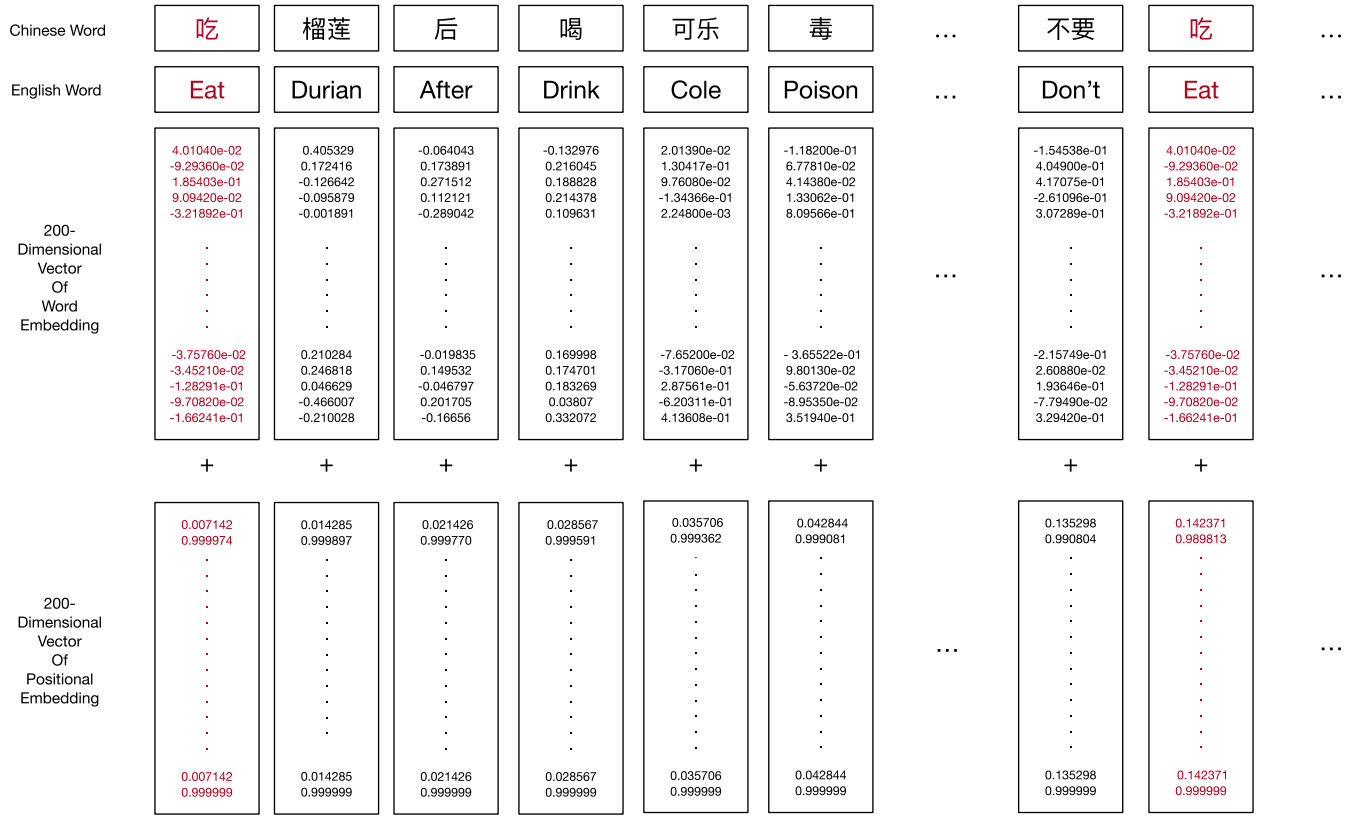


FIGURE 4 Word2vec of a rumor

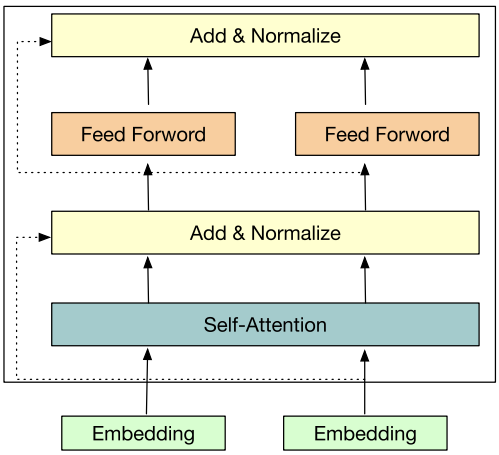


FIGURE 5 Structure of a transformer encoding block

3.2.4 | Self-attention mechanism in transformer encoding blocks

A rumor event is composed of the original text, posts, reposts, and comments. The time of posting a comment plays a significant role in determining the attitude (positive or negative) of people toward a post based on previous comments. This indicates that comments posted at a particular moment may influence the comments posted at long intervals. The integration of interactions among the semantic features in comments and posts seems to be an important factor in rumor detection. Self-attention mechanism is used to capture information related to the interaction features of the context. In the self-attention mechanism, each word can operate with all words in the surrounding context. Thus, there exists a dependency in the information between any two words. The flow of a head self-attention mechanism is shown in Figure 7.

The embedding part yields the input of the first transformer encoding block. The input can be a sequence of vectors Input $\in \mathbb{R}^{N\times D}$, where D is the dimension of embedding and N is the sequence length of the events. There are three matrices, namely, the query matrix Q, the key matrix K,

FIGURE 6 Multihead self-attention

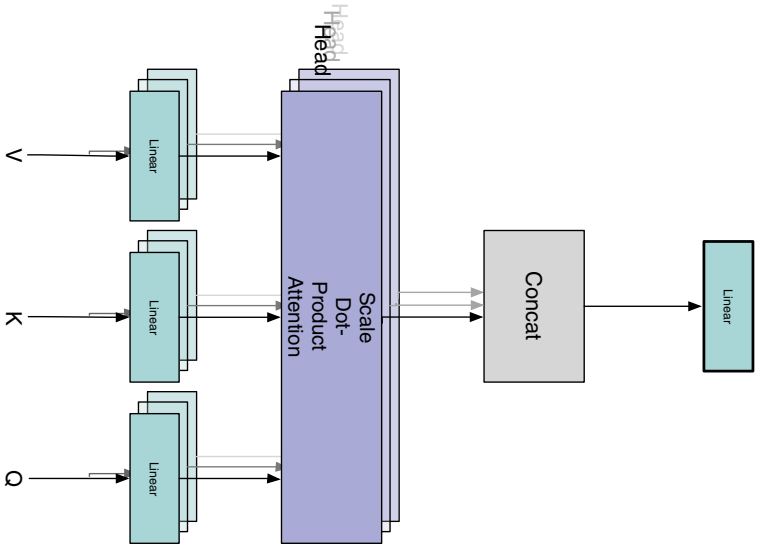
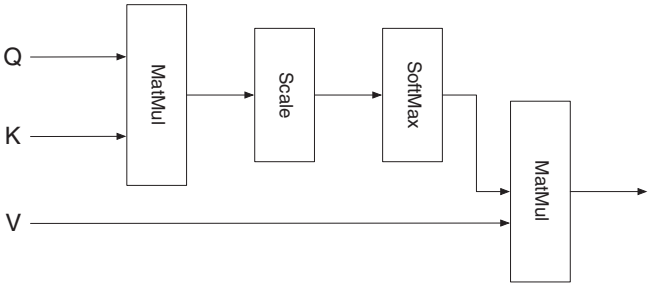


FIGURE 7 Self-attention mechanism



and the value matrix V. These three matrices are the projections of Input. The projection formula can be defined by Formula 7, wherein W^Q, W^K, W^T are the learnable parameters, which can be learned during training. We then decompose these three matrices into subspaces corresponding to the numbers of heads. The single-head self-attention process of comments and posts can be formulated as follows:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V, \tag{3}$$

$$Q = \text{Input}W^Q, \quad K = \text{Input}W^K, \quad V = \text{Input}W^V, \tag{4}$$

where d_k is the dimension of a vector in Q. Next, a self-attention head implements a new contextual information feature for the comments or posts. The multihead self-attention mechanism can help in extracting the interaction features from different subspaces. Thus, we combined this multihead self-attention mechanism with a transformer encoding block with H heads to obtain its semantic features.

$$\text{Multihead(Input)} = \text{Concat}(\text{Attention}_1, \text{Attention}_2, \ldots, \text{Attention}_H), \tag{5}$$

where the number of heads is H. Figure 8 illustrates a case of a self-attention mechanism with two heads. Each word can determine the relation between itself and other contextual words.

The output of multihead self-attention is given as input to the add and normalize block. The add operation helps in preventing the gradient from disappearing, which is the residual network solution.

$$\text{output} = \text{Multihead(Input)} + \text{Input}; \tag{6}$$

The normalized formula is:

$$\text{output} = \frac{\text{output} - \mu}{\sigma}, \tag{7}$$

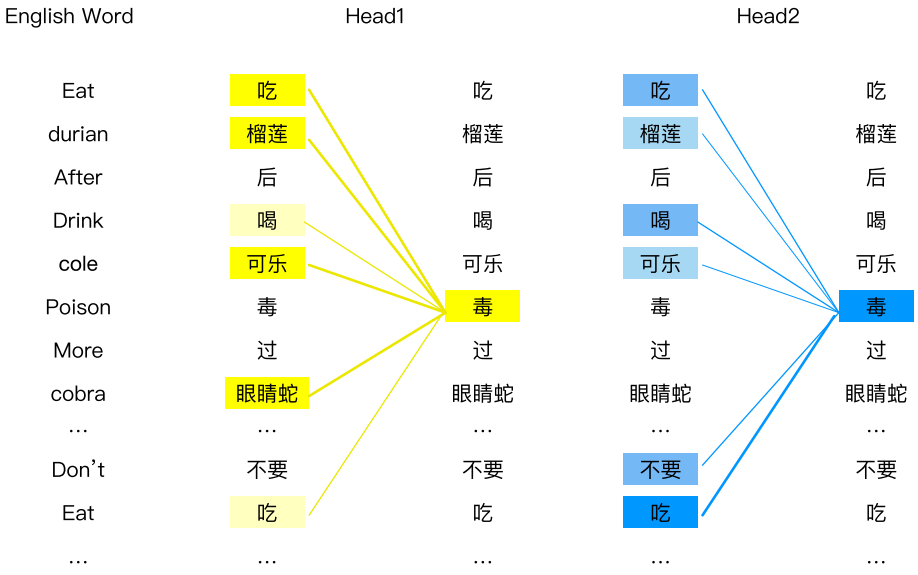


FIGURE 8 Example of self-attention mechanism with two heads

where μ is the mean and σ is the variance. The first normalization is followed by a FFNs, which can be represented by

$$\mathrm{FFN(output)} = \max(0, \mathrm{output}W_1 + b_1)W_2 + b_2. \tag{8}$$

There are two FFNs in a transformer encoding block to ensure that the dimensions of the output are similar to those of the input at the beginning. FFNs is equivalent to mapping the attention result of each position to a feature space of a larger dimension, followed by using rectified linear unit (ReLU) to introduce nonlinearity for filtering, and finally returning to the original dimension. The FFNs is again followed by an add and normalize block.

A K-layer transformer encoding block implies that the aforementioned operation is repeated K times. The output of the K_{i-1}th transformer encoding block is the input of the K_ith transformer encoding block. In Figure 3(A), $K = 1$, whereas in Figure 3(B–D), $K = 3$.

3.2.5 | Feature fusion

Here, the output of each transformer encoding block is combined. In comparison to Figure 3(B), we concatenated the outputs of the three layers to integrate the features of all transformer encoding blocks. The formula in Figure 3(C,D) can be given by

$$\mathrm{Output} = \mathrm{Output}_1 \oplus \mathrm{Output}_2, \ldots, \oplus \mathrm{Output}_k, \tag{9}$$

where Output_i represents the output of the ith layer. We adopted two methods in the proposed model. In Method 1, we directly concatenated the outputs into an FCN in Figure 3(D). In Method 2, we combined all Output_i and put them into an LSTM in Figure 3(C). An LSTM cell updates one memory cell, c_t, at time t. An LSTM unit output h_t is calculated by Reference 1

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i), \tag{10}$$

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f), \tag{11}$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C), \tag{12}$$

$$C_t = f_t * C_t - 1 + i_t * \tilde{C}_t, \tag{13}$$

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o), \tag{14}$$

$$h_t = o_t \tanh(C_t). \tag{15}$$

The amount of new memory added to the memory unit depends on the input gate i_t. In addition, the existing memory that is omitted depends on the forget gate f_t. The forgotten portion of memory $\tilde{C}_t$ updates the existing memory C_t and adds new memory. The output gate o_t represents the total amount of output memory. h_t denotes the output of the LSTM. In Figure 3(C,D), the output is obtained by Formulas 15 and 9, respectively.

3.3 | Loss function

The input of the FCN is the output of the transformer in TB, MTB, and MTB-Combine. The input of the FCN is the output of the LSTM in MTB-LSTM. The parameters in w and b are randomly removed while training to reduce overfitting in the model. We used a sigmoid as the activation function to output the prediction label Predict_i of an event. Under these constraints, the parameters were trained via back-propagation. The loss function can be expressed as:

$$\text{Predict}_i = \text{Sigmoid}(w^T \text{Output} + b), \tag{16}$$

$$\text{Loss} = -\frac{1}{N}\sum_{i=1}^{N}[y_i \ln \text{Predict}_i + (1 - y_i)\ln(1 - \text{Predict}_i)] + \frac{1}{2}\lambda(w^2 + b^2). \tag{17}$$

4 | EXPERIMENT

In this section, we compare the accuracy of rumor prediction, speed of model training, and early rumor detection to determine whether the proposed model in superior to the existing methods in these aspects.

4.1 | Dataset

The Sina Weibo dataset[22]. contains 4664 events and their corresponding tags; each event contains several tweets, and the specific content of each tweet has been provided completely in the original dataset. The attributes of the two datasets used in this study are listed in Table 1.

The PHEME dataset[3]of rumors and nonrumors is related to nine events, as the Weibo dataset presented in Table 1. Table 2lists the size of each event in the PHEME dataset along with the label distribution for the rumor detection task. The size of the events varies significantly, and, in general, the PHEME dataset contains fewer rumors than nonrumors.

4.2 | Experimental environment

This experiment was performed on a RedHat system with Intel(R) Xeon(R) e5-2620v4. We ran the Weibo and PHEME datasets on Nvidia Tesla P100 and Tesla K40m, respectively. The system comprised of 32 GB memory, and the framework was based on Python 3.7 and Tensorflow[4] 1.14. We used 75%, 10%, and 15% of the data for training, verification, and testing, respectively.

4.3 | Hyperparameter setting

In this study, the parameters set in literature[20] are followed. The hyperparameters were set as follows: the word vector dimension, learning rate, regularization coefficient, and number of units in the FCN were set to 200, 0.001, 0.01, and 100, respectively; moreover, the Adam optimizer was implemented. For our model presented in Figure 3(C), there were 100 hidden units for the LSTM. ReLU was used as the activation function in the FFNs of the transformers; it has become a major unit that can provide nonlinear transformations. The probability of neurons becoming inactive depends on the dropout rate, that is, dropout rate = P; therefore, the number of permutations and combinations of all neurons will be maximized when dropout rate = 0.5, which indicates that more randomness will be introduced.

[2]http://alt.qcri.org/wgao/data/rumdect.zip
[3]https://figshare.com/articles/PHEME_dataset_for_Rumour_Detection_and_Veracity_Classification/6392078
[4]https://www.tensorflow.org/

TABLE 1 Attributes of two datasets

Attributes	Weibo	PHEME
Events	4664	6425
Rumors	2313	2402
Nonrumors	2351	4023
Reposts	3,805,656	10,534
Max posts	59,318	289
Min posts	10	12

TABLE 2 Information related to the PHEME dataset

Event	Threads	Tweets	Nonrumor	Rumor
Charliehebdo	2079	38,268	1621	458
Ebola Essien	14	226	0	14
Ferguson	1143	24,175	859	284
Germanwings Crashs	469	4489	231	238
Gurlitt	138	179	77	61
Ottawa Shooting	890	12,284	420	470
Prince Toronto	233	902	4	229
Putin Missing	238	835	112	126
Sydney Siege	1221	23,996	699	522
Total	6425	105,354	4023	2402

4.4 | Experiment setting

The proposed and the baseline methods were tested on the same dataset. The following baseline methods were selected:

- DT-Rank:[34] A rumor classification model used on statistical features that adopts a decision tree to classify rumors.
- DTC:[8] A decision tree classifier to model the information reliability.
- SVM-TS:[12] A linear SVM classifier with the time series structure, which models the changes in the social scene features according to the content, users, and propagation patterns, and extracts these features.
- CAMI:[23] It adopts the doc2vec method to obtain the text representation, constructs a feature matrix of microblog events, and uses CNNs to learn the implicit features of microblog events to perform classification.
- SVM-RBF:[9] Uses manual features such as the statistics of comments, and then performs classification using an SVM.
- RFC:[10] It uses three parameters to fit the time curve of comments, and performs classification based on the random forest method.
- GRU-2:[2] Two-layer GRU is adopted to obtain the hidden representation of each microblog event and further classify these microblog events.
- CSI:[20] The hidden layer representation feature of microblog events by LSTM and global user features is combined with an singular value decomposition to perform classification.
- RCNN:[28] RNNs and CNNs are combined to represent the context information.
- BiLSTM-Attention[21] One-layer LSTM with soft attention.

4.5 | Evaluation index

To assess the effectiveness of our models and compare them with previous works, the following evaluation indicators were adopted.

TABLE 3 Confusion matrix

Predicted actual	Rumor	Nonrumor
Rumor	True rumor	False rumor
Nonrumor	False nonrumor	True nonrumor

TABLE 4 Weibo experimental results

		Rumor			Nonrumor		
Method	**Accuracy**	**Precision**	**Recall**	**F1**	**Precision**	**Recall**	**F1**
DT-Rank	0.732	0.738	0.715	0.726	0.726	0.749	0.737
SVM-RBF	0.818	0.822	0.812	0.817	0.815	0.824	0.819
DTC	0.831	0.847	0.815	0.831	0.815	0.847	0.830
RFC	0.849	0.786	0.959	0.864	0.947	0.739	0.830
SVM-TS	0.857	0.839	0.885	0.861	0.878	0.830	0.857
GRU-2	0.910	0.876	0.956	0.914	0.952	0.864	0.906
CAMI	0.933	0.921	0.945	0.933	0.945	0.921	0.932
CSI	0.953	0.930	**0.976**	0.954	**0.977**	0.931	0.953
TB	0.955	**0.988**	0.927	0.956	0.929	0.982	0.955
MTB	0.950	0.948	0.947	0.947	0.963	0.945	0.954
MTB-LSTM	0.959	0.966	0.956	0.961	0.951	0.968	0.959
MTB-Combine	**0.964**	0.984	0.946	**0.965**	0.945	**0.985**	**0.964**

Bold values are best in each column.

- Accuracy: It denotes the prediction accuracy of the ratio of an event to all events. The higher the accuracy, the better the classification performance. It can be denoted by Accuracy = (TR + TN)/(TR + FR + FN + TN).
- Precision: It denotes the precision of prediction of the ratio of a rumor (or nonrumor) event to all rumors (or nonrumors) events. It can be denoted by Precision = (TR)/(TR + FR).
- Recall: It denotes the actual prediction of the ratio of a rumor (or nonrumor) event to all rumor (or nonrumor) events. It can be represented by Recall = (TR)/(TR + FN).
- F1-score: Its calculation formula is $F1 = (2 * \text{Precision} * \text{Recall})/(\text{Precision} + \text{Recall})$.

The above evaluation indices can be understood more clearly through the confusion matrix, which is presented in Table 3.

4.6 | Experiment result

When the proposed models iterate to the 25th step, the value of loss tends to become stable and infinitely small, thereby proving the convergence of the loss function.

Experiments were performed using the Weibo dataset on DT-Rank, DTC, RFC, SYM-TS, SYM-RBF, GRU-2, CAMI, CSI model, and the model proposed in this study; the results are listed in Table 4,

For the baseline methods, the SVM-TS model achieved an accuracy of 85.7% in the Weibo dataset, which denotes the best result for a machine learning-based method. The deep learning-based CSI model achieved an accuracy of 95.3%, which is the best performance among the benchmark methods. These results demonstrate that the deep learning method is more accurate than the machine learning method. In comparison to the existing methods, the four models proposed in this study yielded excellent performance. This is because the transformer encoding blocks can efficiently extract the global semantic features. Comments that are posted after a significant time interval can significantly affect the source text and previous comments.

The values of Precision and Recall in TB for rumor and nonrumor events were found to be 0.988 and 0.982, respectively, which are the best scores of the evaluation. TB consisted of only one transformer encoding block; thus, it could only extract shallow information about the events,

TABLE 5 PHEME experimental results

		Rumor			Nonrumor		
Method	**Accuracy**	**Precision**	**Recall**	**F1**	**Precision**	**Recall**	**F1**
DT-Rank	0.562	0.588	0.421	0.491	0.549	0.704	0.617
DTC	0.582	0.582	0.573	0.578	0.579	0.588	0.584
SVM-TS	0.651	0.663	0.617	0.639	0.642	0.686	0.663
BiLSTM-Attention	0.814	0.734	0.789	0.761	0.868	0.828	0.847
CAMI	0.813	0.756	0.745	0.750	0.845	0.855	0.849
RCNN	0.819	0.742	**0.807**	0.773	0.876	0.829	0.852
TB	0.819	0.728	0.769	0.748	0.872	0.844	0.858
MTB	0.791	0.716	0.723	0.720	0.836	0.832	0.834
MTB-LSTM	0.806	**0.792**	0.720	0.754	0.817	**0.868**	0.841
MTB-Combine	**0.841**	0.776	0.793	**0.785**	**0.877**	0.867	**0.872**

Bold values are best in each column.

thereby producing unbalanced values. Thus, the values of Precision and Recall in TB for nonrumor and rumor events, respectively, were not significantly good. In MTB with high-level feature extraction, the values of Recall and Precision for rumor and nonrumor events increased owing to the presence of two transformer encoding blocks, while the high score (precision and recall in rumor and nonrumor events, respectively) reduced in the TB model to some extent. Thus, shallow features may be lost while transferring into top blocks, thereby producing relatively balanced values. These two models demonstrated that shallow features are as important as deep features, and their absence may yield inaccurate semantic features.

In MTB-LSTM and MTB-Combine, the features from the shallow, middle, and top layers were combined to ensure that they can efficiently reflect the semantic features of the different levels of transformer encoding blocks, and effectively prevent loss during the extraction of semantic features. Through the fusion of multilayered semantic features, the two models exhibited good performance. MTB-LSTM performed better than TB and MTB. In MTB-Combine, some scores decreased, but most of them increased in comparison to MTB-LSTM. As a type of RNNs, the LSTM inevitably lost some global features when fusing different layers. However, the LSTM and FCN can efficiently integrate the features of different levels.

For the PHEME dataset (Table 5), our methods were compared with three traditional methods and three deep learning methods. Evidently, the deep learning methods outperformed the traditional learning methods. We also performed an experiment using BiLSTM without attention that yielded poor performance; therefore, it has not been included in this study. RCNN exhibited the best performance in the deep learning methods; it achieved a Recall value of 0.807 for the rumor events. In our model, similar to the results of the Weibo dataset, several scores in MTB decreased to some extent. This proves that some shallow semantic features were lost during the transportation to higher layers. Therefore, it is necessary to combine the low- and high-level features. Although MTB-LSTM produced some good results, MTB-Combine yielded the best results in the comparative model.

4.7 | Time consuming

In this section, we discuss some experiments that were performed to compare the execution time of different methods during the training and testing phases. By calculating the training and testing speeds, the four proposed models were compared with four mainstream deep learning methods, including BiLSTM, BiLSTM-Attention, CAMI, and RCNN, on two datasets. Figures 9 and 10 show the results.

For each method on the Weibo and PHEME datasets, iterations were performed for 200 and 100 epochs, respectively (Table 5). The adoption of RNNs by the models would significantly decrease their speed. Using the Weibo dataset, MTB-LSTM required 203,156 and 19 s to train and test, respectively, which is the fastest and most accurate performance among the RNN-based models. Our other three models performed faster execution than all RNN-based models in the training and testing processes. Although MTB-Combine was slower than CAMI, it yielded the best accuracy score in both datasets.

It is evident from Table 1 that the Weibo dataset is significantly larger that the PHEME dataset. The RNN-based model runs 1.5 and 6 times longer on the PHEME and Weibo datasets in comparison to our model. Thus, by comparing the results of the two datasets, it can be concluded that as the data size increases, the proposed model provides greater advantages.

By comparing the four proposed models, it was found that TB has better accuracy, while it consumes half the time required by MTB and MTB-Combine. MTB-LSTM requires more time to reach an accuracy comparable to that of MTB-Combine. Thus, we can conclude that MTB-Combine has better speed and accuracy than others.

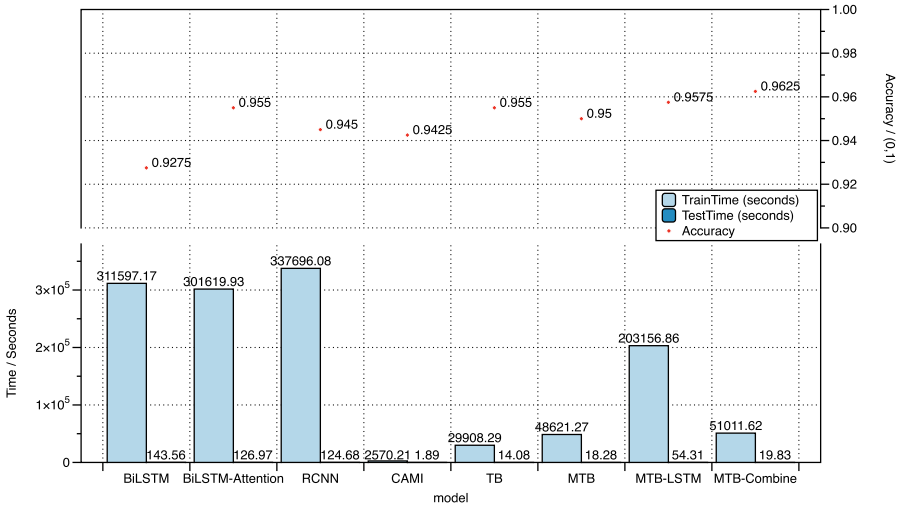


FIGURE 9 Train time, test time, and accuracy on the Weibo dataset

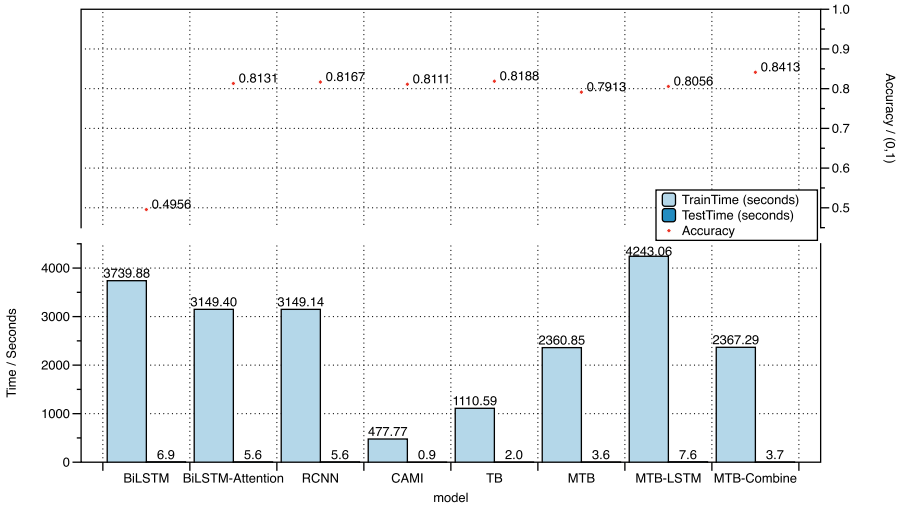


FIGURE 10 Train time, test time, and accuracy on the PHEME dataset

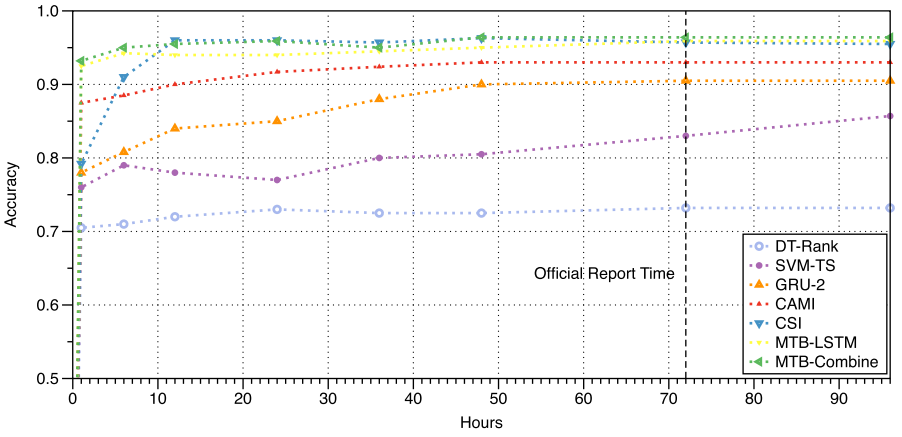


FIGURE 11 Accuracy on different time points of the Weibo dataset

4.8 | Early rumor detection

If a rumor is not detected during the early stages of its propagation, it will generate harmful effects. We evaluated eight time points of 1, 6, 12, 24, 36, 48, 72, and 96 h after an original microblog was posted. The method proposed in this study was compared with DT-RANK, SVM-TS, GRU-2, CAMI, and CSI models. These methods are introduced in Section 4.3.

Figure 11 demonstrates that the dependence of a model on the information generated in an event segment depends on the corresponding gradient; as the gradient rises, the dependence increases. It was found through experiments that MTB-Combine can achieve an accuracy of 92.5%

at 3 h, which is the best performance among all models. MTB-Combine also performed better than any other method. Simultaneously, the detection ability of MTB-Combine was found to be stable and it obtained the highest accuracy before the official release of information.

5 | CONCLUSION AND FUTURE WORK

The detection of rumors on social media has become a serious issue in recent years. Consequently, various studies adopted RNNs to detect rumors; however, there are many drawbacks in RNN-based methods. For example, RNNs consume more resources and time. In addition, they cannot capture the global features. To solve these problems, in this study, we proposed a social rumor detection method based on transformers; we utilized multilayer transformer encoding blocks to obtain implicit sequence representations from the contextual comments and posts on social media. Simultaneously, we implemented an LSTM and FCN for integrating the features of the information extracted from each layer. Using the Weibo and PHEME datasets, we compared the proposed model with some existing methods. The proposed model achieved a higher accuracy in comparison to other methods, thereby verifying its effectiveness in the detection of rumors on social media. In the early rumor detection task, our method exhibited good performance. It should be noted that by using fewer or no RNNs, our model performed faster executions during both the training and testing phases, which can help in saving time and resources.

Currently, rumors are also spread via images, that is, words are only a part of a rumor. The proposed method was based on semantic features, which are not always effective in identifying rumors. In future work, we plan to integrate the semantic features from transformer encoding blocks with image features to verify whether the introduction of image features can further improve our model.

ORCID

Lijun Lin https://orcid.org/0000-0002-3658-9326

REFERENCES

1. Hochreiter S, Schmidhuber J. Long short-term memory. *Neural Comput*. 1997;9(8):1735-1780.
2. Ma J, Gao W, Mitra P, et al. Detecting rumors from microblogs with recurrent neural networks. Paper presented at: Proceedings of the IJCAI. New York, NY; 2016:3818-3824.
3. Lin D, Ma B, Cao D, Li S. Chinese microblog rumor detection based on deep sequence context. *Concurr Comput Pract Exper*. 2019;31(23):e4508.
4. Vaswani A, Shazeer N, Parmar N, et al. Attention is all you need. *Advances in Neural Information Processing Systems*. 30. New York, NY: Curran Associates, Inc.; 2017:5998-6008.
5. Tang G, Müller M, Rios A, Sennrich R. Why self-attention? a targeted evaluation of neural machine translation architectures; 2018:4263-4272; EMNLP. arXiv preprint arXiv:1808.08946
6. Bondielli A, Marcelloni F. A survey on fake news and rumour detection techniques. *Inf Sci*. 2019;497:38-55.
7. Cao J, Guo J, Li X, Jin Z, Guo H, Li J. Automatic rumor detection on microblogs: a survey; 2018. arXiv preprint arXiv:1807.03505.
8. Castillo C, Mendoza M, Poblete B. Information credibility on twitter. Paper presented at: Proceedings of the 20th International Conference on World Wide Web; 2011:675-684; ACM, New York, NY.
9. Yang F, Liu Y, Yu X, Yang M. Automatic detection of rumor on Sina Weibo. Paper presented at: Proceedings of the ACM SIGKDD Workshop on Mining Data Semantics; 2012:13; ACM, New York, NY.
10. Kwon S, Cha M, Jung K, Chen W, Wang Y. Prominent features of rumor propagation in online social media. Paper presented at: Proceedings of the 2013 IEEE 13th International Conference on Data Mining, Dallas, Texas, USA; 2013:1103-1108; IEEE.
11. Liu X, Nourbakhsh A, Li Q, Fang R, Shah S. Real-time rumor debunking on twitter. Paper presented at: Proceedings of the 24th ACM International on Conference on Information and Knowledge Management; 2015:1867-1870; ACM, New York, NY.
12. Ma J, Gao W, Wei Z, Lu Y, Wong KF. Detect rumors using time series of social context information on microblogging websites. Paper presented at: Proceedings of the 24th ACM International on Conference on Information and Knowledge Management; 2015:1751-1754; ACM, New York, NY.
13. Wu K, Yang S, Zhu KQ. False rumors detection on sina weibo by propagation structures. Paper presented at: Proceedings of the IEEE 31st International Conference on Data Engineering, Seoul, South Korea; 2015:651-662; IEEE.
14. Kwon S, Cha M, Jung K. Rumor detection over varying time windows. *PloS One*. 2017;12(1):e0168344.
15. Sampson J, Morstatter F, Wu L, Liu H. Leveraging the implicit structure within social media for emergent rumor detection. Paper presented at: Proceedings of the 25th ACM International on Conference on Information and Knowledge Management, Indianapolis, IN, USA; 2016:2377-2382.
16. Ma J, Gao W, Wong K-F. Detect rumors in microblog posts using propagation structure via kernel learning. *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, Vancouver, Canada; Stroudsburg, PA: Association for Computational Linguistics; 2017:1;708-717.
17. Wu L, Liu H. Tracing fake-news footprints: characterizing social media messages by how they propagate. Paper presented at: Proceedings of the 11th ACM International Conference on Web Search and Data Mining; 2018:637-645; ACM, New York, NY.
18. Ma J, Gao W, Wong K-F. Rumor detection on twitter with tree-structured recursive neural networks. *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*, Melbourne, Australia; Stroudsburg, PA: Association for Computational Linguistics; 2018:1;1980–1989.
19. Liu Y, Wu YFB. Early detection of fake news on social media through propagation path classification with recurrent and convolutional networks. Paper presented at: Proceedings of the 32nd AAAI Conference on Artificial Intelligence, New Orleans, Louisiana, USA; 2018:354-361; AAAI Press.
20. Ruchansky N, Seo S, Liu Y. Csi: a hybrid deep model for fake news detection. Paper presented at: Proceedings of the 2017 ACM on Conference on Information and Knowledge Management; 2017:797-806; ACM, New York, NY.
21. Chen T, Li X, Yin H, Zhang J. Call attention to rumors: deep attention based recurrent neural networks for early rumor detection. Paper presented at: Proceedings of the Pacific-Asia Conference on Knowledge Discovery and Data Mining; 2018:40-52; Springer, New York, NY.

22. Guo H, Cao J, Zhang Y, Guo J, Li J. Rumor detection with hierarchical social attention network. Paper presented at: Proceedings of the 27th ACM International Conference on Information and Knowledge Management; 2018:943-951; ACM, New York, NY.
23. Yu, F., Liu, Q., Wu, S., Wang, L., & Tan, T. A convolutional approach for misinformation identification. *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence*, Melbourne, Australia; 2017. 3901-3907; AAAI Press.
24. Ma J, Gao W, Wong KF. Detect rumor and stance jointly by neural multi-task learning. Paper presented at: Proceedings of the International World Wide Web Conferences Steering Committee, Lyon, France; 2018:585-593.
25. Li Q, Zhang Q, Si L. Rumor detection by exploiting user credibility information, attention and multi-task learning. *Proceedings of the 57th Conference of the Association for Computational Linguistics*. Florence, Italy; Stroudsburg, PA: Association for Computational Linguistics; 2019:1;1173-1179.
26. Jin Z, Cao J, Guo H, Zhang Y, Luo J. Multimodal fusion with recurrent neural networks for rumor detection on microblogs. Paper presented at: Proceedings of the 25th ACM International Conference on Multimedia; 2017:795-816; ACM, New York, NY.
27. Wang Y, Ma F, Jin Z, et al. Eann: event adversarial neural networks for multi-modal fake news detection. Paper presented at: Proceedings of the 25th ACM international conference on Multimedia; 2018:849-857; ACM, New York, NY.
28. Lin X, Liao X, Xu T, Pian W, Wong KF. Rumor detection with hierarchical recurrent convolutional neural network. Paper presented at: Proceedings of the CCF International Conference on Natural Language Processing and Chinese Computing, Dunhuang, China; 2019:338-348; Cham, Springer.
29. Ma J, Gao W, Wong KF. Detect rumors on twitter by promoting information campaigns with generative adversarial learning. Paper presented at: Proceedings of the World Wide Web Conference; 2019:3049-3055; ACM, New York, NY.
30. Wang Z, Guo Y. Rumor events detection enhanced by encoding sentimental information into time series division and word representations. *Neurocomputing*. 2020;397:224–243.
31. Zhang J, Dong B, Philip SY. Fakedetector: effective fake news detection with deep diffusive neural network. Paper presented at: Proceedings of the 2020 IEEE 36th International Conference on Data Engineering, Dallas, TX, USA; 2020:1826-1829; IEEE.
32. Wang Y, Qian S, Hu J, Fang Q, Xu C. Fake news detection via knowledge-driven multimodal graph convolutional networks. Paper presented at: Proceedings of the 2020 International Conference on Multimedia Retrieval; 2020:540-547; ACM, New York, NY.
33. Schwarz S, Theóphilo A, Rocha A. EMET: embeddings from multilingual-encoder transformer for fake news detection. Paper presented at: Proceedings of the ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Barcelona, Spain; 2020:2777-2781; IEEE.
34. Zhao Z, Resnick P, Mei Q. Enquiring minds: early detection of rumors in social media from enquiry posts. Paper presented at: Proceedings of the International World Wide Web Conferences Steering Committee, Florence, Italy; 2015:1395-1405.

How to cite this article: Lin L, Chen Z. Social rumor detection based on multilayer transformer encoding blocks. *Concurrency Computat Pract Exper*. 2020;e6083. https://doi.org/10.1002/cpe.6083