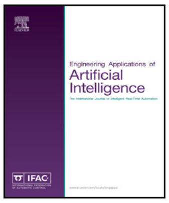
Engineering Applications of Artificial Intelligence 110 (2022) 104757

Contents lists available at ScienceDirect

Engineering Applications of Artificial Intelligence

journal homepage: www.elsevier.com/locate/engappai



SimCLRT: A Simple Framework for Contrastive Learning of Rumor Tracking

Hui Zeng [a,b], Xiaohui Cui [a,c,*]

[a] Engineering Research Center of Cyberspace, Yunnan University, Kunming 650091, China
[b] School of Software, Yunnan University, Kunming 650091, China
[c] School of Cyber Science and Engineering, Wuhan University, Wuhan, 430000, China

ARTICLE INFO

Keywords:
Rumor tracking
Contrastive learning
Text classification
Natural language processing
Deep learning

ABSTRACT

As the second stage of the rumor defeat task pipeline, rumor tracking aims to filter tweets related to a specific event. However, due to the different levels of attention of different events, events related to fewer tweets may be masked by related to more tweets. In some research, the researchers give up detecting the events containing a small number of tweets for better results. To this end, we propose a Simple Framework for Contrastive Learning of Rumor Tracking (SimCLRT)-a novel rumor tracking framework that uses contrastive learning to alleviate the cover between tweets. SimCLRT contains three variants SimCLRT-CNN, SimCLRT-Linear, and SimCLRT-RNN. We conduct experiments on the two commonly used rumor tracking datasets PHEME and RumorEval. The results show that SimCLRT completely defeated baselines. Like the detection performance on the events that contain many tweets, SimCLRT can also effectively detect events containing a small number of tweets. Furthermore, we compare and analyze the performance of SimCLRT variants. SimCLRT-CNN is the model that performs best in our experiments. Although SimCLRT-Linear has a slight advantage on the RumorEval dataset, its robustness is weaker than SimCLRT-RNN and SimCLRT-CNN. If in a long text environment, we consider SimCLRT-RNN will perform more competitively.

1. Introduction

With the development of social networks, massive amounts of data have been generated. People receive news, share opinions, and find communities on social networks. Since the entry barrier of social networks is only a phone number or mailbox, the rumor producer can easily hide on the social network. To gain attention, rumor makers often manipulate public events on social networks to promote the spread of controversial memes or fake news to compete with real information for people's attention. For example, with the coronavirus disease 2019 (COVID-2019) outbreak, many rumors about the "5G network causing COVID-2019" spread on the Internet. However, it turns out "5G network causing COVID-2019" is an excuse for conspiracy theorists to attack the 5G network. Rumors may deceive people's perceptions and hurt people's lives. Therefore, debunking the rumors on social networks is necessary to work.

Social networks are constantly generating data every minute. According to statistics, Facebook users will share 684,000 bits of content in one minute, and Twitter users will send more than 100,000 tweets. It is impractical to use manual methods to eliminate rumors from such a large amount of data. Therefore, many ways of automatically detecting rumors have been proposed by researchers. The existing techniques for automatically eliminating rumors can be divided into feature-based and deep learning-based. The feature-based methods rely on feature selection and extraction, and feature construction consumes many human resources and time. The deep learning-based methods are data-driven, and the deep learning-based models can automatically learn corresponding features from the data. Overall, the performances of deep learning-based methods are better than feature-based methods.

As shown in Fig. 1, a typical rumor confrontation system includes four subtasks: rumor detection, rumor tracking, stance classification, and rumor veracity (Kochkina et al., 2018; Zubiaga et al., 2018). Rumor detection aims to determine whether a text on an online social network is a rumor (Zubiaga et al., 2017; Dong et al., 2019; Ma et al., 2019; Nguyen et al., 2019). The purpose of rumor tracking is to determine and filter text related to rumor (Qazvinian et al., 2011; Cheng et al., 2020; Hamidian and Diab, 2019). Stance classification judges the commenter's attitude towards a piece of information (Enayet and El-Beltagy, 2017a; Kochkina et al., 2017; Zubiaga et al., 2016). Rumor veracity tells users the authenticity of a post (Kochkina et al., 2018; Li et al., 2019; Kumar and Carley, 2019). Researchers have conducted extensive research on rumor detection, rumor tracking, and stance classification tasks. However, only a small amount of studies has explored the rumor tracking task.

The purpose of rumor tracking is to catch those tweets related to a given event. Inspired by the structure of tweets and the work of Zhang et al. (2021), we propose a new rumor tracking method by introducing

* Corresponding author at: Engineering Research Center of Cyberspace, Yunnan University, Kunming 650091, China.
E-mail address: 00030460@whu.edu.cn (X. Cui).

https://doi.org/10.1016/j.engappai.2022.104757
Received 5 November 2021; Received in revised form 17 January 2022; Accepted 7 February 2022
Available online 19 February 2022

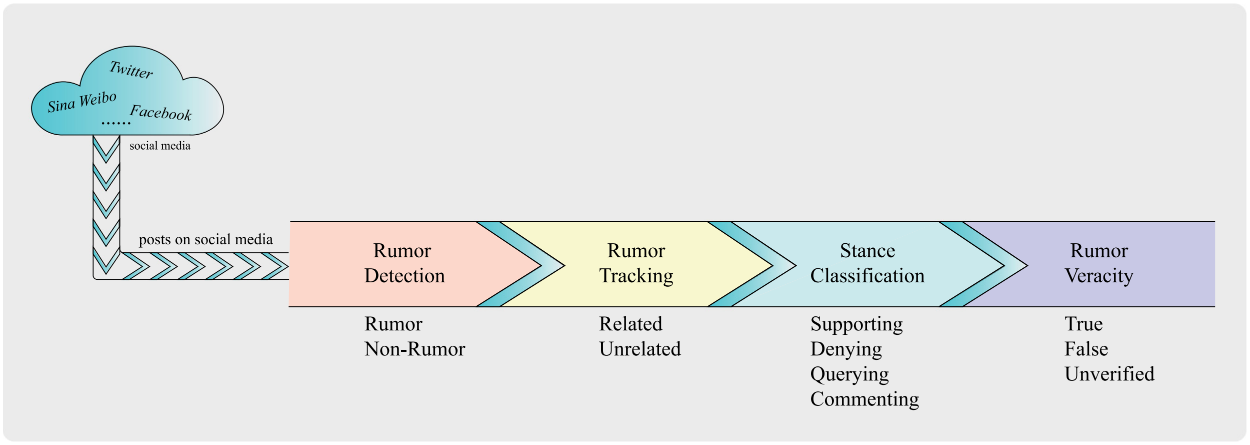


Fig. 1. The pipeline of a rumor defeat system.

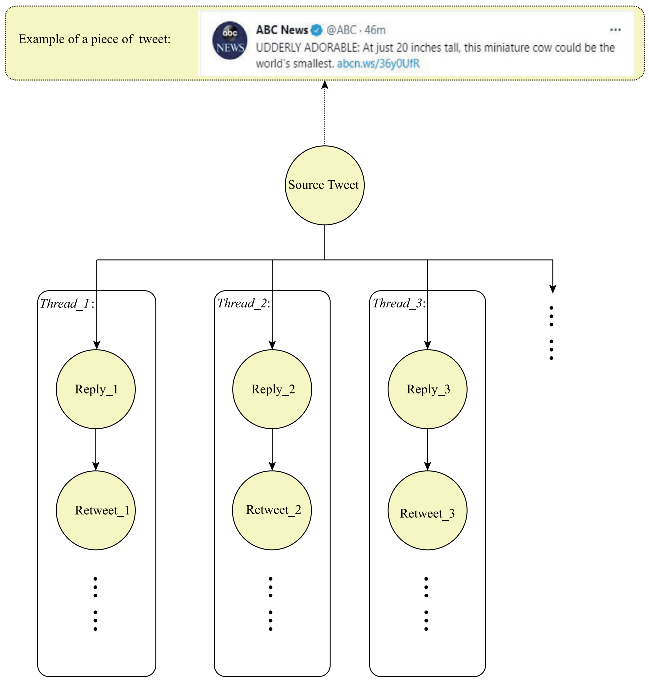

Example of a piece of tweet:
ABC News @ABC · 46m
UDDERLY ADORABLE: At just 20 inches tall, this miniature cow could be the world's smallest. abcn.ws/36y0UfR
Source Tweet
Thread_1:
Thread_2:
Thread_3:
Reply_1
Reply_2
Reply_3
Retweet_1
Retweet_2
Retweet_3

Fig. 2. The format of a tweet data.

contrastive learning. As shown in Fig. 2, the structure of tweets is a tree that uses source tweets as the root nodes, and each thread is a series of tweet sequences containing replies and retweets. We make two intuitive assumptions: first, we believe that tweets related to the same event have an inner connection; second, tweets related to different events may be hidden from each other, so the classifier cannot distinguish them effectively. For example, the tweets associated with a hot event may be obscure the tweets related to unpopular events.

Contrastive learning aims to distinguish whether two instances are sampled/augmented from the same source data. If so, let them be closer in the representation space; if not, let them be farther away in the representation space. In the work of Zhang et al. (2021), based on contrastive learning, a simple and effective unsupervised clustering model SCCL was proposed. The distance-based clustering method is used in the SCCL, where distinguish different categories by instance-wise contrastive learning. Through joint optimization, the Instance-wise contrastive learning loss and the clustering loss make the model keep the distance within the class closer while expanding the distance between classes.

Combining the two assumptions and the contrastive learning, we proposed SimCLRT — a novel rumor tracking framework which leverages contrastive learning to alleviate the hide between tweets. SimCLRT includes a feature extractor, and the features automatically extracted by the feature extractor are not interpretable. Therefore, we design three different variants of SimCLRT based on the fully connected neural network (FCNN), convolutional neural network (CNN), and recurrent neural network (RNN) to explore the performance of SimCLRT from multiple angles. SimCLRT variants based on CNN, Linear and RNN are called SimCLRT-CNN, SimCLRT-Linear and SimCLRT-RNN, respectively.

We use SimCLRT to distinguish tweets corresponding to different events, and then we jointly optimize the Instance-wise contrastive learning loss and classification loss. The experimental results show that our model has achieved state-of-the-art results on multiple models. Compared with the method that does not use contrast learning, events with fewer corresponding tweets have also received equal attention in our model. Meanwhile, the effectiveness of SimCLRT also proves that our hypothesis is valid. That is, tweets related to different events will be hidden from each other. Finally, we compare and analyze the performance differences between SimCLRT variants. To summarize, our contributions are as follows:

- We propose a novel end-to-end framework SimCLRT for rumor tracking. Compared with baselines, SimCLRT can achieve the best results on detecting events with a large number of tweets and effectively filter the events with a small number of tweets. Additionally, the performance of SimCLRT proves our assumption that tweets related to different events will hide from each other.
- We design multiple variants for SimCLRT and compare the performance between variants. The overall performance of SimCLRT-CNN is the best because SimCLRT-CNN can use convolutional networks of different sizes to obtain richer information from the feature generator. Although SimCLRT-Linear has a slight advantage on one of the datasets, its robustness is the worst. On the contrary, SimCLRT-RNN does not achieve the best results on any dataset, but its robustness is stable. We think SimCLRT-RNN is more competitive when facing long texts.
- We are valid the performance of SimCLRT on various data augmentation methods just as Zhang et al. (2021) verify the effectiveness of SCCL on different text augment methods. We improve how the back translation augment method generates instances

to increase the diversity of augmenting examples. We find that all the data augmentation methods used are beneficial for rumor tracking.

2. Related work

2.1. Rumor tracking

As shown in Fig. 1, rumor tracking is the second stage of a rumor defeat system. Its purpose is to filter posts related to rumors that have been identified (Zubiaga et al., 2018). In the existing literature, research on rumor tracking is scarce. In the work of Qazvinian et al. (2011), they manually annotated a dataset of more than 10,000 tweets related to 5 different rumors, and each tweet was marked as related or unrelated to the rumors. They used regular expressions to obtain data from the API provided by tweets roughly, so the tweets collected have similar features, such as overlapping keywords, which is challenging for classification tasks. Based on the labeled dataset, Qazvinian et al. (2011) constructs different Bayesian classifiers to extract different high-level features from the dataset for supervised machine learning. The features they used include "content-based features", "network-based features", and "Twitter specific memes". Specifically, content-based features include unigrams, bigrams and part-of-speech tags (POS) of tweets. Network-based features are RTs, which represent the user's transfer behavior. Twitter specific memes include hashtags and URLs for tweets. Finally, content-based features achieved the highest average precision and recall. Following the work of Qazvinian et al. (2011), Hamidian and Diab (2016) proposed Tweet Latent Vector to create a latent vector representation for each Tweet to overcome the missing word and short-length tweet issue. The creation of Tweet Latent Vector relies on the STS model proposed by Guo and Diab (2012). The STS model uses WordNet, Wiktionary, and Brown clusters to create a reliable latent variable model for the sentence. The latent variable produce by STS can make up for the lack of semantic information of the sentence.

In the work of Cheng et al. (2020), they created a multi-task learning framework called VRoC based on a variational autoencoder (VAE) and a recurrent neural network with long short-term memory network (LSTM), which implemented the coordinated training of rumor detection, rumor tracking, stance classification, and rumor veracity tasks. The backbone of VRoC is variational autoencoder(VAE), so as same as the standard variational autoencoder, VRoC also includes an encoder and a decoder. The encoder uses a series of tweets as input and then extracts tweet-level latent features through an encoder network to compress and represent the information in the text. The decoder uses the features extracted by the decoder to reconstruct the input text and ensures that the features extracted by the encoder network are valid. While using the decoder to decode and reconstruct the text, VRoC uses the features extracted by the encoder to input the four rumor classification components respectively. The four rumor classification components have independent networks and loss functions, which can guide VRoC to tuning parameters in a direction that is conducive to classification. Different from Qazvinian et al. (2011), Hamidian and Diab (2016) and Cheng et al. (2020) that regard rumor tracking as a two-classification task, Li et al. (2021) considers rumor tracking to be a multi-classification task, that is, the tweets are specifically related which rumor. Inspired by Mixture of Experts model (MoE) (Miller and Uyar, 1997), Li et al. (2021) proposed a deep reinforcement learning based ensemble model (RL-ERT) for rumor tracking. First, they selected multiple deep learning-based and non-deep learning-based models as the basic components of RL-ERT and constructed corresponding features for the components respectively on the tweets. The output of the basic components is the probability distribution of the rumor category about tweets. Then, they trained each basic component separately and retained the probability distribution of the rumor category about tweets learned by the basic component. Finally, they designed a CNN-based weight adjustment network WTPN to assign suitable weights for the retained results of basic components learning. The weighted sum of the probability distribution of the rumor category learned by the basic components as the predicted result of RL-ERT.

2.2. Contrastive learning

The lack of labeled data is one of the critical limit factors in the development of supervised learning. Some scholars consider self-supervised learning to be the future of deep learning development because it can reduce the cost of annotating datasets. As a paradigm of self-supervised learning, contrastive learning gets the most focus in the field of self-supervised learning, recently. In the field of computer vision and natural language processing, some works based on contrastive learning have achieved state-of-the-art results at the time on specific tasks, such as MoCo (Vaswani et al., 2017), SimCLR (Reimers and Gurevych, 2019) and SimCSE (Johnson and Zhang, 2017). Some works try to improve the performance of the text classification model by introducing contrastive learning, a paradigm of representation learning. Contrastive learning was used to learn noise-invariant representations for the Transformer-based encoders in the model proposed in Lai et al. (2015) for text classification tasks. Specifically, contrastive learning is used to close the distance of representations between clean examples and adversarial samples generated by perturbing the word embedding matrix of clean examples. In the CARL model proposed by Kim (2014), contrastive learning is used to reduce the distance of sentence-level semantic representation between the original samples and its adversarial samples while keeping samples with different labels away from each other.

3. Model

3.1. Problem statement

Most of the past works treat rumor tracking as a binary classification task, and their goals are to filter the tweets related to rumor events. We think treating the rumor tracking task as a binary classification problem is rough. In our work, we consider rumor tracking as an m-way classification problem to achieve more accurate tracking. Suppose $T = \{t_1, t_2, \ldots, t_n\}$ represents a series of tweets data sets, where $t_i \in T$ represents a piece of tweet information. Let $E = \{e_1, e_2, \ldots, e_m\}$ represent a set of rumor events, where $e_i \in E$ represents a certain rumor event. Each rumor event e_i has multiple tweets corresponding to it, and each tweet t_i belongs to only one rumor event. Given a tweet t_i, the purpose of rumor tracking is to discover the event e_i to which it belongs.

3.2. Model overview

Due to the policy of the tweet platform and the grammar rules of natural language, all tweets have similar format and structure features, i.e., the tweets related to rumors and the tweets un-related to rumors have similar shallow features. In addition, it is different in the number of tweets related to different rumor events. A certain rumor event contains more tweets, which may receive more attention from the model. In other words, rumor events with fewer tweets will be covered by rumor events with more tweets. The goal of contrastive learning is to distinguish whether two instances are sampled/augmented from the same original data, if so, let them be closer in the representation space; if not, let them be farther away in the representation space. Inspired by the comparative learning mechanism, we propose a simple rumor tracking framework to keep tweets related to different rumor events away from each other.

As shown in Fig. 3, our model consists of three components. First, a feature generator ψ maps the input data to the feature space. Then,

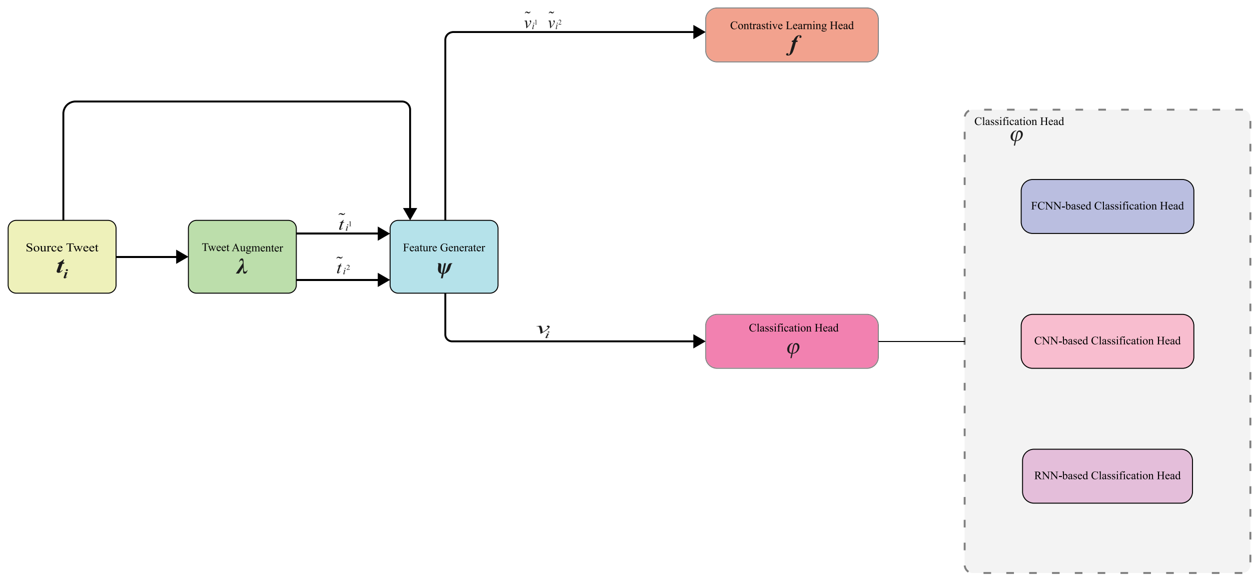

Source Tweet t_i
Tweet Augmenter λ
$\tilde{t}_{i^1}$
$\tilde{t}_{i^2}$
Feature Generater ψ
$\tilde{v}_{i^1}$ $\tilde{v}_{i^2}$
Contrastive Learning Head f
v_i
Classification Head φ
Classification Head φ
FCNN-based Classification Head
CNN-based Classification Head
RNN-based Classification Head

Fig. 3. The framework of SimCLRT. First, each source tweet t_i is sent to the Tweet Augmenter λ to generate an augment pair for it. Then, a BERT-based feature generator ψ to map the source tweets and augment pairs into the feature space. The features of the augment pairs are sent to the Contrastive Learning Head f, and the features extracted from the source tweets are sent to the Classification Head φ. We design three different structures for SimCLRT based on the fully connected neural network (FCNN-based), convolutional neural network (CNN-based), and recurrent neural network RNN(RNN-based). Finally, we jointly optimize the classification loss over the source tweets and the contrastive loss over the associated augment pairs.

the features extracted from the feature generator are fed two different heads $f(\cdot)$ and $\varphi(\cdot)$ where contrastive learning and classification learning are performed, respectively.

The input data includes original data and augmented data. We randomly generate a pair of augmented data $\tilde{t}_{i^1}$, $\tilde{t}_{i^2}$ for each original tweet t_i. Therefore, each mini-batch contains the original data $B = \{t_i\}_{i=1}^{M}$ of size M and the augment data $B^a = \{\tilde{t}_i\}_{i=1}^{2M}$ of size $2M$.

3.3. Feature generator

The feature generator ψ is used for mapping tweets from text to feature space. BERT and its variants have succeeded in many tasks in NLP. We use pre-trained Sentence-BERT (SBERT) as our feature generator. SBERT reduces the computational effort due to the siamese and triplet network structure design compared with BERT.

To enable the tweets and the augment pairs to be input into the pre-trained model, we need numerical Tweets and the augment texts. Specifically, to adapt the pre-training model's input format requirements, we add the special symbol *[CLS]* in front of each text and *[SEP]* in the back. Then, we do tokenization for the tweets and the augment texts in units of words. Finally, we use the vocabulary contained in the pre-training model to map the words to the corresponding ids to digitize Tweets and the augment pairs. The digital tweets and augment pairs can be accepted by the pre-trained SBERT.

SBERT is designed based on BERT, and the implementation of BERT is identical to the Transformer. Because the use of Transformer is well-known, we will omit the background description. We refer readers confused about SBERT to read Vaswani et al. (2017) and Reimers and Gurevych (2019). The parameters of the last hidden layer of SBERT learned from the source Tweets and the augment pairs will be sent to the classification head and the contrastive learning head, respectively. The work of the feature generator ψ can be expressed mathematically as Eq. (1),

$$\begin{cases} v_i = \psi(t_i) \\ \widetilde{v_{ij}} = (\psi(\widetilde{t_{ij}})), j \in \{1,2\} \end{cases}, \tag{1}$$

where t_i, $\widetilde{t_{ij}}$ represent a source tweet and its corresponding augments pair; v_i, $\widetilde{v_{ij}}$ represent the output of the feature generator corresponding to t_i and $\widetilde{t_{ij}}$, that is, the parameters of the last layer of the feature generator.

3.4. Contrastive learning head

Using contrastive learning needs to construct positive and negative examples at first. Suppose $\tilde{t}_{i^1}$, $\tilde{t}_{i^2} \in B^a$ are an augmented pair from the same instance in the original set B. We consider $\tilde{t}_{i^1}$, $\tilde{t}_{i^2} \in B^a$ as a positive pair, and regarding the other $2M-2$ instances in the B^a as corresponding negative examples. The contrastive learning head can produce corresponding outputs $\tilde{z}_{i^1}$, $\tilde{z}_{i^2}$ on the $\tilde{v}_{i^1}$, $\tilde{v}_{i^2}$ produced by the feature generator. The process is expressed by Eq. (2) formally,

$$\begin{cases} \tilde{z}_{i^1} = f(\tilde{v}_{i^1}) \\ \tilde{z}_{i^2} = f(\tilde{v}_{i^2}) \end{cases}. \tag{2}$$

In coding, we rely on the linear neural network and ReLU (Rectified Linear Unit) activation function to realize Eq. (2).

Then, for one positive instance in a certain pair, we use Eq. (3) to find the other one positive instance in the same pair from all the negative instances,

$$l_{i^1}^{I} = \frac{\exp(sim(\tilde{z}_{i^1}, \tilde{z}_{i^2})/\tau)}{\sum_{j=1}^{2M} I_{j\neq i^1} \cdot \exp(sim(\tilde{z}_{i^1}, \tilde{z}_j)/\tau)}, \tag{3}$$

here $I_{j\neq i^1}$ and τ denote the indicator function and temperature parameter, respectively. The definition of $sim(\cdot)$ is as Eq. (4), which represents the dot product between two tensors,

$$sim(\tilde{z}_i, \tilde{z}_j) = \frac{\tilde{z}_i^{\top}\tilde{z}_j}{||\tilde{z}_i||_2||\tilde{z}_j||_2}. \tag{4}$$

Finally, the loss on contrastive learning head is the average value over all instances in B^a,

$$L_{ct} = \frac{\sum_{i=1}^{2M} l_i^I}{2M}. \tag{5}$$

3.5. Classification head

We design multiple variants for the classification head to explore the best adaptation between the classification head and the feature generator. Specifically, we design three different architectures based on fully connected neural network(FCNN), convolutional neural network(CNN), and recurrent neural network(RNN), respectively. Let the representation of $t_i \in B$ in the feature space is $v_i = \psi(t_i)$. We use different classification heads to learn the probability distribution of t_i with respect to the event E.

3.5.1. FCNN-based classification head

Fully connected neural network sometimes also called linear layer, which transforms the input to the output with a specified dimension. At first, we design a classification head based on the fully connected network followed by the $softmax$ layer to calculate the probability distribution of t_i with respect to the event E,

$$p_c = soft\max(W^\top v_i + b), \tag{6}$$

where W and b are the parameters and bias of the network, respectively.

3.5.2. RNN-based classification head

RNN is one of the most widely used networks in the field of natural language processing, which can effectively capture long-distance dependencies in text. Long short-term memory(LSTM) is a variant of RNN, which solves the time-delay problem existing in traditional RNN. The details of LSTM are as follows:

$$\begin{aligned} C_t &= f_t \times C_{t-1} + i_t \times \tilde{C}_t \\ f_t &= \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \\ i_t &= \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \\ \tilde{C}_t &= \tanh(W_C \cdot [h_{t-1}, x_t] + b_C) \\ o_t &= \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \\ h_t &= o_t * \tanh(C_t), \end{aligned} \tag{7}$$

where x_t is the input at time step t, W_f, W_i, W_C, and W_o denote weight matrices, b_f, b_i, b_C, and b_o denote bias vectors. $\sigma(\cdot)$ represents the sigma function, and $\tanh(\cdot)$ denotes the hyperbolic tangent non-linear function. f_t, i_t, and o_t are called forget gate, input gate, and output gate. C_t and h_t denote the cell state and the hidden state of the current time step, which will continue to be passed backwards to achieve the ability to store long and short-term information on time series. The update of C_t and h_t are controlled by f_t, i_t, and o_t. In order to capture forward and backward information simultaneously, we use bidirectional long short-term memory(Bi-LSTM) as the RNN-based classification head.

3.5.3. CNN-based classification head

Convolutional neural networks use convolution kernels of different sizes to capture features in the text. The definition of two-dimensional convolutional neural networks is as follows:

$$\begin{aligned} out(N_i, C_{out_j}) = \sum_{k=0}^{C_{in}-1} weight(C_{out_j}, k) \star input(N_i, k) + \\ bias(C_{out_j}), \end{aligned} \tag{8}$$

where $\star$ is the valid 2D cross-correlation operator. N is a batch size. C denotes a number of channels. H and W are the height and width of the input planes, respectively.

Finally, all classification heads use cross entropy loss as the classification loss,

$$L_{cf} = -\frac{1}{N} \sum_i \sum_{c=1}^{M} y_{ic} \log(p_{ic}). \tag{9}$$

The overall loss of the model is the sum of the contrastive learning loss and the classification loss:

$$L = L_{ct} + L_{cf} \tag{10}$$

Table 1
PHEME.

Events	Tweets
Charlie Hebdo	38268
Ferguson	24175
Sydney Siege	23996
Ottawa Shooting	12284
Germanwings Crash	4489
Prince Toronto	902
Putin Missing	835
Ebola Essien	226
Gurlitt	179

3.6. Why and how SimCLRT is work?

This section will discuss why SimCLRT is effective and why we design three different variants for classification heads — SimCLRT-Linear, SimCLRT-CNN and SimCLRT-RNN.

The first factor in ensuring the effectiveness of SimCLRT is pre-training and fine-tuning. Deep learning is data-driven. However, the lack or imbalance of datasets limits the application of deep learning in many fields. There are currently very few datasets available for rumor tracking tasks, and they are almost unbalanced. Fortunately, the pre-training and fine-tuning mechanism can apply the features learned from the source dataset to the target dataset, which can alleviate the lack of target data and accelerate the convergence of the model. In the natural language processing field, BERT and GPT-3 have been great successes in many downstream tasks. Therefore, we design a pre-trained model based on BERT as our feature extractor, which allows us to use the features learned by the pre-trained model from the large dataset, alleviating the lack of rumor tracking dataset and speeding up the convergence of SimCLRT.

Here, we will discuss the motivation to set three different variants for the classification head. BERT can extract enrich features from datasets. However, the features extracted by BERT are not interpretable. The deep learning model learns information from data through forwarding propagation and backpropagation in training, and the learned information is saved in parameters. The learned information and adaptation occasions of the different models are various. In SimCLRT, the feature generator and the classification head interact. On the one hand, the features extracted by the feature generator will promote the classification head to achieve better results; on the other hand, the classification head will urge the fine-tuning of the feature extractor more adapt the classification head. We expect to explore the network structure that can more effectively adapt/utilize feature generators and datasets by designing different variants for classification heads to obtain better experimental results.

We design three different variants for the classification head of SimCLRT based on FCNN, CNN, and RNN because they are the three most widely used deep learning structures and have distinct characteristics. FCNN is the most straightforward neural network, usually used as a baseline and dimensional transformation. RNN and its variants are typically used to process temporal relationships by retaining the dependent information in the sequence and are widely used in natural language processing. CNN can keep local relevance and spatial invariance in forwarding propagation through local links, weight sharing, pooling, and multi-level structure.

To sum up, since (1) the features extracted by the feature generator are not interpretable, (2) the different networks have different strengths, (3) and the feature generator and the classification head are interactive, so we design three different variants for the classification head to explore which structure adapt better with the feature generator and the datasets.

Table 2
RumorEval.

Events	Tweets
Sydney Siege	1107
Ferguson	1084
Charlie Hebdo	1071
Ottawa Shooting	777
Germanwings Crash	281
Prince Toronto	103
Putin Missing	62
Ebola Essien	34

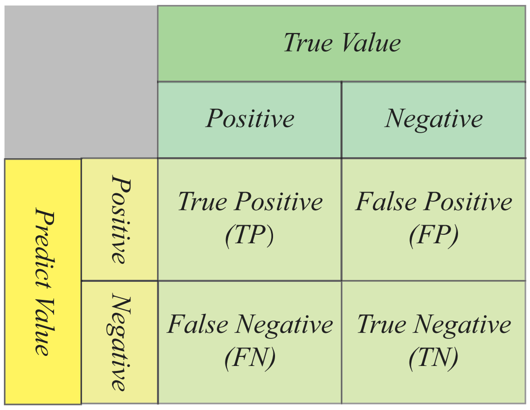


Fig. 4. Confusion Matrix.

4. Experiment settings

4.1. Dataset

PHEME (Kochkina et al., 2018) and RumorEval (Enayet and El-Beltagy, 2017b) are two benchmark datasets, which are a collection of tweets related to some rumored events. The tweets are organized in a tree structure. The source tweet is the root node, and a series of comments and retweets following the source tweet are used as branch nodes or leaf nodes (if there are no comments and retweets follow). Each tweet has many items, including user, id, text, etc. Table 1 and Table 2 list the events and the number of tweets contained in PHEME and RumorEval, respectively.

4.2. Evaluation metrics

In the experiment, we use precision, recall, and f1-score as our evaluation metrics. As shown in Fig. 4, examples in the dataset can be grouped into true positive (TP), true negative (TN), false positive (FP), and false negative (FN) based on the combination of the actual label of the example and the predictive label of the learner. The definition of precision is as follows:

$$precision = \frac{TP}{TP+FP}, \tag{11}$$

which reflects how many of the examples classified as positive by the classifier are true positive.

The recall is defined as follows:

$$recall = \frac{TP}{TP+FN}. \tag{12}$$

The recall reflects the proportion of examples labeled as positive by the classifier that occupy all examples with actual labels as positive.

However, precision and recall are a pair of contradictory metrics, when precision (recall) is higher, recall (precision) will be lower. The f1-score realize a well-balance between precision and recall. The definition of f1-score is as follows:

$$f1-score = \frac{2 * precision * recall}{precision + recall}. \tag{13}$$

4.3. Implementation

We introduce different methods of Tweet Augmenter in Section 4.5. We use *distilbert-base-nli-stsb-mean-tokens* as the backbone of Feature Generator. For the contrastive learning head, we optimize an MLP (g) with one hidden layer of size 768, and output vectors of size 128. For the FCNN-based classification head, we use a single-layer linear network in which output features numbers equal the event categories. For the RNN-based Classification head, we use a two-layer bi-directional LSTM with a hidden layer size of 256. We set the filter sizes of CNN-based Classification head to 2, 3, and 4 in order. We implement SimCLRT with the help of Pytorch[1] and the Sentence Transformer library (Reimers and Gurevych, 2019).

4.4. Baselines

To prove the effectiveness of SimCLRT in rumor tracking, we choose the following benchmarks as our comparison:

- SVM-RBF: SVM-RBF is a support vector machine classifier that uses radial basis function (RBF) kernel as the kernel function.
- DecisionTree: The decision tree modeling rumor tracking task as a simple decision process and learns the decision rules from the rumor data.
- RandomForest: Random Forest is an ensemble-based method. It will build multiple sub-trees on sub-samples, and then return an average voting result.
- GaussianNB: We use Gaussian Naive Bayes as one of our baselines, which assumes that the features obey the Gaussian distribution
- DPCNN: Deep pyramid convolutional neural network (DPCNN) is a word-level, extensive and effective deep text classification convolutional neural network (Johnson and Zhang, 2017).
- TextRCNN: TextRCNN uses a bidirectional RNN to replace the convolutional layer in the CNN network to capture the critical components of the text while capturing the context information as much as possible (Lai et al., 2015).
- TextCNN: Kim (2014) explored the effectiveness of CNN in text classification on multiple datasets.
- TextRNN: RNN is a traditional text classification model because it can effectively capture the contextual information in the text.
- Transformer: Vaswani et al. (2017) proposed the Transformer to solve the problem of machine translation and achieved the best results. The Transformer has been successful in natural language generation, text classification, and natural language understanding.

Unlike SimCLRT can extract the features of tweets directly, the baselines need to convert the source tweets to the embedding space manually. Therefore, we build the vocabularies for the PHEME and RumorEval datasets, respectively.

4.5. Data augmentation

We explore the performance of SimCLRT on three different unsupervised data augmentation methods, which borrow from the work of Zhang et al. (2021). The methods we used for data augmentation are as follows:

[1] https://pytorch.org/.

Table 3
The F1-Score on PHEME dataset.

Model	Augment Methods	Substitution Rate/Language Path	charliehebdo F1-score	ferguson F1-score	sydneysiege F1-score	ottawashooting F1-score	germanwings F1-score	prince F1-score	putinmissing F1-score	ebola-essien F1-score	gurlitt F1-score
SVM-RBF	–	–	0.879447	0.948012	0.869452	0.844322	0.777580	0.720379	0.693467	0.807018	0.941176
DecisionTree	–	–	0.760701	0.833819	0.700349	0.726035	0.718276	0.707819	0.616071	0.806452	0.888889
RandomForest	–	–	0.794480	0.868235	0.733962	0.771619	0.754717	0.701422	0.735294	0.827586	0.961538
GaussianNB	–	–	0.847501	0.888766	0.811892	0.745879	0.630303	0.661765	0.500000	0.567901	0.493506
DPCNN	–	–	0.870641	0.907500	0.833716	0.809329	0.765000	0.633205	0.491429	0.409091	0.744186
TextRCNN	–	–	0.888254	0.931399	0.845544	0.842893	0.790336	0.735426	0.620321	0.612245	0.826087
TextCNN	–	–	0.877583	0.925911	0.839141	0.823799	0.775685	0.724138	0.695652	0.785714	0.851064
TextRNN	–	–	0.889162	0.928108	0.841563	0.825482	0.772436	0.656250	0.555102	0.405405	0.784314
Transformer	–	–	0.920910	0.956581	0.896532	0.896787	0.862460	0.866667	0.823009	0.903226	0.875000
SimCLRT-CNN	ContextualAugmenter	10%	0.918612	0.954860	0.889679	0.894427	0.854518	0.896825	0.881057	0.969697	0.943396
SimCLRT-RNN	ContextualAugmenter	10%	0.915379	0.951541	0.884674	0.879668	0.834337	0.854962	0.854772	0.969697	0.943396
SimCLRT-Linear	ContextualAugmenter	10%	0.911837	0.953977	0.879934	0.880092	0.826970	0.813853	0.743961	0.875000	0.961538
SimCLRT-CNN	ContextualAugmenter	20%	0.916659	0.960985	0.884594	0.883845	0.857795	0.876494	0.826255	0.970588	0.943396
SimCLRT-RNN	ContextualAugmenter	20%	0.914033	0.953383	0.886983	0.882288	0.822556	0.863813	0.751773	0.970588	0.943396
SimCLRT-Linear	ContextualAugmenter	20%	0.906927	0.955694	0.879715	0.870025	0.830128	0.754545	0.738318	0.875000	0.961538
SimCLRT-CNN	ContextualAugmenter	30%	0.911205	0.956842	0.879153	0.887320	0.833579	0.870293	0.886076	0.985075	0.943396
SimCLRT-RNN	ContextualAugmenter	30%	0.912864	0.952717	0.884084	0.884046	0.840349	0.833948	0.864407	0.953846	0.943396
SimCLRT-Linear	ContextualAugmenter	30%	0.906927	0.955694	0.879715	0.870025	0.830128	0.754545	0.738318	0.875000	0.961538
SimCLRT-CNN	WordNetAugmenter	10%	0.920621	0.959517	0.893443	0.888029	0.858877	0.859504	0.860987	0.985075	0.961538
SimCLRT-RNN	WordNetAugmenter	10%	0.913999	0.952864	0.878666	0.876111	0.836473	0.866142	0.826446	0.914286	0.925926
SimCLRT-Linear	WordNetAugmenter	10%	0.911510	0.950446	0.874697	0.873926	0.823349	0.776786	0.740741	0.857143	0.961538
SimCLRT-CNN	WordNetAugmenter	20%	0.915391	0.955272	0.887763	0.890438	0.850345	0.878431	0.855856	0.969697	0.961538
SimCLRT-RNN	WordNetAugmenter	20%	0.913655	0.953082	0.883625	0.881459	0.839719	0.861538	0.805556	0.953846	0.925926
SimCLRT-Linear	WordNetAugmenter	20%	0.907715	0.952752	0.877718	0.874198	0.830601	0.794760	0.753488	0.870968	0.961538
SimCLRT-CNN	WordNetAugmenter	30%	0.919592	0.961952	0.894131	0.889135	0.853810	0.896825	0.844037	0.985075	0.943396
SimCLRT-RNN	WordNetAugmenter	30%	0.915671	0.955117	0.881531	0.878131	0.853125	0.795848	0.828452	0.914286	0.961538
SimCLRT-Linear	WordNetAugmenter	30%	0.909770	0.954265	0.883280	0.867829	0.848051	0.801762	0.746411	0.906250	0.961538
SimCLRT-CNN	Back translation	German–French	0.915260	0.955361	0.872258	0.883492	0.850080	0.810573	0.838710	0.920635	0.961538
SimCLRT-RNN	Back translation	German–French	0.909509	0.950239	0.873224	0.872225	0.824549	0.846774	0.742049	0.937500	0.925926
SimCLRT-Linear	Back translation	German–French	0.911050	0.946774	0.880842	0.874692	0.848532	0.786026	0.775862	0.923077	0.961538
SimCLRT-CNN	Back translation	French–Chinese	0.909046	0.957574	0.885470	0.879054	0.852824	0.869919	0.855856	0.969697	0.961538
SimCLRT-RNN	Back translation	French–Chinese	0.891995	0.943593	0.855356	0.866392	0.829195	0.849817	0.801688	0.878788	0.961538
SimCLRT-Linear	Back translation	French–Chinese	0.899349	0.938631	0.869748	0.849105	0.800967	0.755760	0.721154	0.875000	0.920000

- WordNet Augmenter[2]: WordNetAugmenter uses the synonyms that exist in WordNet to substitute the words of the source text (Morris et al., 2020; Ren et al., 2019).
- ContextualAugmenter[3]: ContextualAugmenter employs the top-n suitable words that adapt to the source text, found by pre-trained transformers, to substitute or insert. We use substitution to augment the source text and choose Bertbase and Roberta to generate the words for substitution (Ma, 2019).
- Back translation[4]: Back translation leverages the translation model to translate source text from the original language to another language (*intermediary language*) and then to the source language (Ma, 2019).

To explore the impact of different substitution ratios on the results of rumor tracking, we set the substitution ratios of WordNetAugmenter and ContextualAugmenter at 10%, 20%, 30%, respectively. For the augmentation way of back translation, we find that using different intermediary languages to generate augment pair is more helpful than only using one intermediary language to generate multiple candidates and then selecting a couple from the candidates as the augment pair. We use French&German, French&Chinese as two instances of back translation.

5. Results and analysis

5.1. Results

In the experiment, although we use precision, recall, and f1-score as our evaluation metrics, we can only put all the experimental results in Tables A.7 and B.8 in the Appendix for a proper typographic structure. Fortunately, F1-Score can reflect a well-balanced result based on precision and recall. Therefore, we can use F1-Score to discuss the performance of SimCLRT and baselines.

The column *Substitution Rate/Language Path* is the substitution ratio if the augment method is ContextualAugmenter or WordNetAugmenter, else it represents the intermediary language when the augment method is Back translation. Our SimCLRT outperforms all baselines on PHEME and RumorEval datasets.

It is worth noting that the ebola-essin and gurlitt events of the PHEME dataset, and the prince-toronto, putinmissing, and ebola-essien events of the RumorEval dataset, there are situations where multiple classifiers obtain the same experimental results. It is caused by the lack of tweets corresponding to the events. Li et al. (2021), Cheng et al. (2020) choose to discard the events that lack corresponding tweets to avoid the influence of the sparse category on the experimental results. We do not follow this setting, because the difference in the number of tweets corresponding to different events is an objective reality. The number of tweets involved in discussing an event depends on the spread scope of the event and the level of people's attention. People tend to participate in topics that relate to themselves in health, career, art, and so on. For example, both Ebolavirus and COVID-19 are infectious diseases with a high fatality rate, and even Ebolavirus has a higher fatality rate than COVID-19. However, the attention of the world to COVID-19 far exceeds that of Ebolavirus. It is because Ebolavirus is mainly distributed in Africa, and COVID-19 has surrounded the whole world. Therefore, due to differences in people's regions, occupations, interests, etc., the distribution of tweets related to different events must be different. In the research related to rumors, the researchers should positively respond to distribution imbalances from the model's perspective instead of destroying the actual distribution of data. We should ensure that the model is still highly detectable for events with a small number of tweets.

The events are sorted in descending order according to the number of related tweets left to right in Tables 3 and 4. On the dataset of PHEME, the F1-score of TextCNN and TextRNN from left to right in Table 3 shows a downward trend as the number of tweets corresponding to events decreases. Similarly, TextCNN, TextRNN, and Transformer have the same performance on the dataset of RumorEval in Table 4. Additionally, some abnormal fluctuations occurred in the last column of Tables 3 and 4, the number of tweets related to the event in the column is lesser than other columns. However, the F1-score is higher than some columns. Intuitively, we think the F1-score of the last column should be smaller than other columns, but it is not. We believe that the lack of training data causes the model's performance to decline with the decrease of the number of event-related tweets, while the lack of test data leads to the abnormal version of the last column. Luckily, our SimCLRT overcomes the two shortcomings. On the one hand, using data augmenters can alleviate the lack of data, which helps improve the model's performance. On the other hand, the weak position of a small amount of data in unbalanced classification is primarily caused

[2] https://github.com/QData/TextAttack.
[3] https://github.com/makcedward/nlpaug.
[4] https://github.com/makcedward/nlpaug.

Table 4
The result on RumorEval dataset.

Model	Augment Methods	Substitution Rate/Language Path	sydneysiege F1-score	ferguson F1-score	charliehebdo F1-score	ottawashooting F1-score	germanwings-crash F1-score	prince-toronto F1-score	putinmissing F1-score	ebola-essien F1-score
SVM-RBF	–	–	0.862637	0.940476	0.883582	0.878505	0.771429	0.500000	0.200000	0.571429
DecisionTree	–	–	0.849398	0.895238	0.790419	0.801653	0.810127	0.666667	0.888889	0.888889
RandomForest	–	–	0.856322	0.946708	0.829971	0.875576	0.846154	0.750000	0.800000	0.750000
GaussianNB	–	–	0.827988	0.863905	0.758170	0.775000	0.864198	0.769231	0.571429	0.750000
DPCNN	–	–	0.807843	0.821705	0.773913	0.713450	0.690909	0.303030	0.375000	0.666667
TextRCNN	–	–	0.858209	0.887029	0.825758	0.871166	0.838710	0.714286	0.200000	0.500000
TextCNN	–	–	0.827381	0.884956	0.834356	0.827586	0.825000	0.695652	0.615385	0.571429
TextRNN	–	–	0.764940	0.824074	0.689655	0.577778	0.330579	0.352941	0.000000	0.222222
Transformer	–	–	0.893491	0.951220	0.843077	0.852459	0.805556	0.800000	0.625000	0.750000
SimCLRT-CNN	ContextualAugmenter	10%	0.911765	0.956790	0.897690	0.912000	0.926829	0.800000	1.000000	0.888889
SimCLRT-RNN	ContextualAugmenter	10%	0.879765	0.930091	0.905063	0.895397	0.923077	0.888889	0.941176	0.888889
SimCLRT-Linear	ContextualAugmenter	10%	0.902077	0.957055	0.900000	0.920502	0.923077	0.928571	0.947368	0.888889
SimCLRT-CNN	ContextualAugmenter	20%	0.900585	0.950920	0.900958	0.941667	0.925000	0.888889	0.947368	0.888889
SimCLRT-RNN	ContextualAugmenter	20%	0.900302	0.956790	0.892966	0.933333	0.938272	0.888889	0.941176	0.888889
SimCLRT-Linear	ContextualAugmenter	20%	0.923977	0.954407	0.898089	0.932203	0.925000	0.846154	0.900000	0.888889
SimCLRT-CNN	ContextualAugmenter	30%	0.920821	0.968944	0.905063	0.940171	0.928571	0.928571	0.818182	0.888889
SimCLRT-RNN	ContextualAugmenter	30%	0.886364	0.954128	0.895899	0.929204	0.936709	0.888889	0.947368	0.888889
SimCLRT-Linear	ContextualAugmenter	30%	0.901163	0.960245	0.890282	0.923077	0.923077	0.846154	0.947368	0.888889
SimCLRT-CNN	WordNetAugmenter	10%	0.909639	0.959502	0.895062	0.921811	0.911392	0.928571	0.900000	0.888889
SimCLRT-RNN	WordNetAugmenter	10%	0.879310	0.948012	0.858044	0.923077	0.909091	0.846154	1.000000	0.888889
SimCLRT-Linear	WordNetAugmenter	10%	0.905882	0.960245	0.884735	0.914530	0.936709	0.888889	0.947368	0.888889
SimCLRT-CNN	WordNetAugmenter	20%	0.904348	0.971787	0.876623	0.882591	0.950000	0.928571	0.800000	0.888889
SimCLRT-RNN	WordNetAugmenter	20%	0.889535	0.954683	0.885246	0.907563	0.904762	0.846154	0.947368	0.888889
SimCLRT-Linear	WordNetAugmenter	20%	0.892754	0.956790	0.889590	0.909871	0.938272	0.928571	0.947368	0.888889
SimCLRT-CNN	WordNetAugmenter	30%	0.908555	0.947692	0.898089	0.917355	0.938272	0.846154	0.947368	1.000000
SimCLRT-RNN	WordNetAugmenter	30%	0.902208	0.954407	0.858824	0.914530	0.925000	0.888889	0.900000	0.888889
SimCLRT-Linear	WordNetAugmenter	30%	0.885057	0.957317	0.878981	0.909091	0.925000	0.928571	1.000000	0.888889
SimCLRT-CNN	Back translation	German–French	0.888889	0.975000	0.886076	0.898305	0.928571	0.838710	1.000000	0.888889
SimCLRT-RNN	Back translation	German–French	0.894895	0.921687	0.867089	0.895397	0.938272	0.846154	0.900000	0.888889
SimCLRT-Linear	Back translation	German–French	0.886297	0.960486	0.886792	0.935622	0.936709	0.888889	1.000000	0.888889
SimCLRT-CNN	Back translation	French–Chinese	0.920000	0.960000	0.885993	0.919831	0.963855	0.928571	0.941176	0.888889
SimCLRT-RNN	Back translation	French–Chinese	0.904762	0.942598	0.893082	0.902128	0.913580	0.785714	1.000000	0.888889
SimCLRT-Linear	Back translation	French–Chinese	0.906977	0.966967	0.896104	0.918455	0.975610	0.928571	0.947368	0.888889

Table 5
The mean and standard deviation of SimCLRT on the PHEME dataset.

Model		charliehebdo	ferguson	sydneysiege	ottawashooting	germanwings	prince	putinmissing	ebola-essien	gurlitt
SimCLRT-CNN	mean	**0.915798**	**0.957795**	**0.885811**	**0.886968**	**0.851478**	**0.869858**	**0.856104**	**0.969442**	0.952467
	standard deviation	*0.004026*	*0.002733*	0.007327	*0.004752*	*0.007881*	0.02724	0.020298	*0.021101*	*0.009697*
SimCLRT-Linear	mean	0.908136	0.951029	0.878206	0.869986	0.829841	0.779755	0.744782	0.88218	**0.956346**
	standard deviation	0.00408	0.005828	*0.004214*	0.009244	0.014956	0.023198	*0.015599*	0.021417	0.014686
SimCLRT-RNN	mean	0.910888	0.951567	0.878518	0.87754	0.835038	0.846605	0.809393	0.936605	0.94138
	standard deviation	0.007865	0.003516	0.010283	0.005851	0.009854	*0.023045*	0.04419	0.032037	0.014839

Table 6
The mean and standard deviation of SimCLRT on the RumorEval dataset.

Model		sydneysiege	ferguson	charliehebdo	ottawashooting	germanwings-crash	prince-toronto	putinmissing	ebola-essien
SimCLRT-CNN	mean	**0.908075**	**0.961329**	**0.893194**	0.916716	0.934061	0.886005	0.919262	**0.902778**
	standard deviation	0.010413	0.009814	0.009456	0.019765	0.016336	0.051378	0.075507	0.039284
SimCLRT-Linear	mean	0.900523	0.959189	0.890572	**0.920419**	**0.935432**	**0.898046**	**0.954605**	0.888889
	standard deviation	0.012678	*0.003800*	*0.007169*	*0.009675*	0.017519	*0.036411*	*0.032437*	*0.000000*
SimCLRT-RNN	mean	0.892143	0.945300	0.882027	0.912579	0.923595	0.859966	0.947136	0.888889
	standard deviation	*0.009948*	0.013008	0.018174	0.014861	*0.013472*	0.036707	0.037992	*0.000000*

by being obscured by other data. SimCLRT through the contrastive learning head uses contrastive learning to make tweets related to the same event closer in the representation space and tweets about different events further away. It helps to solve the imbalance situation of the number of event-related tweets. SimCLRT has greater robustness and stability with the help of data augmenter and contrastive learning.

The superiority of SimCLRT not only reflects on the performance, and the complexity of feature engineering is smaller than baselines. The feature engineering pipeline of baselines mainly includes data cleaning, building vocabulary, and training word embedding. The pipeline requires a lot of manual build processes, which need an amount of time and effort. However, SimCLRT's feature generator is BERT-based, which can automatically process variable-length data and extract features. Tweets contain many formatting symbols to indicate special meanings, such as the symbol @ indicating the users involved in the tweets. Through auxiliary experiments, we find to keep the formatting symbols is more beneficial to SimCLRT. Therefore, except specifying a suitable parameter of truncation length, we do not need to do more for the feature engineering of SimCLRT.

5.2. Comparison between different variants

Section 3.5 shows that we designed SimCLRT-CNN, SimCLRT-Linear, SimCLRT-RNN for SimCLRT based on CNN, RNN, and FCNN. In Tables 5 and 6, we report the mean and standard deviation of SimCLRT-CNN, SimCLRT-Linear, and SimCLRT-RNN on PHEME and RumorEval, respectively. We use boldface and italic to indicate the maximum mean and the minimum standard deviation of SimCLRT on a specific event, respectively. We find that the standard deviations are tiny on all variants. Therefore, we primarily focus on the mean acquired of models. On the PHEME dataset, SimCLRT-CNN acquires the most number of items that are bolded. On the dataset of RumorEval, SimCLRT-CNN and SimCLRT-Linear have the same amount of bolded items. From the number of bolded values, the performance of SimCLRT-Linear is the same as SimCLRT-CNN on the RumorEval dataset, but its advantage in the bolded items is minimal. The performance of SimCLRT-Linear is significantly weaker than SimCLRT-CNN and SimCLRT-RNN on the events prince, putinmissing, and ebola-essien as shown in Table 5.

More intuitively, Fig. 5 shows the overall performance of SimCLRT-CNN, SimCLRT-Linear, and SimCLRT-RNN on the PHEME and RumorEval datasets. As we can see, SimCLRT-CNN and SimCLRT-Linear

Table A.7
The result on PHEME dataset.

Model	Augment Methods	Substitution Rate/ Language Path	charliehebdo			ferguson			sydneysiege		
			precision	recall	f1-score	precision	recall	f1-score	precision	recall	f1-score
SVM-RBF	–	–	0.807967	0.964802	0.879447	0.955985	0.940171	0.948012	0.908733	0.833426	0.869452
DecisionTree	–	–	0.736499	0.786548	0.760701	0.839765	0.827957	0.833819	0.702998	0.697720	0.700349
RandomForest	–	–	0.694626	0.927862	0.794480	0.910963	0.829336	0.868235	0.826314	0.660178	0.733962
GaussianNB	–	–	0.899941	0.800836	0.847501	0.889749	0.887786	0.888766	0.822908	0.801168	0.811892
DPCNN	–	–	0.817464	0.931217	0.870641	0.876691	0.940553	0.907500	0.916751	0.764474	0.833716
TextRCNN	–	–	0.872829	0.904233	0.888254	0.940273	0.922690	0.931399	0.823044	0.869309	0.845544
TextCNN	–	–	0.844308	0.913589	0.877583	0.923246	0.928591	0.925911	0.841958	0.836343	0.839141
TextRNN	–	–	0.897286	0.881185	0.889162	0.929006	0.927213	0.928108	0.822157	0.861906	0.841563
Transformer	–	–	0.897750	0.945296	0.920910	0.941758	0.971878	0.956581	0.906073	0.887191	0.896532
SimCLRT-CNN	ContextualAugmenter	10%	0.921916	0.915331	0.918612	0.939200	0.971050	0.954860	0.875673	0.904140	0.889679
SimCLRT-RNN	ContextualAugmenter	10%	0.917703	0.913066	0.915379	0.937166	0.966363	0.951541	0.881623	0.887747	0.884674
SimCLRT-Linear	ContextualAugmenter	10%	0.888834	0.936063	0.911837	0.950712	0.957265	0.953977	0.871187	0.888858	0.879934
SimCLRT-CNN	ContextualAugmenter	20%	0.928201	0.905401	0.916659	0.964454	0.957541	0.960985	0.866898	0.903029	0.884594
SimCLRT-RNN	ContextualAugmenter	20%	0.908449	0.919686	0.914033	0.945499	0.961401	0.953383	0.897582	0.876632	0.886983
SimCLRT-Linear	ContextualAugmenter	20%	0.891733	0.922648	0.906927	0.956882	0.954508	0.955694	0.868635	0.891081	0.879715
SimCLRT-CNN	ContextualAugmenter	30%	0.936065	0.887631	0.911205	0.948038	0.965812	0.956842	0.848900	0.911642	0.879153
SimCLRT-RNN	ContextualAugmenter	30%	0.902451	0.923519	0.912864	0.939191	0.966639	0.952717	0.887675	0.880522	0.884084
SimCLRT-Linear	ContextualAugmenter	30%	0.891733	0.922648	0.906927	0.956882	0.954508	0.955694	0.868635	0.891081	0.879715
SimCLRT-CNN	WordNetAugmenter	10%	0.902292	0.939711	0.920621	0.955191	0.963882	0.959517	0.908333	0.879032	0.893443
SimCLRT-RNN	WordNetAugmenter	10%	0.907716	0.920369	0.913999	0.947153	0.958644	0.952864	0.878300	0.879032	0.878666
SimCLRT-Linear	WordNetAugmenter	10%	0.915758	0.907301	0.911510	0.946419	0.954508	0.950446	0.848406	0.902670	0.874697
SimCLRT-CNN	WordNetAugmenter	20%	0.919332	0.911483	0.915391	0.939483	0.971602	0.955272	0.874124	0.901835	0.887763
SimCLRT-RNN	WordNetAugmenter	20%	0.910965	0.916362	0.913655	0.943050	0.963331	0.953082	0.875307	0.892102	0.883625
SimCLRT-Linear	WordNetAugmenter	20%	0.886673	0.929779	0.907715	0.960493	0.945134	0.952752	0.874207	0.881257	0.877718
SimCLRT-CNN	WordNetAugmenter	30%	0.905986	0.933612	0.919592	0.961952	0.961952	0.961952	0.892274	0.895996	0.894131
SimCLRT-RNN	WordNetAugmenter	30%	0.897440	0.934658	0.915671	0.956835	0.953405	0.955117	0.885738	0.877364	0.881531
SimCLRT-Linear	WordNetAugmenter	30%	0.886958	0.933786	0.909770	0.950739	0.957816	0.954265	0.885874	0.880701	0.883280
SimCLRT-CNN	Back translation	German–French	0.904819	0.925945	0.915260	0.963805	0.947064	0.955361	0.835498	0.912403	0.872258
SimCLRT-RNN	Back translation	German–French	0.924572	0.894929	0.909509	0.969315	0.931900	0.950239	0.842732	0.906007	0.873224
SimCLRT-Linear	Back translation	German–French	0.923862	0.898589	0.911050	0.923682	0.971050	0.946774	0.865932	0.896274	0.880842
SimCLRT-CNN	Back translation	French–Chinese	0.868571	0.953476	0.909046	0.966311	0.948994	0.957574	0.926207	0.848165	0.885470
SimCLRT-RNN	Back translation	French–Chinese	0.930858	0.856247	0.891995	0.937177	0.950096	0.943593	0.804806	0.912681	0.855356
SimCLRT-Linear	Back translation	French–Chinese	0.873154	0.927165	0.899349	0.926652	0.950924	0.938631	0.879012	0.860679	0.869748

Model	Augment Methods	Substitution Rate/ Language Path	ottawashooting			germanwings			prince		
			precision	recall	f1-score	precision	recall	f1-score	precision	recall	f1-score
SVM-RBF	–	–	0.965108	0.750407	0.844322	0.968958	0.649331	0.777580	1.000000	0.562963	0.720379
DecisionTree	–	–	0.760547	0.694520	0.726035	0.775862	0.668648	0.718276	0.796296	0.637037	0.707819
RandomForest	–	–	0.926941	0.660879	0.771619	0.954545	0.624071	0.754717	0.973684	0.548148	0.701422
GaussianNB	–	–	0.682268	0.822572	0.745879	0.532242	0.772660	0.630303	0.656934	0.666667	0.661765
DPCNN	–	–	0.862112	0.762637	0.809329	0.861163	0.688156	0.765000	0.650794	0.616541	0.633205
TextRCNN	–	–	0.850196	0.835714	0.842893	0.930894	0.686657	0.790336	0.911111	0.616541	0.735426
TextCNN	–	–	0.871143	0.781335	0.823799	0.915152	0.673105	0.775685	0.865979	0.622222	0.724138
TextRNN	–	–	0.814580	0.836679	0.825482	0.838261	0.716196	0.772436	0.694215	0.622222	0.656250
Transformer	–	–	0.942055	0.855670	0.896787	0.946714	0.791976	0.862460	0.990476	0.770370	0.866667
SimCLRT-CNN	ContextualAugmenter	10%	0.924190	0.866522	0.894427	0.881517	0.829123	0.854518	0.965812	0.837037	0.896825
SimCLRT-RNN	ContextualAugmenter	10%	0.897291	0.862724	0.879668	0.845802	0.823180	0.834337	0.881890	0.829630	0.854962
SimCLRT-Linear	ContextualAugmenter	10%	0.933658	0.832339	0.880092	0.912186	0.756315	0.826970	0.979167	0.696296	0.813853
SimCLRT-CNN	ContextualAugmenter	20%	0.870137	0.897992	0.883845	0.878505	0.838039	0.857795	0.948276	0.814815	0.876494
SimCLRT-RNN	ContextualAugmenter	20%	0.894590	0.870320	0.882288	0.832572	0.812779	0.822556	0.909836	0.822222	0.863813
SimCLRT-Linear	ContextualAugmenter	20%	0.893143	0.848074	0.870025	0.900870	0.769688	0.830128	0.976471	0.614815	0.754545
SimCLRT-CNN	ContextualAugmenter	30%	0.888043	0.886598	0.887320	0.826277	0.841010	0.833579	1.000000	0.770370	0.870293
SimCLRT-RNN	ContextualAugmenter	30%	0.911290	0.858383	0.884046	0.902730	0.786033	0.840349	0.830882	0.837037	0.833948
SimCLRT-Linear	ContextualAugmenter	30%	0.893143	0.848074	0.870025	0.900870	0.769688	0.830128	0.976471	0.614815	0.754545
SimCLRT-CNN	WordNetAugmenter	10%	0.905299	0.871405	0.888029	0.877519	0.841010	0.858877	0.971963	0.770370	0.859504
SimCLRT-RNN	WordNetAugmenter	10%	0.897553	0.855670	0.876111	0.848624	0.824666	0.836473	0.924370	0.814815	0.866142
SimCLRT-Linear	WordNetAugmenter	10%	0.892978	0.855670	0.873926	0.852146	0.796434	0.823349	0.977528	0.644444	0.776786
SimCLRT-CNN	WordNetAugmenter	20%	0.915711	0.866522	0.890438	0.879365	0.823180	0.850345	0.933333	0.829630	0.878431
SimCLRT-RNN	WordNetAugmenter	20%	0.898086	0.865437	0.881459	0.886139	0.797920	0.839719	0.896000	0.829630	0.861538
SimCLRT-Linear	WordNetAugmenter	20%	0.899541	0.850244	0.874198	0.875000	0.790490	0.830601	0.968085	0.674074	0.794760
SimCLRT-CNN	WordNetAugmenter	30%	0.908782	0.870320	0.889135	0.895595	0.815750	0.853810	0.965812	0.837037	0.896825
SimCLRT-RNN	WordNetAugmenter	30%	0.912281	0.846446	0.878131	0.899506	0.811293	0.853125	0.746753	0.851852	0.795848
SimCLRT-Linear	WordNetAugmenter	30%	0.900759	0.837222	0.867829	0.912671	0.791976	0.848051	0.989130	0.674074	0.801762
SimCLRT-CNN	Back translation	German–French	0.945545	0.829083	0.883492	0.917384	0.791976	0.850080	1.000000	0.681481	0.810573
SimCLRT-RNN	Back translation	German–French	0.870340	0.874118	0.872225	0.801966	0.848440	0.824549	0.929204	0.777778	0.846774
SimCLRT-Linear	Back translation	German–French	0.884146	0.865437	0.874692	0.884058	0.815750	0.848532	0.957447	0.666667	0.786026
SimCLRT-CNN	Back translation	French–Chinese	0.890808	0.867607	0.879054	0.917808	0.796434	0.852824	0.963964	0.792593	0.869919
SimCLRT-RNN	Back translation	French–Chinese	0.876249	0.856755	0.866392	0.839939	0.818722	0.829195	0.840580	0.859259	0.849817
SimCLRT-Linear	Back translation	French–Chinese	0.891408	0.810635	0.849105	0.875000	0.738484	0.800967	1.000000	0.607407	0.755760

Model	Augment Methods	Substitution Rate/ Language Path	putinmissing			ebola-essien			gurlitt		
			precision	recall	f1-score	precision	recall	f1-score	precision	recall	f1-score
SVM-RBF	–	–	0.932432	0.552000	0.693467	1.000000	0.676471	0.807018	1.000000	0.888889	0.941176
DecisionTree	–	–	0.696970	0.552000	0.616071	0.892857	0.735294	0.806452	0.888889	0.888889	0.888889
RandomForest	–	–	0.949367	0.600000	0.735294	1.000000	0.705882	0.827586	1.000000	0.925926	0.961538
GaussianNB	–	–	0.474820	0.528000	0.500000	0.489362	0.676471	0.567901	0.380000	0.703704	0.493506
DPCNN	–	–	0.843137	0.346774	0.491429	0.900000	0.264706	0.409091	1.000000	0.592593	0.744186
TextRCNN	–	–	0.920635	0.467742	0.620321	1.000000	0.441176	0.612245	1.000000	0.703704	0.826087
TextCNN	–	–	0.878049	0.576000	0.695652	1.000000	0.647059	0.785714	1.000000	0.740741	0.851064
TextRNN	–	–	0.566667	0.544000	0.555102	0.375000	0.441176	0.405405	0.833333	0.740741	0.784314
Transformer	–	–	0.920792	0.744000	0.823009	1.000000	0.823529	0.903226	1.000000	0.777778	0.875000
SimCLRT-CNN	ContextualAugmenter	10%	0.980392	0.800000	0.881057	1.000000	0.941176	0.969697	0.961538	0.925926	0.943396
SimCLRT-RNN	ContextualAugmenter	10%	0.887931	0.824000	0.854772	1.000000	0.941176	0.969697	0.961538	0.925926	0.943396
SimCLRT-Linear	ContextualAugmenter	10%	0.939024	0.616000	0.743961	0.933333	0.823529	0.875000	1.000000	0.925926	0.961538
SimCLRT-CNN	ContextualAugmenter	20%	0.798507	0.856000	0.826255	0.970588	0.970588	0.970588	0.961538	0.925926	0.943396
SimCLRT-RNN	ContextualAugmenter	20%	0.675159	0.848000	0.751773	0.970588	0.970588	0.970588	0.961538	0.925926	0.943396
SimCLRT-Linear	ContextualAugmenter	20%	0.887640	0.632000	0.738318	0.933333	0.823529	0.875000	1.000000	0.925926	0.961538
SimCLRT-CNN	ContextualAugmenter	30%	0.937500	0.840000	0.886076	1.000000	0.970588	0.985075	0.961538	0.925926	0.943396
SimCLRT-RNN	ContextualAugmenter	30%	0.918919	0.816000	0.864407	1.000000	0.911765	0.953846	0.961538	0.925926	0.943396
SimCLRT-Linear	ContextualAugmenter	30%	0.887640	0.632000	0.738318	0.933333	0.823529	0.875000	1.000000	0.925926	0.961538
SimCLRT-CNN	WordNetAugmenter	10%	0.979592	0.768000	0.860987	1.000000	0.970588	0.985075	1.000000	0.925926	0.961538
SimCLRT-RNN	WordNetAugmenter	10%	0.854701	0.800000	0.826446	0.888889	0.941176	0.914286	0.925926	0.925926	0.925926
SimCLRT-Linear	WordNetAugmenter	10%	0.879121	0.640000	0.740741	0.931034	0.794118	0.857143	1.000000	0.925926	0.961538
SimCLRT-CNN	WordNetAugmenter	20%	0.979381	0.760000	0.855856	1.000000	0.941176	0.969697	1.000000	0.925926	0.961538
SimCLRT-RNN	WordNetAugmenter	20%	0.956044	0.696000	0.805556	1.000000	0.911765	0.953846	0.925926	0.925926	0.925926
SimCLRT-Linear	WordNetAugmenter	20%	0.900000	0.648000	0.753488	0.964286	0.794118	0.870968	1.000000	0.925926	0.961538
SimCLRT-CNN	WordNetAugmenter	30%	0.989247	0.736000	0.844037	1.000000	0.970588	0.985075	0.961538	0.925926	0.943396
SimCLRT-RNN	WordNetAugmenter	30%	0.868421	0.792000	0.828452	0.888889	0.941176	0.914286	1.000000	0.925926	0.961538
SimCLRT-Linear	WordNetAugmenter	30%	0.928571	0.624000	0.746411	0.966667	0.852941	0.906250	1.000000	0.925926	0.961538
SimCLRT-CNN	Back translation	German–French	0.989130	0.728000	0.838710	1.000000	0.852941	0.920635	1.000000	0.925926	0.961538
SimCLRT-RNN	Back translation	German–French	0.664557	0.840000	0.742049	1.000000	0.882353	0.937500	0.925926	0.925926	0.925926
SimCLRT-Linear	Back translation	German–French	0.841121	0.720000	0.775862	0.967742	0.882353	0.923077	1.000000	0.925926	0.961538
SimCLRT-CNN	Back translation	French–Chinese	0.979381	0.760000	0.855856	1.000000	0.941176	0.969697	1.000000	0.925926	0.961538
SimCLRT-RNN	Back translation	French–Chinese	0.848214	0.760000	0.801688	0.906250	0.852941	0.878788	1.000000	0.925926	0.961538
SimCLRT-Linear	Back translation	French–Chinese	0.903614	0.600000	0.721154	0.933333	0.823529	0.875000	1.000000	0.851852	0.920000

Table B.8
The result on RumorEval dataset.

Model	Augment Methods	Substitution Rate/ Language Path	sydneysiege precision	recall	f1-score	ferguson precision	recall	f1-score	charliehebdo precision	recall	f1-score
SVM-RBF	–	–	0.792929	0.945783	0.862637	0.913295	0.969325	0.940476	0.850575	0.919255	0.883582
DecisionTree	–	–	0.849398	0.849398	0.849398	0.927632	0.865031	0.895238	0.763006	0.819876	0.790419
RandomForest	–	–	0.818681	0.897590	0.856322	0.967949	0.926380	0.946708	0.774194	0.894410	0.829971
GaussianNB	–	–	0.802260	0.855422	0.827988	0.834286	0.895706	0.863905	0.800000	0.720497	0.758170
DPCNN	–	–	0.804688	0.811024	0.807843	0.757143	0.898305	0.821705	0.872549	0.695313	0.773913
TextRCNN	–	–	0.815603	0.905512	0.858209	0.876033	0.898305	0.887029	0.801471	0.851563	0.825758
TextCNN	–	–	0.817647	0.837349	0.827381	0.852273	0.920245	0.884956	0.824242	0.844720	0.834356
TextRNN	–	–	0.774194	0.755906	0.764940	0.908163	0.754237	0.824074	0.676692	0.703125	0.689655
Transformer	–	–	0.877907	0.909639	0.893491	0.945455	0.957055	0.951220	0.835366	0.850932	0.843077
SimCLRT-CNN	ContextualAugmenter	10%	0.890805	0.933735	0.911765	0.962733	0.950920	0.956790	0.957746	0.844720	0.897690
SimCLRT-RNN	ContextualAugmenter	10%	0.857143	0.903614	0.879765	0.921687	0.938650	0.930091	0.922581	0.888199	0.905063
SimCLRT-Linear	ContextualAugmenter	10%	0.888889	0.915663	0.902077	0.957055	0.957055	0.957055	0.905660	0.894410	0.900000
SimCLRT-CNN	ContextualAugmenter	20%	0.875000	0.927711	0.900585	0.950920	0.950920	0.950920	0.927632	0.875776	0.900958
SimCLRT-RNN	ContextualAugmenter	20%	0.903030	0.897590	0.900302	0.962733	0.950920	0.956790	0.879518	0.906832	0.892966
SimCLRT-Linear	ContextualAugmenter	20%	0.897727	0.951807	0.923977	0.945783	0.963190	0.954407	0.921569	0.875776	0.898089
SimCLRT-CNN	ContextualAugmenter	30%	0.897143	0.945783	0.920821	0.981132	0.957055	0.968944	0.922581	0.888199	0.905063
SimCLRT-RNN	ContextualAugmenter	30%	0.838710	0.939759	0.886364	0.951220	0.957055	0.954128	0.910256	0.881988	0.895899
SimCLRT-Linear	ContextualAugmenter	30%	0.870787	0.933735	0.901163	0.957317	0.963190	0.960245	0.898734	0.881988	0.890282
SimCLRT-CNN	WordNetAugmenter	10%	0.909639	0.909639	0.909639	0.974684	0.944785	0.959502	0.889571	0.900621	0.895062
SimCLRT-RNN	WordNetAugmenter	10%	0.840659	0.921687	0.879310	0.945122	0.950920	0.948012	0.871795	0.844720	0.858044
SimCLRT-Linear	WordNetAugmenter	10%	0.885057	0.927711	0.905882	0.957317	0.963190	0.960245	0.887500	0.881988	0.884735
SimCLRT-CNN	WordNetAugmenter	20%	0.871508	0.939759	0.904348	0.993590	0.950920	0.971787	0.918367	0.838509	0.876623
SimCLRT-RNN	WordNetAugmenter	20%	0.859551	0.921687	0.889535	0.940476	0.969325	0.954683	0.937500	0.838509	0.885246
SimCLRT-Linear	WordNetAugmenter	20%	0.860335	0.927711	0.892754	0.962733	0.950920	0.956790	0.903846	0.875776	0.889590
SimCLRT-CNN	WordNetAugmenter	30%	0.890173	0.927711	0.908555	0.950617	0.944785	0.947692	0.921569	0.875776	0.898089
SimCLRT-RNN	WordNetAugmenter	30%	0.947020	0.861446	0.902208	0.945783	0.963190	0.954407	0.815642	0.906832	0.858824
SimCLRT-Linear	WordNetAugmenter	30%	0.846154	0.927711	0.885057	0.951515	0.963190	0.957317	0.901961	0.857143	0.878981
SimCLRT-CNN	Back translation	German–French	0.863636	0.915663	0.888889	0.993631	0.957055	0.975000	0.903226	0.869565	0.886076
SimCLRT-RNN	Back translation	German–French	0.892216	0.897590	0.894895	0.905325	0.938650	0.921687	0.883871	0.850932	0.867089
SimCLRT-Linear	Back translation	German–French	0.858757	0.915663	0.886297	0.951807	0.969325	0.960486	0.898089	0.875776	0.886792
SimCLRT-CNN	Back translation	French–Chinese	0.875000	0.969880	0.920000	0.962963	0.957055	0.960000	0.931507	0.844720	0.885993
SimCLRT-RNN	Back translation	French–Chinese	0.894118	0.915663	0.904762	0.928571	0.957055	0.942598	0.904459	0.881988	0.893082
SimCLRT-Linear	Back translation	French–Chinese	0.876404	0.939759	0.906977	0.947059	0.987730	0.966967	0.938776	0.857143	0.896104

Model	Augment Methods	Substitution Rate/ Language Path	ottawashooting precision	recall	f1-score	germanwings-crash precision	recall	f1-score	prince-toronto precision	recall	f1-score
SVM-RBF	–	–	0.969072	0.803419	0.878505	0.964286	0.642857	0.771429	1.000000	0.333333	0.500000
DecisionTree	–	–	0.776000	0.829060	0.801653	0.864865	0.761905	0.810127	0.750000	0.600000	0.666667
RandomForest	–	–	0.950000	0.811966	0.875576	0.916667	0.785714	0.846154	1.000000	0.600000	0.750000
GaussianNB	–	–	0.756098	0.794872	0.775000	0.897436	0.833333	0.864198	0.909091	0.666667	0.769231
DPCNN	–	–	0.726190	0.701149	0.713450	0.791667	0.612903	0.690909	0.208333	0.555556	0.303030
TextRCNN	–	–	0.934211	0.816092	0.871166	0.838710	0.838710	0.838710	1.000000	0.555556	0.714286
TextCNN	–	–	0.834783	0.820513	0.827586	0.868421	0.785714	0.825000	1.000000	0.533333	0.695652
TextRNN	–	–	0.812500	0.448276	0.577778	0.222222	0.645161	0.330579	0.375000	0.333333	0.352941
Transformer	–	–	0.818898	0.888889	0.852459	0.966667	0.690476	0.805556	1.000000	0.666667	0.800000
SimCLRT-CNN	ContextualAugmenter	10%	0.857143	0.974359	0.912000	0.950000	0.904762	0.926829	0.800000	0.800000	0.800000
SimCLRT-RNN	ContextualAugmenter	10%	0.877049	0.914530	0.895397	1.000000	0.857143	0.923077	1.000000	0.800000	0.888889
SimCLRT-Linear	ContextualAugmenter	10%	0.901639	0.940171	0.920502	1.000000	0.857143	0.923077	1.000000	0.866667	0.928571
SimCLRT-CNN	ContextualAugmenter	20%	0.918699	0.965812	0.941667	0.973684	0.880952	0.925000	1.000000	0.800000	0.888889
SimCLRT-RNN	ContextualAugmenter	20%	0.910569	0.957265	0.933333	0.974359	0.904762	0.938272	1.000000	0.800000	0.888889
SimCLRT-Linear	ContextualAugmenter	20%	0.924370	0.940171	0.932203	0.973684	0.880952	0.925000	1.000000	0.733333	0.846154
SimCLRT-CNN	ContextualAugmenter	30%	0.940171	0.940171	0.940171	0.928571	0.928571	0.928571	1.000000	0.866667	0.928571
SimCLRT-RNN	ContextualAugmenter	30%	0.963303	0.897436	0.929204	1.000000	0.880952	0.936709	1.000000	0.800000	0.888889
SimCLRT-Linear	ContextualAugmenter	30%	0.923077	0.923077	0.923077	1.000000	0.857143	0.923077	1.000000	0.733333	0.846154
SimCLRT-CNN	WordNetAugmenter	10%	0.888889	0.957265	0.921811	0.972973	0.857143	0.911392	1.000000	0.866667	0.928571
SimCLRT-RNN	WordNetAugmenter	10%	0.923077	0.923077	0.923077	1.000000	0.833333	0.909091	1.000000	0.733333	0.846154
SimCLRT-Linear	WordNetAugmenter	10%	0.914530	0.914530	0.914530	1.000000	0.880952	0.936709	1.000000	0.800000	0.888889
SimCLRT-CNN	WordNetAugmenter	20%	0.838462	0.931624	0.882591	1.000000	0.904762	0.950000	1.000000	0.866667	0.928571
SimCLRT-RNN	WordNetAugmenter	20%	0.892562	0.923077	0.907563	0.904762	0.904762	0.904762	1.000000	0.733333	0.846154
SimCLRT-Linear	WordNetAugmenter	20%	0.913793	0.905983	0.909871	0.974359	0.904762	0.938272	1.000000	0.866667	0.928571
SimCLRT-CNN	WordNetAugmenter	30%	0.888000	0.948718	0.917355	0.974359	0.904762	0.938272	1.000000	0.733333	0.846154
SimCLRT-RNN	WordNetAugmenter	30%	0.914530	0.914530	0.914530	0.973684	0.880952	0.925000	1.000000	0.800000	0.888889
SimCLRT-Linear	WordNetAugmenter	30%	0.921053	0.897436	0.909091	0.973684	0.880952	0.925000	1.000000	0.866667	0.928571
SimCLRT-CNN	Back translation	German–French	0.890756	0.905983	0.898305	0.928571	0.928571	0.928571	0.812500	0.866667	0.838710
SimCLRT-RNN	Back translation	German–French	0.877049	0.914530	0.895397	0.974359	0.904762	0.938272	1.000000	0.733333	0.846154
SimCLRT-Linear	Back translation	German–French	0.939655	0.931624	0.935622	1.000000	0.880952	0.936709	1.000000	0.800000	0.888889
SimCLRT-CNN	Back translation	French–Chinese	0.908333	0.931624	0.919831	0.975610	0.952381	0.963855	1.000000	0.866667	0.928571
SimCLRT-RNN	Back translation	French–Chinese	0.898305	0.905983	0.902128	0.948718	0.880952	0.913580	0.846154	0.733333	0.785714
SimCLRT-Linear	Back translation	French–Chinese	0.922414	0.914530	0.918455	1.000000	0.952381	0.975610	1.000000	0.866667	0.928571

Model	Augment Methods	Substitution Rate/ Language Path	putinmissing precision	recall	f1-score	ebola-essien precision	recall	f1-score
SVM-RBF	–	–	1.000000	0.111111	0.200000	1.000000	0.400000	0.571429
DecisionTree	–	–	0.888889	0.888889	0.888889	1.000000	0.800000	0.888889
RandomForest	–	–	1.000000	0.666667	0.800000	1.000000	0.600000	0.750000
GaussianNB	–	–	0.800000	0.444444	0.571429	1.000000	0.600000	0.750000
DPCNN	–	–	0.428571	0.333333	0.375000	0.666667	0.666667	0.666667
TextRCNN	–	–	1.000000	0.111111	0.200000	1.000000	0.333333	0.500000
TextCNN	–	–	1.000000	0.444444	0.615385	1.000000	0.400000	0.571429
TextRNN	–	–	0.000000	0.000000	0.000000	0.166667	0.333333	0.222222
Transformer	–	–	0.714286	0.555556	0.625000	1.000000	0.600000	0.750000
SimCLRT-CNN	ContextualAugmenter	10%	1.000000	1.000000	1.000000	1.000000	0.800000	0.888889
SimCLRT-RNN	ContextualAugmenter	10%	1.000000	0.888889	0.941176	1.000000	0.800000	0.888889
SimCLRT-Linear	ContextualAugmenter	10%	0.900000	1.000000	0.947368	1.000000	0.800000	0.888889
SimCLRT-CNN	ContextualAugmenter	20%	0.900000	1.000000	0.947368	1.000000	0.800000	0.888889
SimCLRT-RNN	ContextualAugmenter	20%	1.000000	0.888889	0.941176	1.000000	0.800000	0.888889
SimCLRT-Linear	ContextualAugmenter	20%	0.818182	1.000000	0.900000	1.000000	0.800000	0.888889
SimCLRT-CNN	ContextualAugmenter	30%	0.692308	1.000000	0.818182	1.000000	0.800000	0.888889
SimCLRT-RNN	ContextualAugmenter	30%	0.900000	1.000000	0.947368	1.000000	0.800000	0.888889
SimCLRT-Linear	ContextualAugmenter	30%	0.900000	1.000000	0.947368	1.000000	0.800000	0.888889
SimCLRT-CNN	WordNetAugmenter	10%	0.818182	1.000000	0.900000	1.000000	0.800000	0.888889
SimCLRT-RNN	WordNetAugmenter	10%	1.000000	1.000000	1.000000	1.000000	0.800000	0.888889
SimCLRT-Linear	WordNetAugmenter	10%	0.900000	1.000000	0.947368	1.000000	0.800000	0.888889
SimCLRT-CNN	WordNetAugmenter	20%	0.727273	0.888889	0.800000	1.000000	0.800000	0.888889
SimCLRT-RNN	WordNetAugmenter	20%	0.900000	1.000000	0.947368	1.000000	0.800000	0.888889
SimCLRT-Linear	WordNetAugmenter	20%	0.900000	1.000000	0.947368	1.000000	0.800000	0.888889
SimCLRT-CNN	WordNetAugmenter	30%	0.900000	1.000000	0.947368	1.000000	1.000000	1.000000
SimCLRT-RNN	WordNetAugmenter	30%	0.818182	1.000000	0.900000	1.000000	0.800000	0.888889
SimCLRT-Linear	WordNetAugmenter	30%	1.000000	1.000000	1.000000	1.000000	0.800000	0.888889
SimCLRT-CNN	Back translation	German–French	1.000000	1.000000	1.000000	1.000000	0.800000	0.888889
SimCLRT-RNN	Back translation	German–French	0.818182	1.000000	0.900000	1.000000	0.800000	0.888889
SimCLRT-Linear	Back translation	German–French	1.000000	1.000000	1.000000	1.000000	0.800000	0.888889
SimCLRT-CNN	Back translation	French–Chinese	1.000000	0.888889	0.941176	1.000000	0.800000	0.888889
SimCLRT-RNN	Back translation	French–Chinese	1.000000	1.000000	1.000000	1.000000	0.800000	0.888889
SimCLRT-Linear	Back translation	French–Chinese	0.900000	1.000000	0.947368	1.000000	0.800000	0.888889

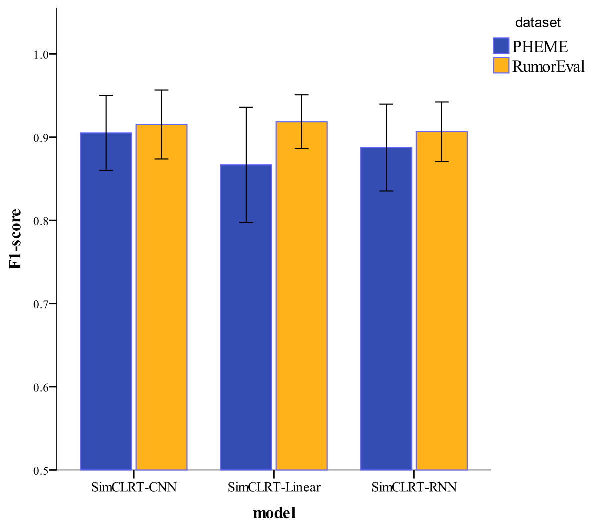


Fig. 5. The mean and standard deviation of SimCLRT on the PHEME and RumorEval datasets.

achieved the best F1-Score on PHEME and RumorEval, respectively, and the performance difference between SimCLRT-CNN and SimCLRT-Linear on RumorEval is tiny. Although SimCLRT-Linear has achieved the best performance on RumorEval, its performance on the two datasets fluctuates wildly. The fluctuating on different datasets reflects the weak robustness of SimCLRT-Linear. In addition, although SimCLRT-RNN does not achieve the best performance on any dataset, it has very little difference in performance between the two datasets. On the whole, SimCLRT-CNN is stronger than SimCLRT-Linear and SimCLRT-RNN on the PHEME and RumorEval datasets. We think it is because SimCLRT-CNN can capture richer information using convolution kernels of different sizes. It is worth noting that SimCLRT-RNN does not play the most significant advantage in the experiments. SimCLRT-RNN can use the Long Short Time Memory(LSTM) structure to capture long-term dependent information in sentences, but tweets are short texts. We think SimCLRT-RNN will be an up-and-coming model when faced with long text.

SimCLRT is a worth-trying classification model, especially when faced with imbalanced classification problems. Here, we would like to discuss the usage of SimCLRT-Linear, SimCLRT-RNN, and SimCLRT-CNN combined with the experimental results and the structural details of the classification heads we discussed in Section 3. For a certain dataset, we cannot directly determine which of SimCLRT-Linear, SimCLRT-CNN, SimCLRT-RNN performs best without conducting experiments. Although SimCLRT-Linear shows weak robustness in our experiments, we suggest to try SimCLRT-Linear first because it is the easiest to implement and can be used to quickly implement SimCLRT-RNN or SimCLRT-CNN. The trade-off between SimCLRT-CNN and SimCLRT-RNN depends on the text length. If it is a long text classification problem, we recommend SimCLRT-RNN because the design of the SimCLRT-RNN is more helpful to maintain contextual information in long sequences; otherwise, we recommend SimCLRT-CNN because the classification head of SimCLRT-CNN can obtain richer information from the feature generator through convolution operations, and our experiments have proved the effectiveness of SimCLRT-CNN on short text classification.

6. Conclusions

We propose a new framework, SimCLRT, which uses contrastive learning to support rumor tracking. The SimCLRT includes three variants, SimCLRT-CNN, SimCLRT-Linear, and SimCLRT-RNN. We evaluate our model on two standard datasets, and the results showed that our model completely defeated the baseline. We prove that the introduction of contrastive learning can effectively alleviate the overlap between tweets related to different events, and effectively improve the performance of rumor tracking. Furthermore, we carefully compare and analyze the performance of SimCLRT-CNN, SimCLRT-Linear, and SimCLRT-RNN. SimCLRT-CNN achieves the best performance because it can use different sizes of convolution kernels to extract richer information. Although SimCLRT-Linear has a slight advantage on the dataset of RumorEval, its robustness is weaker than SimCLRT-CNN and SimCLRT-RNN. Conversely, although SimCLRT-RNN did not achieve the best results on any dataset in the experiment, we believe that SimCLRT-RNN will is a competitive model when faced with long text.

Our model is not sensitive to the number of tweets contained in the event and achieved a great score on the existing datasets. However, the amount of datasets for rumor tracking is scarce currently, and the size of the existing datasets is small. The construction of the rumor tracking dataset is a long-term follow-up and valuable work.

CRediT authorship contribution statement

Hui Zeng: Conceptualization, Methodology, Software, Validation, Formal analysis, Investigation, Data curation, Writing – original draft, Visualization, Writing – review & editing. **Xiaohui Cui:** Resources, Writing – review & editing, Supervision, Project administration, Funding acquisition.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This research is supported by National Natural Science Foundation of China (No. 62002313).

Appendix A. All the results on the dataset PHEME

See Table A.7.

Appendix B. All the results on the dataset RumorEval

See Table B.8.

References

Cheng, M., Nazarian, S., Bogdan, P., 2020. Vroc: Variational autoencoder-aided multi-task rumor classifier based on text. In: Proceedings of the Web Conference 2020. pp. 2892–2898.
Dong, X., Victor, U., Chowdhury, S., Qian, L., 2019. Deep two-path semi-supervised learning for fake news detection. arXiv preprint arXiv:1906.05659.
Enayet, O., El-Beltagy, S.R., 2017a. NileTMRG at SemEval-2017 task 8: Determining rumour and veracity support for rumours on Twitter. In: Proceedings of the 11th International Workshop on Semantic Evaluation. SemEval-2017, pp. 470–474.
Enayet, O., El-Beltagy, S.R., 2017b. Niletmrg at semeval-2017 task 8: determining rumour and veracity support for rumours on twitter.. In: Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017). Association for Computational Linguistics, Vancouver, Canada, pp. 470–474. http://dx.doi.org/10.18653/v1/S17-2082, https://aclanthology.org/S17-2082.
Guo, W., Diab, M., 2012. Modeling sentences in the latent space. In: Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). pp. 864–872.
Hamidian, S., Diab, M., 2016. Rumor identification and belief investigation on twitter. In: Proceedings of the 7th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis. pp. 3–8.
Hamidian, S., Diab, M.T., 2019. Rumor detection and classification for twitter data. arXiv preprint arXiv:1912.08926.

Johnson, R., Zhang, T., 2017. Deep pyramid convolutional neural networks for text categorization. In: Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). pp. 562–570.

Kim, Y., 2014. Convolutional neural networks for sentence classification. In: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP). Association for Computational Linguistics, Doha, Qatar, pp. 1746–1751. http://dx.doi.org/10.3115/v1/D14-1181, URL https://aclanthology.org/D14-1181.

Kochkina, E., Liakata, M., Augenstein, I., 2017. Turing at SemEval-2017 task 8: Sequential approach to rumour stance classification with branch-LSTM. In: Proceedings of the 11th International Workshop on Semantic Evaluation. SemEval-2017, pp. 475–480.

Kochkina, E., Liakata, M., Zubiaga, A., 2018. All-in-one: Multi-task learning for rumour verification. In: Proceedings of the 27th International Conference on Computational Linguistics. pp. 3402–3413.

Kumar, S., Carley, K.M., 2019. Tree lstms with convolution units to predict stance and rumor veracity in social media conversations. In: Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics. pp. 5047–5058.

Lai, S., Xu, L., Liu, K., Zhao, J., 2015. Recurrent convolutional neural networks for text classification. In: Twenty-Ninth AAAI Conference on Artificial Intelligence.

Li, G., Dong, M., Ming, L., Luo, C., Yu, H., Hu, X., Zheng, B., 2021. Deep reinforcement learning based ensemble model for rumor tracking. Inf. Syst. 101772.

Li, Q., Zhang, Q., Si, L., 2019. 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.

Ma, E., 2019. NLP Augmentation, https://github.com/makcedward/nlpaug.

Ma, J., Gao, W., Wong, K.-F., 2019. Detect rumors on twitter by promoting information campaigns with generative adversarial learning. In: The World Wide Web Conference. pp. 3049–3055.

Miller, D.J., Uyar, H.S., 1997. A mixture of experts classifier with learning based on both labelled and unlabelled data. In: Advances in Neural Information Processing Systems. pp. 571–577.

Morris, J., Lifland, E., Yoo, J.Y., Grigsby, J., Jin, D., Qi, Y., 2020. Textattack: a framework for adversarial attacks, data augmentation, and adversarial training in nlp. In: Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations. pp. 119–126.

Nguyen, D.M., Do, T.H., Calderbank, R., Deligiannis, N., 2019. Fake news detection using deep markov random fields. In: Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers). pp. 1391–1400.

Qazvinian, V., Rosengren, E., Radev, D., Mei, Q., 2011. Rumor has it: Identifying misinformation in microblogs. In: Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing. pp. 1589–1599.

Reimers, N., Gurevych, I., 2019. Sentence-BERT: Sentence embeddings using siamese BERT-networks. In: Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing. Association for Computational Linguistics, URL https://arxiv.org/abs/1908.10084.

Ren, S., Deng, Y., He, K., Che, W., 2019. Generating natural language adversarial examples through probability weighted word saliency. In: Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics. Association for Computational Linguistics, Florence, Italy, pp. 1085–1097. http://dx.doi.org/10.18653/v1/P19-1103, https://aclanthology.org/P19-1103.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I., 2017. Attention is all you need. In: Advances in Neural Information Processing Systems. pp. 5998–6008.

Zhang, D., Nan, F., Wei, X., Li, S.-W., Zhu, H., McKeown, K., Nallapati, R., Arnold, A.O., Xiang, B., 2021. Supporting clustering with contrastive learning. In: Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies. pp. 5419–5430.

Zubiaga, A., Aker, A., Bontcheva, K., Liakata, M., Procter, R., 2018. Detection and resolution of rumours in social media: A survey. ACM Comput. Surv. 51 (2), 1–36.

Zubiaga, A., Kochkina, E., Liakata, M., Procter, R., Lukasik, M., 2016. Stance classification in rumours as a sequential task exploiting the tree structure of social media conversations. In: Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers. pp. 2438–2448.

Zubiaga, A., Liakata, M., Procter, R., 2017. Exploiting context for rumour detection in social media. In: International Conference On Social Informatics. Springer, pp. 109–123.