Journal of Ambient Intelligence and Humanized Computing (2021) 12:4315–4333
https://doi.org/10.1007/s12652-019-01527-4

ORIGINAL RESEARCH

Exploring deep neural networks for rumor detection

Muhammad Zubair Asghar[1] · Ammara Habib[1] · Anam Habib[1] · Adil Khan[2] · Rehman Ali[3] · Asad Khattak[4]

Received: 15 February 2019 / Accepted: 26 September 2019 / Published online: 17 October 2019
© Springer-Verlag GmbH Germany, part of Springer Nature 2019

Abstract

The widespread propagation of numerous rumors and fake news have seriously threatened the credibility of microblogs. Previous works often focused on maintaining the previous state without considering the subsequent context information. Furthermore, most of the early works have used classical feature representation schemes followed by a classifier. We investigate the rumor detection problem by exploring different Deep Learning models with emphasis on considering the contextual information in both directions: *forward and backward*, in a given text. The proposed system is based on Bidirectional Long Short-Term Memory with Convolutional Neural Network, effectively classifying the tweet into rumors and non-rumors. Experimental results show that the proposed method outperformed the baseline methods with 86.12% accuracy. Furthermore, the statistical analysis also shows the effectiveness of the proposed model than the comparing methods.

Keywords Rumor detection · Microblogs · Deep learning · BiLSTM · CNN · Social networking services · Twitter

Abbreviations

ML	Machine learning
DL	Deep learning
RNN	Recurrent neural network
LSTM	Long short-term memory
GRU	Gated recurrent unit
BiLSTM	Bidirectional long short- term memory
CNN	Convolutional neural network
KNN	K-nearest neighbors
DT	Decision tree
RF	Random forest
LR	Logistic regression
NB	Naïve Bayes
BOW	Bag of words
TF-IDF	Term frequency-inverse document frequency
CRF	Conditional random fields
SVM	Support vector machines
LSVM	Linear support vector machine

Electronic supplementary material The online version of this article (https://doi.org/10.1007/s12652-019-01527-4) contains supplementary material, which is available to authorized users.

✉ Muhammad Zubair Asghar
zubair@gu.edu.pk

Ammara Habib
ammarahabib10@gmail.com

Anam Habib
anamhabib19@gmail.com

Adil Khan
adil.adil25@yahoo.com; Dradil@hit.edu.cn

Rehman Ali
rehmanali@uop.edu.pk; rahmanac1@gmail.com

Asad Khattak
asad.khattak@zu.ac.ae

[1] Institute of Computing and Information Technology, Gomal University, Dikhan, KP, Pakistan

[2] School of Computer Science and Technology, SZIC, Computer Science Department University of Peshawar, KPK, Peshawar, Pakistan

[3] QACC, University of Peshawar, KPK, Peshawar, Pakistan

[4] College of Technological Innovation, Zayed University, Dubai, United Arab Emirates

1 Introduction

In recent times, social media platforms have emerged as a central place for exchanging information among individuals and groups around the globe. In such platforms, microblogging sites like Twitter are one of the rapid ways of information propagation. Before appearing on the conventional media, most of the breaking news appears first on Twitter (Hamidian and Diab 2015). However, the credibility of such information is still an open challenge (Zubiaga et al. 2018). There is a number of exemplary events about which rumors

or fake news were disseminated and became viral on micro-blogging sites, misleading public opinion, disrupting the social order, decreasing government credibility, and leading to a huge threat to the social stability (Liang et al. 2015). For example, a rumor of an attack on the White House was posted on the hacked Twitter account of Associated Press, resulting in the loss of $136 billion in the stock market within 2 min.[1]

Twitter has been considered as a powerful social networking tool for the politicians to manage the political campaigns and approaching their audience in shortest possible time. For example, in 2016 US elections, Barack Obama and Hillary Clinton were remained as the frequent Twitter users for publicizing their political campaigns (Chang et al. 2016). Although social networking sites like Facebook and Twitter allow their users to access and disseminate the information quickly, however, there is a lack of advanced filtering mechanism to verify the credibility of propagated information, which leads to bullying, rumor propagation, trolling and other unsocial behaviors (Alzanin and Azmi 2018). In politics, the extreme political campaigners can easily create rumors on Twitter and widely spread by readers and followers, who cannot judge their truthfulness (Shao et al. 2017). Rumors also have important political implications on individuals, such as a political rumor circulated about Obama's health care policy in summer 2009 and later this rumor have been discredited by media organizations. The dissemination of rumors and malicious information can have harmful effects also on society, as rumors decrease citizens' trust in the government and support of the regime (Huang 2017). Different researchers have conducted studies (Castillo et al. 2011; Gupta et al. 2014; Jaho et al. 2014) to analyze the credibility of information. However, detecting rumors and false information on Twitter has been considered as a challenging research area (Morris et al. 2012; Liu et al. 2015). Furthermore, automatic identification of political rumors from micro-blogging websites, as well as preventing rumor propagation in early stages, is an important research area."

The work proposed by Ajao et al. (2018) used LSTM-CNN model for tweet (breaking news) classification into rumors and non-rumors (fake vs genuine). The state-of-the-art LSTM-CNN model (Ajao et al. 2018) has an initial unidirectional LSTM layer that only maintains information from the previous context, without retaining the subsequent word (contextual information). It lacks the capability to maintain the context information efficiently to be made an input to the CNN layer. Therefore, the LSTM-CNN model does not provide an efficient way of classifying rumors from online content.

To overcome the aforementioned limitations, we investigate Bidirectional Long Short-Term Memory (BiLSTM) to retain context information both ways in a given tweet by maintaining both the previous and next states at word-level. In this way, the CNN layer receives input from BiLSTM with sufficient context information, minimizing the issue of limited context information associated with the unidirectional LSTM layer for tweet classification into rumors and non-rumors.

In this work, we focus on the problem of classifying a tweet into a rumor or non-rumor. To discriminate rumor tweet from non-rumor, the rumor detection task is considered as a binary classification problem. We define a given set of training data as $D = \{d_1, d_2, d_3, \ldots d_n\} \in \mathcal{R}^{zxm}$ with each row di $\in R^n$ being a data instance and each column $C_i \in \mathcal{R}^z$ is a class label for training data $y \in \{0, 1\}$, if 1, then rumor, otherwise non-rumor. We aim to develop a model which can learn from the given training data and a class label y that accurately classifies rumor and non-rumor based on Twitter-based posts.

In this study, different ML classifiers are experimented such as K-nearest neighbors (KNN), Decision tree (DT), Random forest (RF), Logistic regression (LR), Naïve Bayes (NB) and also different variants of DL models: LSTM, CNN, RNN, LSTM-CNN are implemented. We have used the task-specific word embeddings feature representation scheme for DL classifiers. As the baseline classifiers consist of bag of words (BOW), such as CountVectorizer, and Term Frequency-Inverse Document Frequency (TF-IDF), as the feature representation scheme (Hamidian and Diab 2015).

The proposed system aims at applying DL model, namely BiLSTM-CNN, where the BiLSTM layer is used to learn the long term dependency in a tweet by considering both the previous (past) and next (future) context information. After that, CNN is applied to extract features for efficient classification of the tweet as *rumor and non-rumor.* The BiLSTM-CNN has already been applied in many early works in the domain of Natural Language Processing (Zhang and Xiang 2018; Zeng et al. 2016) but to the best of our knowledge, we have applied the BiLSTM-CNN model in Rumor detection domain, for the first time. Additionally, we compared it with most the other DL models. We want to answer the following research questions: *RQ#1: How to recognize and classify tweets as rumor vs non-rumor, by applying deep learning-based techniques, RQ#2: What is the performance of Word Embedding learned using BiLSTM-CNN over the classical feature sets bag-of-word techniques(BOW) like CountVectorizer, Tf-IDF? and RQ3: What is the performance of the proposed technique for tweet classification into rumor and non-rumor with respect to the baseline methods?*

Following contributions are made in this study:

(i) Classifying tweets as rumor or non-rumor with DL based techniques.
(ii) To investigate the classical feature sets like CountVectorizer, TF-IDF over Word Embedding learned using CNN, LSTM, RNN, and LSTM-CNN for tweet classification as rumor and non-rumor.

[1] http://www.bloomberg.com/news/articles/2013-04-23/dow-jones-drops-recovers-after-false-report-on-ap-twitter-page.

(iii) Development of Web-based interface to detect rumors.
(iv) Comparing the efficiency of the proposed model with other baseline methods.
(v) Our method outperforms baseline methods by a significant margin.

The rest of the article is organized as follows: related work is presented in Sect. 2; Sect. 3 presents proposed methodology; in Sect. 4, we present experimental setup, analyzes the results obtained from experiments and the final section concludes the study and gives the recommendation for the future work.

2 Related work

In this section, we present a review of literature on rumor detection.

2.1 What about the recent fake news idea?

Nowadays the issue of rumors and fake news is viewed as the greatest threats to journalism, democracy, and freedom of expression (Hosseinimotlagh and Papalexakis 2018). It has a significant impact on various aspects of life such as weakening the public trust in government, employed in election campaigns against the famous figures, and economically, affecting the consumption of products and food (Conroy et al. 2015). Fake news has got a huge potential to produce a real-world impact in a very short time period, manipulating the public perception significantly. The fake news is published intentionally to mislead readers (Long et al. 2017; Asghar et al. 2019b). For instance, on August 25th of 2015, fake news about "shootouts and kidnappings by drug gangs happening near schools in Veracruz" spread through Twitter and Facebook. This caused severe chaos in the city involving 26 car crashes because people left their cars in the middle of a street and rushed to pick up their children from school. This incident of fake news highlights that automatically predicting the veracity of the information on social media is of high practical value (Ma et al. 2016).

2.2 Why rumors are problematic?

Rumors and fake news detection in social media text is a challenging problem it is even difficult for a human to accurately distinguish between real news and fake news. As reported by the Forbes,[2] only 50–63% success rate is achieved by human judges in identifying the fake news using a manual inspection method. In another work (Zhou and Zafarani 2018), it is observed that 80% of the high school students had faced a hard time in determining, whether an article is fake or original. While conducting experiments on fake reviews identification, it has been observed the fake news data is limited because the fake news is usually mixed with true stories and it becomes difficult to recognize them accurately (Pham 2018).

2.3 What are the previous studies that researched about rumors?

In the microblogging platform such as Twitter, various issues have been investigated including spam detection (Ahmed and Abulaish 2012), sentiment detection (Barbosa and Feng 2010) and event detection (Kimmey 2015).The research in rumor detection in Twitter is less deeply explored so far, although in social psychology (Allport and Postman 1947) rumors have already been investigated for a long time. In one of the early studies on rumor detection (Castillo et al. 2011), authors have analyzed the information credibility in Twitter Posts by extracting 68 features categorized them into four types namely (i) user-based features, (ii) content-based features, (iii) propagation- based features, and (iv) topic-based features. With the similar objective, the task of misinformation identification in microblogs is carried out by Qazvinian et al. (2011) to extract the attributes related to the different constructs, such as the content of tweets, network features, and Twitter specific memes. Different Bayes classifiers are developed to detect the rumor spreading on Twitter. While highlighting the importance of information propagation on Twitter, the studies conducted by Seo et al. (2012), Tripathy et al. (2010) has focused on the detection of rumors from Twitter posts. The study conducted by Tripathy et al. (2010) applied logistic regression classifier using a small amount of provenance information. Feature engineering also plays an important role in rumor detection, in this connection, Kwon et al. (2013) proposed a set of text features identified from the comments and retweet for efficient detection of rumors. Use of temporal traits for identification of rumor has been focused by Ma et al. (2015). They introduced a time series fitting model by exploiting three sets of features, namely structural, temporal and linguistic. This work is extended by Yang et al. (2012) using dynamic time series model to capture the temporal characteristics of the social context features. In the work proposed by Yang et al. (2012), two new features are investigated: (i) location-based feature, and (ii) client-based features, and the Support Vector Machine classifier is trained to identify the misinformation on the microblog Sina Weibo. Another work performed by Zhou and Zafarani (2018), utilized several classical features, namely user, propagation, and other Meta features along with the belief of the crowd. In a similar work, Wu et al. (2015) developed a graph-kernel-based SVM classifier along

[2] https://www.forbes.com/sites/brettedkins/2016/12/20/americans-believe-they-can-detect-fake-news-studies-show-they-cant/#5fd9a07c4022.

with three sets of features: user, message, and repost, for the detection of the false rumors in microblogging sites.

Liu et al. (2019) proposed an Attention-Based BiGRU-CNN network for Chinese question classification task. The system yielded improved results over the state-of-the-art works. However, the performance can be further improved by using different types of features as input to deep learning model. Edara et al. (2019) applied several text mining and machine learning techniques on social media text collected from cancer supported communities to perform sentiment analysis. The work proposed long short-term memory (LSTM) neural network as an alternative to the conventional sentiment analysis techniques. Experimental results proved that the proposed method performed better in terms of both accuracy and execution time among the other methods. Liu (2018) proposed a text sentiment analysis framework based on the Bag of Word(CBOW) model and deep learning. The proposed system was experimented on two datasets COAE2014 and IMDB. Results depict the proposed method can accurately determine the emotional category of the text.

The aforementioned studies have applied different approaches for the identification and classification of rumors. These techniques are based on the supervised machine learning, unsupervised learning, deep learning and hybrid models.

Most of the aforementioned existing studies, mainly focus on the manual feature engineering methods, followed by a machine learning (ML) classifier (Jin et al. 2017a; Yang et al. 2015; Ma et al. 2016). Furthermore, existing studies conducted on rumor detection using deep learning (DL) techniques, applied different models such as recurrent neural network (RNN), long short-term memory (LSTM), gated recurrent unit (GRU) (Duong et al. 2017; Chen et al. 2018; Ruchansky et al. 2017; Veyseh et al. 2017), temporal attention (att) CNN-LSTM (Chen et al. 2017a, b), att- RNN (Jin et al. 2017b), Convolutional Neural Network (CNN) (Yu et al. 2017a, b) and LSTM-CNN (Ajao et al. 2018 In the field of NLP, both long short term memory (LSTM) and convolutional neural networks (CNN) have shown promising results. Different studies (Zhang and Xiang 2018; Ayutthaya and Pasupa 2018; Chiu and Nichols 2016) conducted in this area indicated that the BiLSTM-CNN has exhibited best performance with respect to other models. Therefore, we investigate the applicability of the deep learning model, namely BiLSTM + CNN for rumor detection in the online content. We applied deep neural network BiLSTM-CNN model on different datasets.

3 Materials and methods

The proposed method consists of different modules namely (i) dataset acquisition and splitting, (ii) overview of the method, (iii) network architecture, and applying an example. The detail of each module is described below.

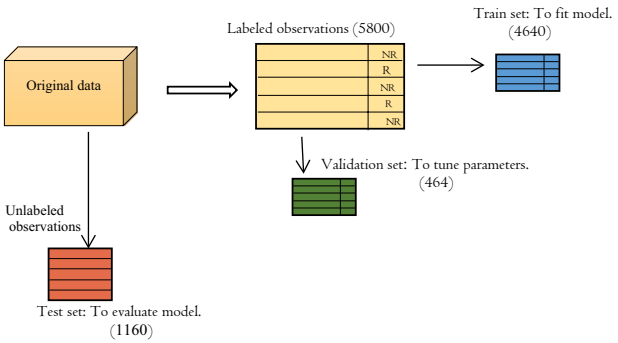

Fig. 1 Train, validation and test split

3.1 Dataset acquisition and splitting

We acquired a benchmark Pheme rumor dataset.[3] The dataset consists of 5800 tweets about five breaking news events.

We split the dataset into three parts: (i) Train data, (ii) Validation data, and (iii) Test data.

The proposed deep learning models are implemented with the Keras library[4] built on TensorFlow using, 64-bit OS with an 8 GB memory, and Intel Core i7. Figure 1 depicts the diagrammatic representation of train, validation and test split.

3.1.1 Training data

The training of the proposed model is performed using 70% of the data as a training set, which can vary for different experiments. The model learns from the train data and used to fit the model. It includes both the input and the corresponding expected output (Asghar et al. 2019a).

A sample output of the Training phase is presented in Table 1.

In addition to the aforementioned dataset, we also used two more public datasets for further experimentation: The dataset D2 is a merger of two datasets,[5][6] which contains two types of news, namely; fake news (size = 2999) and real news (size = 4320). The dataset contains a csv file, namely "fake-news", in which majority of the news focus on political topics. Similarly, the dataset D3[7] is comprised of fake news (size = 3160) and real news (size = 3165). The dataset contains different types of topics, such as world news topics and politics in the file namely "fake_real_news" (Table 2).

[3] https://figshare.com/articles/PHEME_dataset_of_rumours_and_non-rumours/4010619.

[4] https://keras.io/.

[5] https://www.kaggle.com/jruvika/fake-news-detection.

[6] https://github.com/alumag/ADAFake/tree/master/data.

[7] https://www.kaggle.com/rchitic17/real-or-fake.

3.1.2 Validation data

During the training phase, the accuracy of training is often high, showing a high level of accuracy but the performance gets degraded when tested against the test set. Therefore, to avoid the performance error in terms of overfitting and underfitting, we used 10% validation set. To ensure the optimal functioning of the model, Keras provides two techniques for parameter tuning (Acharya 2017), namely: (i) manual verification of the dataset, and (ii) automatic verification dataset. We applied automatic verification dataset because it provides an unbiased evaluation of the model and assists in minimizing the problem of overfitting.

3.1.3 Test data

The test data assists in evaluating the performance of the model on unseen/new data. We used 20% test data, completely independent of the training set. It is used only once when the model is completely trained (using trained and validation set). The final evaluation of the model is performed through the test data. A sample listing of the Testing data is presented in Table 3.

The statistics of the dataset is shown in Table 4.

Firstly, the dataset is segmented into 80:20 split by using the train-test split method of scikit-learn. In the next step, we dissect the train set into 70:10 split, in which the 10% is the validation set. The validation set is used for different purposes, such as tuning the model hyperparameters, to configuring and evaluating the model.

3.2 Overview of the method

The proposed method is comprised of different modules (Fig. 4), namely (i) input data preprocessing, (ii) feature representation, (iii) feature encoding, (iv) feature extraction, and (v) classification (label predictor). Specifically, the following module is included in our method:

- *Input data preprocessing text preparation*: In this step, we apply, stop word removal, depletion to lowercase, and data splitting (tokenization). After tokenization, a unique integer is assigned to each word and resultantly, an integer-based sentence vector (set of integer/indexes) is created. For example, the input sentence: "sad news plane crashed southern France", is transformed into an integer vector, based on a unique value assigned to each word (Fig. 2).
- *Feature representation:* In this step, the integer vector generated for a sentence in the previous stage (Fig. 2) is transformed into a dense vector of fixed size (Yang et al. 2015) using Keras embedding layer. The dense vector is a real-valued encoding (Fig. 3).

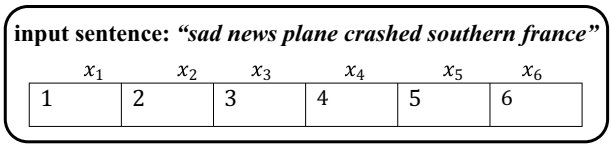

Fig. 2 Integer vector representation of a sentence

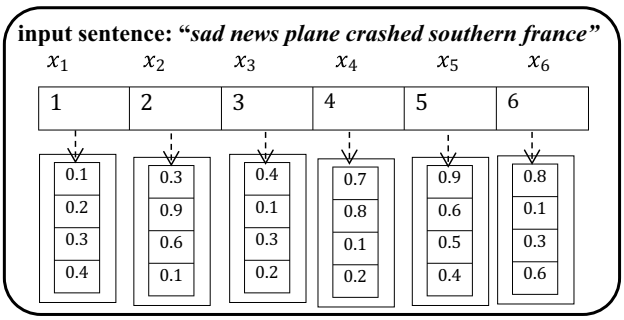

Fig. 3 Dense vector representation of a sentence

- *Feature encoding:* A BiLSTM is used to perform double ended encoding of a sentence, i.e. a sentence is encoded from beginning to end and then from end to beginning, receiving two representations. Finally, the two representations are merged to form a final encoded representation (Zhang et al. 2018). The dually encoded representation captures both previous and forward dependencies. The information captured (sentence matrix) is made an input to the CNN layer.
- *Feature extraction:* The CNN layer performs feature extraction (e.g. n-grams) by processing the information in a hierarchical manner.
- *Classification (Label predictor):* Finally, classification is done by applying sigmoid activation function to the classify the input sentence/tweet into rumor or non-rumor.

3.3 Network architecture

We trained BiLSTM-CNN classifier to classify a tweet into rumor and non-rumor. Our network process information into the following five steps;

The proposed BiLSTM-CNN model for tweet classification into rumors and non-rumors operates in eight layers: embedding (sentence matrix), Dropout layer(sentence matrix), BiLSTM (sentence matrix), Convolution (sentence matrix), Maxpool (sentence matrix), flatten (feature vector), and single neuron output(classification). Figure 5 shows an end-to-end model with different layers, explained as follows:

Embedding layer: We represent the dataset as a set of multiple tweets t and each tweet t is comprised of a sequence of z words, i.e. $x_1, x_2, x_3 \ldots x_z$. Each word x_i constitutes a real valued embedding vector $w_i \in \mathcal{R}^m$,

Table 1 Set of tweets from the dataset (training)

Tweet #	Tweet	Tweet label (R/NR)
1.	Ottawa police are confirming a shooting at the War Memorial. Minutes ago. No other info	Non-rumor
2.	Shooting at the war memorial in Ottawa	Non-rumor
3.	Breaking: Police confirm multiple suspects in Ottawa shooting	Rumor
4.	At least 30 more shots were fired inside Parliament in Ottawa	Rumor
5.	Police with guns drawn at seen of war memorial shooting	Non-rumor
6.	#BREAKING: PMO says PM Harper is safe and has left Parliament Hill	Non-rumor
8.	Gunman on the loose in downtown Ottawa. Canadian citizens disarmed by their government. Target rich environment. #OttawaShooting	Rumor
10.	Canada Prime Minister Stephen Harper safe has left Parliament Hill: TV quoting PM's office	Non-rumor

Table 2 Description of dataset D1, D2, D3

Dataset	Description	Size	Remarks
D1	Pheme rumor dataset	5800	Publically available benchmark datasets
D2	Fake-news	7319	
D3	Fake_real_news	6325	

Table 3 Set of tweets from the dataset (testing)

Tweet#	Tweet	Tweet label (R/NR)
7.	Uniformed Canadian soldier shot at War Memorial in #Ottawa	Rumor
9.	#Ottawa City Hall is currently in lock-down. Please avoid the area	Non-rumor

Table 4 Statistics of the dataset

Original set (5800)/100%		
Train set (4640)/80%		Test set (1160)/20%
Train set (4176)/70%	Validation set (464)/10%	Test set (1160)/20%

known as word embedding. In this work, we used keras embedding layer (Duong et al. 2017). The embedding layer is comprised of two dimensional input matrix, also called embedding matrix/sentence matrix, represented as: $E \in \mathcal{R}^{z \times m}$, where z is the tweet length, and m is the embedding dimension. In this way, the, the matrix E is made input to the BiLSTM layer.

The mathematical notations used in Embedding Layer are listed in Table 5.

BiLSTM layer: The BiLSTM layer is a special kind of RNN, capable of learning long term dependency (Alayba et al. 2018).It assists in accessing both previous(left) and next (right) context of an encoded tweet (sequence labeling). However, in a unidirectional LSTM, the hidden states (h_t)

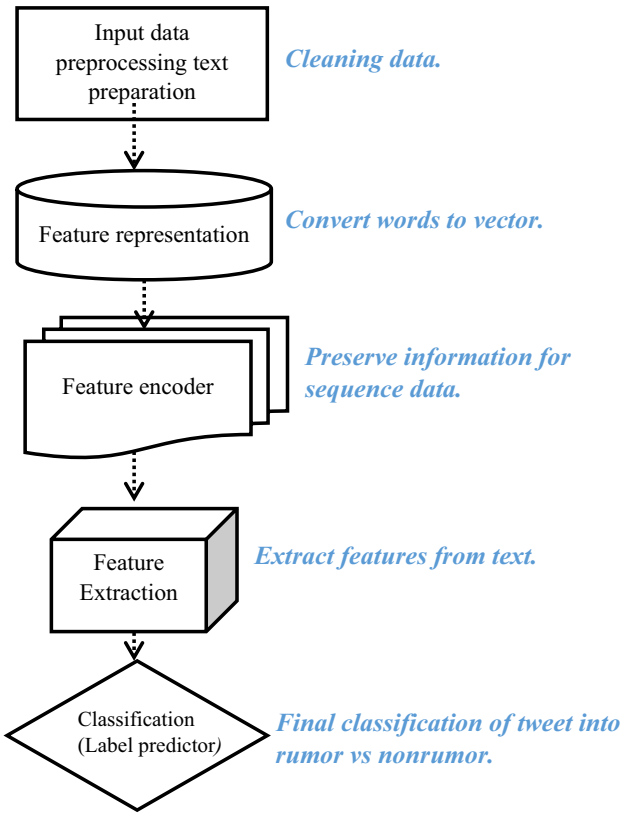

Fig. 4 Block diagram for method overview

take into account only the previous (past) information, without considering the next(previous) information. Therefore, the BiLSTM keeps track of richer information required for processing the encoded tweet, sequence labeling (Ma and

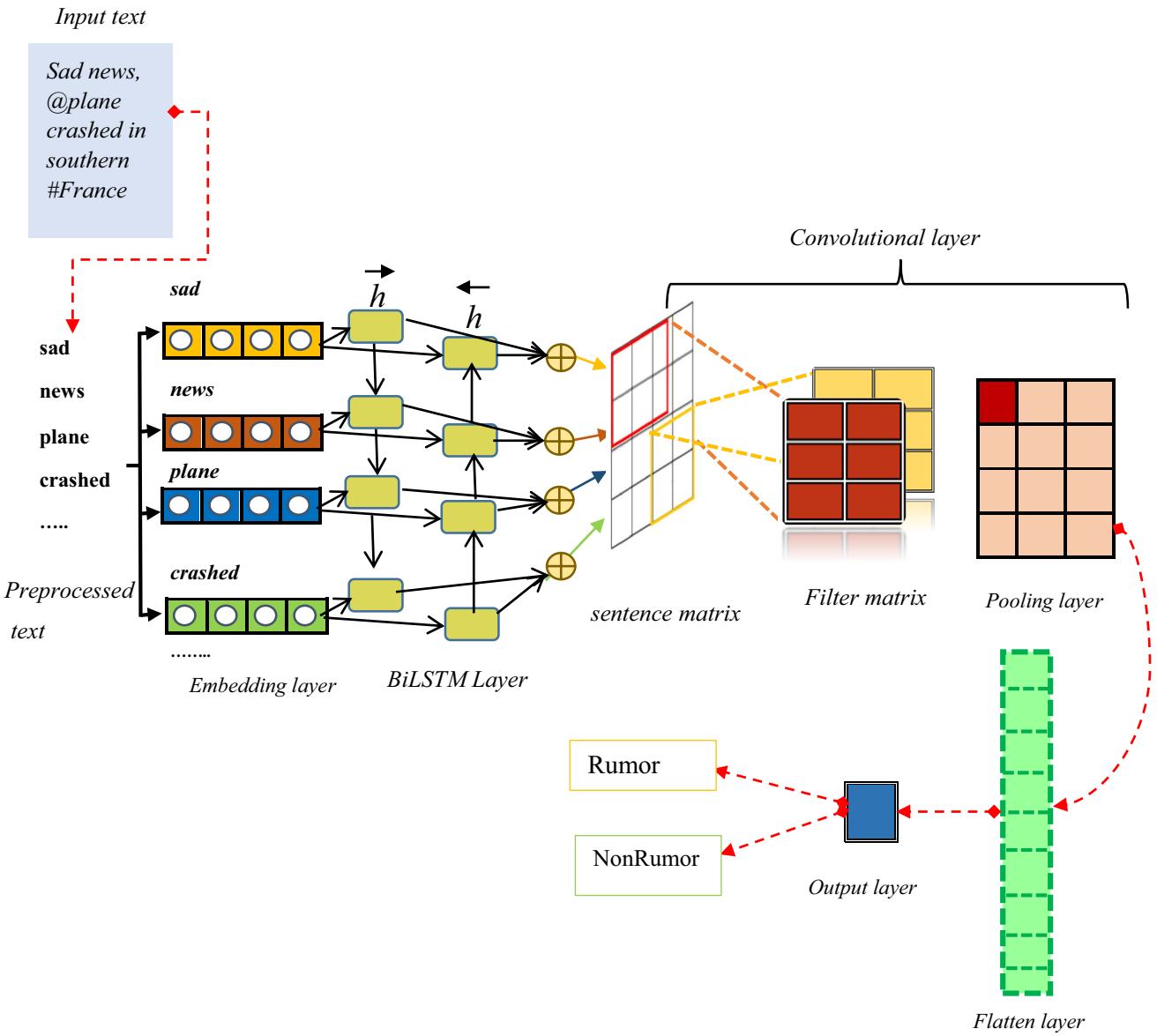

Fig. 5 Network architecture

Table 5 Symbols used in the embedding layer

Mathematical symbol	Description
t	Tweet
w_i	Embedding vector
E	Sentence matrix generates from Embedding layer
R	Real numbers

Hovy 2016). To learn information from preceding as well as subsequent tokens, BiLSTM incorporates a Forward LSTM layer and a Backward LSTM layer (Zhou et al. 2016).

Forward LSTM: It processes the sequence from left to right by concatenating the two inputs, i.e. the current input "x_1", and the previous input (hidden state) "h_{t-1}". For a given input sequence: $x_1, x_2, \ldots x_{z-1}$ the Forward LSTM layer produces an output sequence "$\vec{h}$".

Backward LSTM: It processes the sequence from right to left by concatenating the two inputs: the current input "$s1$" with the next input (future hidden state) "h_{t+1}". For a given input sequence: $x_{z+1}, \ldots, x_2, x_1$, the Backward LSTM layer produces an output sequence "$\overleftarrow{h}$".

The forward and backward context representations, denoted by "$\vec{h}$" and "$\overleftarrow{h}$" respectively, are merged to produce a new sentence matrix $H = [h_1, h_2, h_3, \ldots ..h_z]$, $H \in \mathcal{R}^{z \times m}$

(Zhou et al. 2016). To combine forward and backward output, the element-wise sum is computed as follows (Eq. 1):

$$\overleftrightarrow{h} = \vec{h} \oplus \overleftarrow{h} \tag{1}$$

Finally, the new sentence matrix (Eq. 1) is made an input to the CNN layer.

Following equations are used for the computation of forward and backward LSTM.

Forward LSTM Gates [5]

$$f_t = \sigma\left(W_f x_t + U_f h_{t-1} + b_f\right) \tag{2}$$

$$i_t = \sigma\left(W_i x_t + U_i h_{t-1} + b_i\right) \tag{3}$$

$$o_t = \sigma\left(W_o x_t + U_o h_{t-1} + b_o\right) \tag{4}$$

$$c \sim t = \tau\left(W_c x_t + U_c h_{t-1} + b_c\right) \tag{5}$$

$$c_t = f_t \odot c_{t-1} + i_t \odot c \sim_t \tag{6}$$

$$h_t = o_t \odot \tau\left(c_t\right) \tag{7}$$

Backward LSTM Gates

$$f_t = \sigma\left(W_f x_t + U_f h_{t+1} + b_f\right) \tag{8}$$

$$i_t = \sigma\left(W_i x_t + U_i h_{t+1} + b_i\right) \tag{9}$$

$$o_t = \sigma(W_o x_t + U_o h_{t+1} + b_o \tag{10}$$

$$c \sim t = \tau\left(W_c x_t + U_c h_{t+1} + b_c\right) \tag{11}$$

$$c_t = f_t \odot c_{t+1} + i_t \odot c \sim_t \tag{12}$$

$$h_t = o_t \odot \tau\left(c_t\right) \tag{13}$$

Where n is the size of the input, m is the size of cell state, x_t: input vector of size $n \times 1$, f_t: forget gate vector of size $m \times 1$, i_t: input gate vector of size $m \times 1$, o_t: output gate vector of size $m \times 1$, h_t: output vector of size $m \times 1$, c_t: cell state vector of size $m \times 1$. The W_f, W_i, W_o, and W_c, represent input gate weight matrices of size $m \times n$ n. U_f, U_i, U_o, and U_c represent output gate weight matrices of size $m \times m$. b_f, b_i, b_o, and b_c represent bias vectors of size $m \times 1$. σ shows logistic sigmoid activation function and finally the, τ shows hyperbolic tangent function.

Each of these gates is considered as separate modules within the BiLSTM, performing different functions. The input gate (i_t) determines how much emphasis to put on each of the inputs, the forget gate (f_t) discards the unnecessary information, and the output gate computes the final output h_t.

The mathematical notations used in BiLSTM Layer are listed in Table 6.

Table 6 Symbols used in BiLSTM

Mathematical symbol	Description
x_t	Input vector
f_t	Forget gate vector
i_t	Input gate vector of size
o_t	Output gate vector
h_t	Hidden state
h_{t-1}	Previous hidden state
h_{t+1}	Future hidden state
$\overleftrightarrow{h}$	Final representation (element-wise sum of previous and future hidden state)
c_t	Cell state vector
W_f, W_i, W_o, and W_c	t input gate weight matrices
U_f, U_i, U_o, and U_c	Represent output gate weight matrices
b_f, b_i, b_o, and b_c	Represent bias vectors
σ	Shows logistic sigmoid activation function
τ	Shows hyperbolic tangent function
f	A nonlinear function
H	Sentence matrix generated by BiLSTM layer

CNN layer The CNN layer is comprised of different layers, namely convolution, maxpool, and flatten, generating a feature vector. Detail of each layer is presented as follows:

Convolution: A convolutional operation involves a convolution filter matrix MR^{kxd} (Zhou et al. 2016). To generate a feature map, a filter matrix M is applied to each possible window of words over the matrix H of the previous BiLSTM layer. For example, a feature map U is learned as follows:

$$U_{ij} = f\left(W \circ s_{i:i+k-1,j+d-1} + b\right) \tag{14}$$

where i ranges from 1to$(z-k+1)$ and j ranges from 1to$(m-d+1)$, bR is bias term, "$\circ$" is convolutional operation between M and E, and f is a nonlinear function. We used the Relu function because of its better performance than the other functions like Tanh (Zhang et al. 2017).

After performing the convolution operation (Eq. 14), the matrix U is obtained as follows: $U = \left[u_{1,1}, u_{1,2}, \ldots u_{z-k+1,m-d+1}\right], UR^{z-k+1}$.

Maxpool: The size of the feature map (U) is further reduced by applying the maxpool layer by selecting the important features (max: value). This assists in reducing the computation time by discarding the non-maximal (unimportant features).

The maxpool operation is computed as follows:

$$O_{ij} = \max\left(o_{i+k-1,j+d-1}\right). \tag{15}$$

After applying Eq. 15, the pooled feature matrix is obtained as follows: $O = \left[o_{1,1}, o_{1,2}, \ldots o_{u-k+1,v-d+1}\right], O \in R^{u-k+1,v-d+1}$.

Table 7 Symbols used in CNN

Mathematical symbol	Description
M	Convolution filter matrix
U	Feature map
O	Pooled feature matrix

The mathematical notations used in CNN Layer are listed in Table 7.

Flattening The Flatten layer is used to transform the pooled feature map obtained in the previous stage, into a feature vector, in order to prepare the input for the final classification layer. For this purpose, the elements or the features of the pooled feature matrix O are transformed into a flattened vector produced by the reshaping operation (Lopez-Martin et al. 2017) as follows:

$$T = pooled.reshape[(u - k + 1) \times (v - d + 1)] \tag{16}$$

Classification For the final classification, we use Dense layer with a single neuron and a sigmoid activation function, to compute the probability of the two classes i.e. *rumor and nonrumor.*

The net input is obtained by applying the Eq. 17, as follows:

$$y_{in} = \sum x_i.w_i + b \tag{17}$$

We take an example tweet and provide a step-wise detail explaining how different layers of the proposed BiLSTM-CNN model operated for classifying the tweet as rumor or non-rumor (Fig. 6).

3.3.1 Input preparation and embedding layer

We take a sample Tweet: "*sad news plane crashed southern France*", and apply the proposed model for its classification as *rumor or non-rumor.* Firstly, the input tweet is segmented into a set of tokens as follows: *"sad", "news", "plane", "crashed", "southern", "France*". Next, we build the vocabulary index which is a mapping of the index to each unique word. As there are 6 unique tokens the vocabulary is 6, represented as: *"sad:* 1*", "news:* 2*", "plane:* 3*", "crashed:* 4*", "southern:* 5*", "France:* 6*"*. After that, the given tweet is transformed into a sequence of integers i.e. [1, 2, 3, 4, 5, 6]. Next each word with a single index in the tweet is transformed into a vector using an embedding layer, represented as follows: [0.10.20.30.4],[0.30.90.60.1], where the first row is an embedding vector for the token *"sad"* and the second row represents the token *"news*". The process is repeated for such tokens. Resultantly, a matrix is constituted as follow; [[0.10.20.30.4],[0.30.90.60.1], [0.40.10.30.2],[0.70.80.10.2] [0.90.60.50.4], [0.80.10.30.6]].

3.3.2 Dropout layer

The functionality of the Dropout Layer is to prevent overfitting. We set the value of "rate" parameter to "0.5", where its value ranges between0 and 1. The Dropout layer is applied after the embedding layer, so it randomly turns off or removes the activation of neurons in the embedding layer (Shen et al. 2017).

3.3.3 BiLSTM layer

The unidirectional LSTM can only get information from the previous context. But most of the sequence data, especially in the classification task, depending on overall information including past and future. Therefore, we choose to use bidirectional architecture for processing the sequence in two directions: left-to-right (Forward LSTM) and right-to-left (Backward LSTM), yielding overall information of the sequence (Zhang and Xiang 2018).

The input received from the Dropout layer is made an input to the BiLSTM layer. The computation is comprised of 4 components, namely an input gate (it), a forget gate (ft), an output gate (ot), and a new memory container ($c \sim t$).

3.3.4 Forward LSTM

In Forward LSTM each gate takes a current input (x_t), previous state (h_{t-1}), performs some computation (Eqs. 2–7), and finally information is aggregated in the form of hidden state "$\vec{h}$", show as follows:

$$\begin{matrix}\vec{h}\\ \begin{bmatrix}0.6\\0.4\\0.5\\0.7\end{bmatrix}\end{matrix} = \begin{matrix}o_t\\ \begin{bmatrix}1\\1\\1\\1\end{bmatrix}\end{matrix} \odot \tau \begin{matrix}c_t\\ \begin{pmatrix}0.7\\0.4\\0.6\\0.8\end{pmatrix}\end{matrix}.$$

3.3.5 Backward LSTM

In backward LSTM, each gate takes current input (x_t), future state (h_{t+1}), performs some computation (Eqs. 8–13), and aggregate the hidden state "$\overleftarrow{h}$", as follows:

$$\begin{matrix}\vec{h}\\ \begin{bmatrix}0.3\\0.2\\0.2\\0.1\end{bmatrix}\end{matrix} = \begin{matrix}o_t\\ \begin{bmatrix}1\\1\\1\\1\end{bmatrix}\end{matrix} \odot \tau \begin{matrix}c_t\\ \begin{pmatrix}0.4\\0.3\\0.3\\0.2\end{pmatrix}\end{matrix}.$$

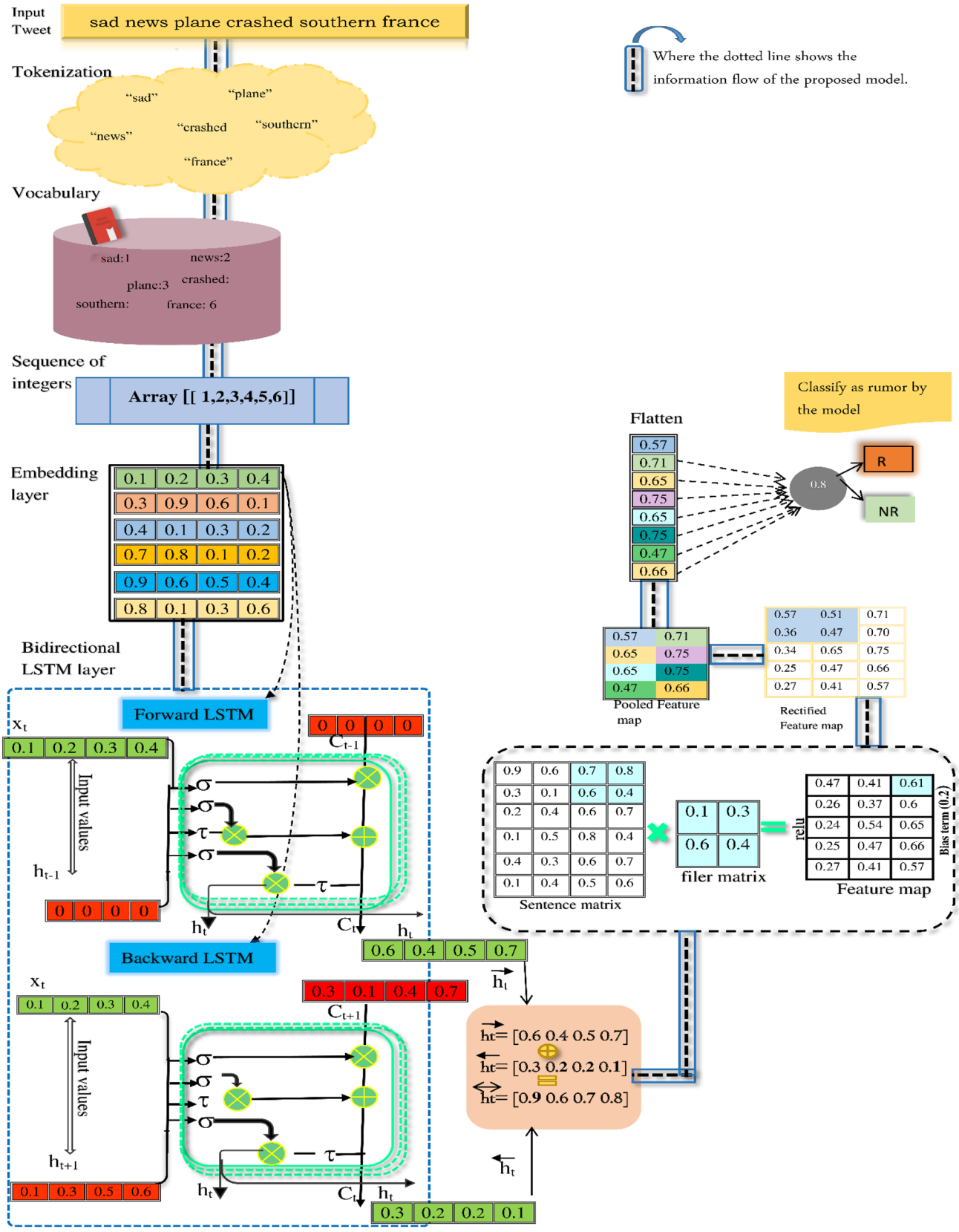

Fig. 6 Working flow of proposed BiLSTM-CNN model for an example sentence

3.3.6 The final output of BiLSTM Layer

Finally, the forward LSTM $\vec{h}$ and backward LSTM $\overleftarrow{h}$ are accumulated (element-wise summation) using Eq. 1, to get the final representation $\overleftrightarrow{h}$, represented as follows:

$$\begin{matrix}\overleftrightarrow{h} & & \vec{h} & & \overleftarrow{h} \\ \begin{bmatrix}0.9\\0.6\\0.7\\0.8\end{bmatrix} & = & \begin{bmatrix}0.6\\0.4\\0.5\\0.7\end{bmatrix} & \oplus & \begin{pmatrix}0.3\\0.2\\0.2\\0.1\end{pmatrix}.\end{matrix}$$

3.3.7 CNN layer

The input received from the previous layer (BiLSTM) is made an input to the convolutional layer for extracting local n-gram features. It works as follows:

3.4 Stage-1: Filtering

In the filtering step, the filter matrix is convolved over the input matrix to generate the convolved feature map (Eq. 14). The filtering stage is comprised of the following three steps, namely: (i) alignment of sentence matrix with filter matrix, (ii) element-wise multiplication of the filter matrix with the selected patch of the sentence matrix, and (iii) Computation of the feature u_{12} by summing all the values obtained in the previous step.

- ***Feature matrix Row1:*** 0.7, 0.8, 0.6, 0.4
- ***Filter matrix:*** 0.3, 0.1, 0.4, 0.2
- ***Feature map Element1:*** 0.7×0.3, 0.8×0.1, 0.6×0.4, $0.4\times0.\ 2=0.61$
- ***Adding bias:*** $0.61+0.1=0.71$
- ***Apply Relu Activation function:*** Output = max (0, 0.71) as $0.71>0$ => **0.71**

3.5 Stage-2: Pooling

In this step, the convolved feature map obtained from the previous stage is reduced using Eq. 15. The pooling process is carried out by applying the following steps: (i) an appropriate window size (commonly 2 or 3) is selected, (ii) *choosing a stride:* The stride controls the movement of window/filter around the Rectified Feature map by shifting the window with a specific threshold/size (stride = 1, in our example), and (iii) A maximum value is selected as output from the window/filter obtained from the previous step.

- ***Max pool for Element1:*** max (0.57, 0.51, 0.36, 0.47) => the maximal number found is 0.57.

3.6 Stage-3: Flattening

Applying Eq. 16, the pooled feature matrix obtained from the previous stage is transformed into a feature vector.

3.6.1 Classification layer

The feature vector received from the previous layer(CNN) is made an input to the classification layer, in which a sigmoid activation function is applied to compute the probability of the classes (rumor and non-rumor).

For the BiLSTM-CNN model, the net input can be calculated (using Eq. 17) as follows:

$$yin = s1.w1 + s2.w2 + s3.w3 \ldots sz.wz + b$$

Putting the values of s $1 - s8(s1 = 0.57, s2 = 0.71, s3 = 0.65, s4 = 0.75, s5 = 0.65, s6 = 0.75, s7 = 0.47, s8 = 0.66)$, $w1 - w8(w1 = 0.3, w2 = 0.2, w3 = 0.5, w4 = 0.1, w5 = 0.4, w6 = 0.6, w7 = 0.8, w8 = 0.1)$, *and* $b = 0.2$ we get:

$$\begin{aligned} y_{in} &= 0.57 * 0.3 + 0.71 * 0.2 + 0.65 * 0.5 + 0.75 * 0.1 \\ &\quad + 0.65 * 0.4 + 0.75 * 0.6 + 0.47 * 0.8 \\ &\quad + 0.66 * 0.1 + (0.2) => y_{in} = 2.57. \end{aligned}$$

Applying sigmoid activation function: $y = F(y_{in})$, where y is the output. F is the sigmoid activation function, and yin is the net input computed (2.57). The computation of y takes place as follows:

$$=> y = {}^{1}/_{1 + e^{-y_{in}}}$$

$$= {}^{1}/_{1 + e^{-2.57}}$$

$$\boldsymbol{y} = 0.8$$

Taking "y" as probability, we formulate the decision rule as follows.

$$f(x) = \begin{cases} 1\,(rumor), & y > 0.5 \\ 0\,(non-rumor), & otherwise \end{cases}$$

Using the aforementioned computation, $y = 0.8 > 0.5$, therefore, the input tweet: "*sad news plane crashed southern france*" is predicted as "*rumor*".

4 Experiment results and evaluation

In this section, we present the detail of the experimental setup design for the proposed model, along with a detailed answer to each of the posed research question.

Table 8 Parameters settings of the proposed BiLSTM-CNN

Parameter	Value
Vocabulary size	10,000
Input vector size	40
Embedding dimension	128
BiLSTM unit size	280, 230, 215, 200,170,127, 120,50,30, 20
Number of convolutional layers	1
Number of hidden layers	7
Number of filters	8, 10, 12, 32,64
Filter size	2, 3
Dropout	0.5
Activation function	Sigmoid
Number of epochs	3
Batch size	32

Table 9 Parameter setting for 10 variations of BiLSTM-CNN models

Model name	No. of filters	BiLSTM unit size	Filter size
BiLSTM-CNN(1)	64	20	3
BiLSTM-CNN(2)	64	30	2
BiLSTM-CNN(3)	32	50	3
BiLSTM-CNN(4)	32	120	2
BiLSTM-CNN(5)	12	127	3
BiLSTM-CNN(6)	12	170	2
BiLSTM-CNN(7)	10	200	3
BiLSTM-CNN(8)	10	215	2
BiLSTM-CNN(9)	8	230	3
BiLSTM-CNN(10)	8	280	2

4.1 Answers to posed research questions

4.1.1 Answer to RQ#1: How to recognize and classify tweets as rumor vs non-rumor, by applying deep learning-based techniques?

We applied different BiLSTM-CNN models for classifying tweets into rumors and non-rumors using different parameters. Inspired by the work conducted by Rani and Kumar (2018) on deep learning based sentiment analysis.

Table 8 shows the parameters setting for the proposed BiLSTM-CNN model. We performed experimentation with varying parameters: a number of filters are varied from 8 to 64, BiLSTM unit is varied from 20 to 278, and different filer sizes are used, such as 2×2, and 3×3. We used certain parameters with fixed sizes, such as vocabulary size, input vector size, embedding dimension, activation function, batch size, dropout, and a number of epochs. In Table 9, we present the configuration setting of the selected parameters (number of filters, BiLSTM unit size, and Filter size) for the 10 BiLSTM-CNN models.

Table 10 Test accuracy, loss and training time of BiLSTM-CNN models

Model name	Test accuracy (%)	Test loss	Training time (s)
BiLSTM-CNN(1)	84.91	0.37	16
BiLSTM-CNN(2)	84.48	0.36	18
BiLSTM-CNN(3)	84.66	0.36	20
BiLSTM-CNN(4)	85.17	0.35	29
BiLSTM-CNN(5)	83.28	0.40	48
BiLSTM-CNN(6)	83.79	0.44	64
BiLSTM-CNN(7)	84.40	0.36	53
BiLSTM-CNN(8)	85.09	0.35	58
BiLSTM-CNN(9)	85.17	0.39	64
BiLSTM-CNN(10)	86.12	0.35	69

In all models, BiLSTM unit size has been varied along with the filter size and number of filter size. Table 9 shows the configuration settings of all 10 BiLSTM-CNN models.

After experimentation with different parameter settings of the BiLSTM-CNN models, we recorded the test accuracy, loss score, and training time, as listed in Table 10. It is observed that the BiLSTM-CNN model (BiLSTM-CNN (10)), with BiLSTM unit size = 278 (neurons), filter size = 2, and the number of filter = 8, performed better and achieved maximum accuracy of 86.12%. Why our results are better. The experimental results suggested that through proper training, the BiLSTM-CNN (10), can outperform the other variants of DL models. It is also observed that the accuracy of the model is increased by increasing the number of neurons and by reducing the number of filters.

The average test accuracy and loss score of all BiLSTM-CNN models is shown in Fig. 7a, b. In Fig. 7a, X-axis denotes training time and the y-axis denotes the test accuracy, whereas in Fig. 7b, X-axis denotes the training time and y-axis denotes the loss score.

In Fig. 7a, with the increase in training time, there is a gradual increase in the test accuracy and the learning curve shows that the models are learning from data, whereas in Fig. 7b, the learning curve shows that the errors are decreasing as the number of training steps is increased.

The performance evaluation measures, such as precision, and recall (Table 11) are obtained for different variations of the BiLSTM-CNN models, as listed in Table 10. The computations of precision and recall, are shown as follows (Eq. 18, 19)

$$precision = Truepoistive/Truepositive + Falsepositive \tag{18}$$

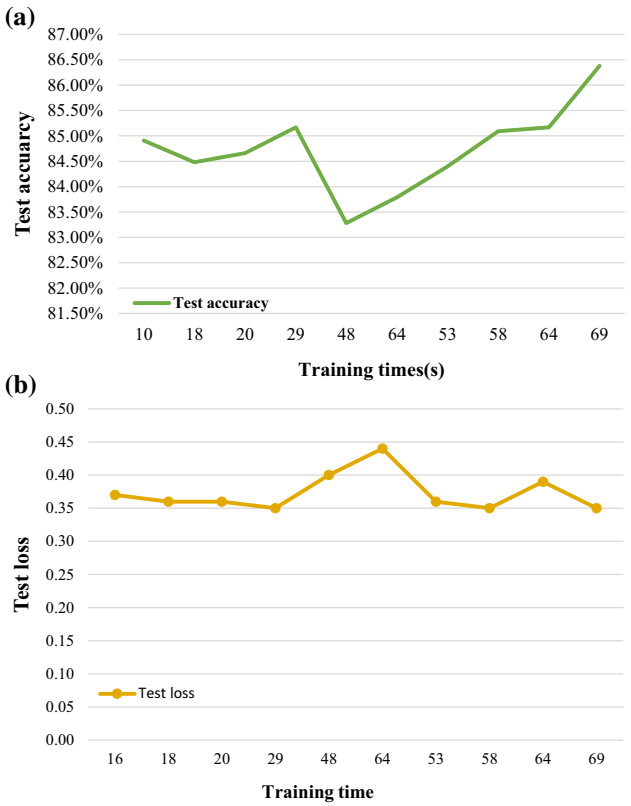

Fig. 7 **a** Average learning curve of accuracy. **b** Average learning curve of loss

Table 11 The performance measure of BiLSTM-CNN models

Model name	Precision	Recall	F-score
BiLSTM-CNN(1)	0.85	0.85	0.85
BiLSTM-CNN(2)	0.84	0.84	0.84
BiLSTM-CNN(3)	0.84	0.85	0.85
BiLSTM-CNN(4)	0.85	0.85	0.85
BiLSTM-CNN(5)	0.85	0.83	0.83
BiLSTM-CNN(6)	0.84	0.84	0.84
BiLSTM-CNN(7)	0.84	0.84	0.84
BiLSTM-CNN(8)	0.85	0.85	0.85
BiLSTM-CNN(9)	0.85	0.85	0.85
BiLSTM-CNN(10)	0.86	0.86	0.86

$$recall = Truepoistive/Truepositive + Falsenegative \quad (19)$$

The F-score is defined as the weighted average of precision and recall. The greater the F-score, the better is the performance of the model. Mathematically, F-score is computed using the following Eq. (20):

$$F{-}score = 2 \times (Recall \times Precision)/(Recall + Precision) \quad (20)$$

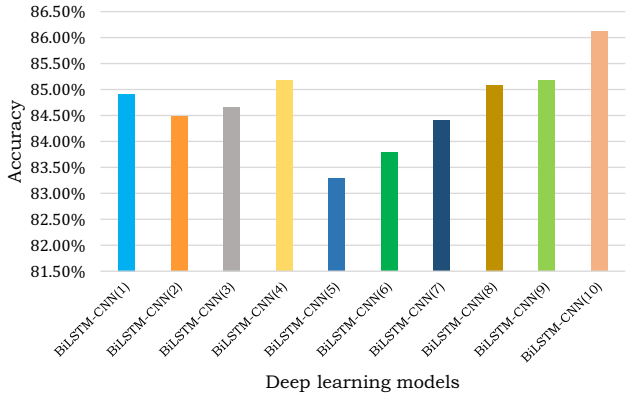

Fig. 8 Comparison of accuracy for all BiLSTM-CNN models

Figure 8 shows comparative results in terms of accuracy, of all the models (BILSTM-CNN). It is evident that the proposed BiLSTM-CNN (BiLSTM-CNN (10)) performed better than the other variants of BiLSTM-CNN. *Why our results are better*? After performing experimentation with different parameters variations of BiLSTM-CNN models, it has been analyzed that through proper training the BiLSTM-CNN (10) is able to achieve an accuracy of 86.12%.

4.1.2 RQ#2: What is the performance of Word Embedding learned using BiLSTM-CNN over the classical feature sets bag-of-word techniques (BOW) like Tf-idf, CountVectorizer?

We conducted an experiment to evaluate the efficiency of word embedding features learned using BiLSTM-CNN w.r.t the classical feature set on the Bag-of-Word(BOW) approach, namely CountVectorizer and Tf-IDF used in the machine learning algorithms. The performance evaluation results reported in Table 12 are summarized as follows.

Part-A In this experiment, we evaluated the performance of different baseline(classical) feature representation schemes, such as CountVectorizer and Tf-IDF. The CountVectorizer uses count-of-word approach, whereas the Tf-IDF transforms the text into a feature vector for the text classification. For experimentation, multiple classifiers are used for both CountVectorizer and TF-IDF techniques, among them, DT achieved the best accuracy (81%) and the RF model achieved the lowest accuracy (65.43%).

Part B In this experiment, we performed a rumor classification task using BiLSTM-CNN model with task-specific word embedding feature set. The basic advantage of using word embedding over the traditional BOW model is that when the vocabulary size is too large, then the BOW model exhibits poor performance, whereas the deep learning based word embedding gives improved results.

Table 12 Performance evaluation of machine learning vs proposed model

	Methods	Acc	Pre	Recall	F-score
Part-A					
Baselines					
Machine learning with traditional features	TF-IDF + KNN (Zubiaga et al. 2018)	80.94	0.81	0.81	0.81
	BOW + DT (Zubiaga et al. 2018)	**81%**	**0.82**	**0.82**	**0.82**
	BOW + NB	68.15	0.79	0.66	0.64
	CountVectorizer + LR	78.18	0.78	0.78	0.77
	CountVectorizer + RF	65.43	0.77	0.65	0.52
Part-B					
Proposed methodology	**BiLSTM-CNN + task-specific word embedding**	**86.12%**	**0.86**	**0.86**	**0.86**

Bold entries show that a particular method has shown promising results in terms of better accuracy, precision, recall, and F-score

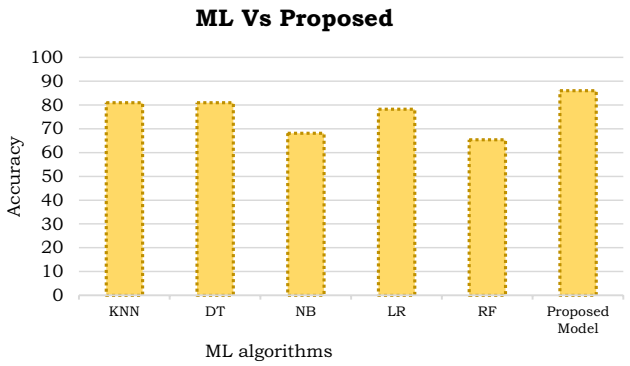

Fig. 9 Performance analysis of proposed system (accuracy-based) with the baseline ML algorithms

Part A vs B The results reported in Table 11 show that the proposed deep learning model(BiLSTM-CNN) based on task-specific word embedding yields more improved results when compared with that of classical machine learning classifiers using traditional feature representation schemes (CountVectorizer, Tf-IDF).

Figure 9 presents the comparison of the proposed model with the baseline ML algorithms to analyze the performance improvement in terms of accuracy. The baseline ML algorithms such as K-NN, DT, LR, NB, and RF are applied to the same dataset using Anaconda based jupyter notebook.[8]

It is observed that the BiLSTM-CNN model performed better than different ML classifiers (Fig. 9) *Why our results are better*? The word embedding based features in the BiLSTM-CNN capture both the syntactic and semantic information about the given token(word). The feature representation is encoded in a dense and low dimensional vector. On the contrary, the classical BOW feature in ML classifier cannot capture the semantics of an input sentence, because it treats each word as a one-hot vector. The feature encoding requires large dimensional space with the lack of capturing complex linguistic information due to its sparse representation.

[8] https://www.anaconda.com/.

4.1.3 RQ#3: What is the performance of the proposed technique for tweet classification into rumor and non-rumor with respect to the state-of-the-art methods?

To evaluate the performance of the proposed BiLSTM-CNN model (word embeddings) for rumor detection, we compare with different deep learning models on the 3 datasets. Results presented in Tables 13, 14, 15 are summarized as follows.

- ***BiLSTM-CNN (Proposed) VS CNN (Baseline):*** In the first experiment, we compare the performance of our proposed model with the work performed by [30] using a single layer CNN model for rumor classification. Results presented in Table 13 show that the CNN model with single convolution layer yielded poor results (Acc: 79.74%, Pre: 80%, Rec: 80% and F-score: 78%) when compared with the proposed BiLSTM-CNN model. The basic reason behind the degraded performance of CNN is that single convolution layer doesn't keep track of maintaining the sequence information in text, which is required for text classification problem. Furthermore, the CNN model needs a large dataset for providing improved classification results.
- ***BiLSTM-CNN (proposed) vs RNN(Baseline):*** In the next experiment, we compared the performance of our model with the single layer RNN proposed by Ma et al. (2016). Our proposed model outperformed the single layer RNN because it can't process extra-long sequences, i.e. the RNN lacks in learning long term dependencies. For retaining the context information, it is required to maintain information for long a time interval, which the single layer RNN can't do, because of it. remembers only the short-term memory sequences. Results presented in

Table 13 Performance evaluation of deep learning vs proposed model for D1

	Methods	Acc (%)	Pre	Recall	F-score
Baselines Deep learning with Word embedding	1-layer CNN + task specific word embedding (Yu et al. 2017a)	79.74	0.80	0.80	0.78
	1-Layer LSTM + task specific word embedding (Ajao et al. 2018)	82.76	0.83	0.83	0.82
	LSTM-CNN + task specific word embedding (Ajao et al. 2018)	**83.53**	**0.83**	**0.84**	**0.83**
	Attention-based BiLSTM-CNN	84.66	0.85	0.85	0.85
	RNN + task specific word embedding (Ma et al. 2016)	80.86	0.81	0.81	0.80
Proposed methodology	**BiLSTM-CNN + task-specific word embedding**	**86.12**	**0.86**	**0.86**	**0.86**

Bold entries show that a particular method has shown promising results in terms of better accuracy, precision, recall, and F-score

Table 14 Performance evaluation of deep learning vs proposed model for D2

	Methods	Acc (%)	Pre	Recall	F-score
Baselines Deep learning with Word embedding	1-layer CNN + task specific word embedding	79.92	0.80	0.80	0.80
	1-Layer LSTM + task specific word embedding	81.69	0.82	0.82	0.82
	1-Layer BiLSTM + task specific word embedding	83.74	0.84	0.84	0.84
	RNN + task specific word embedding	80.05	0.81	0.80	0.80
	Attention-based BiLSTM-CNN	84.29	0.84	0.84	0.84
Proposed methodology	**BiLSTM-CNN + task-specific word embedding**	**86.61**	**0.87**	**0.87**	**0.87**

Bold entries show that a particular method has shown promising results in terms of better accuracy, precision, recall, and F-score

Table 15 Performance evaluation of deep learning vs proposed model for D3

	Methods	Acc (%)	Pre	Recall	F-score
Baselines Deep learning with Word embedding	1-layer CNN + task specific word embedding (Yu et al. 2017a)	80	0.80	0.80	0.80
	1-Layer LSTM + task specific word embedding (Ajao et al. 2018)	82	0.82	0.82	0.82
	1-Layer BiLSTM + task specific word embedding	83	0.83	0.83	0.83
	RNN + task specific word embedding (Ma et al. 2016)	81	0.81	0.81	0.81
	Attention-based BiLSTM-CNN	84	0.84	0.84	0.84
Proposed methodology	**BiLSTM-CNN + task-specific word embedding**	**85**	**0.85**	**0.85**	**0.85**

Bold entries show that a particular method has shown promising results in terms of better accuracy, precision, recall, and F-score

Table 13 show that the RNN model has exhibited 5.26% less efficient in terms of precision, recall, f-measure, and accuracy, as compared to our proposed BiLSTM-CNN model

- ***BiLSTM-CNN (proposed) vs LSTM (Baseline):*** In this experiment, we evaluated the performance of our proposed model with the work performed by Ruchansky et al. (2017) using the LSTM model for rumor classification. The major limitation of unidirectional LSTM is that it preserves only past information, and can't preserve information of the future/next state for better interpretation of the sentence context. Therefore, a single layer of LSTM yielded a performance degradation of up to 3.36% than the proposed model.
- ***BiLSTM-CNN (proposed) vs LSTM-CNN (Baseline):*** In the next experiment, we evaluated the performance of our proposed model with the work performed by Ajao et al. (2018) using LSTM-CNN model for fake news identification. Experimental results (Table 10) show that the proposed model has shown a performance improvement of 2.95% than the state of the art LSTM-CNN model. Experimental results (Table 13) show that the proposed model has shown a performance improvement of 2.95% than the state of the art LSTM-CNN model (Ajao et al. 2018). *Why our results are better?* The major reason behind the poor performance of the baseline work is that it used initially a unidirectional LSTM, which can only maintain information from the

previous input without acquiring future context information. Due to such information loss, the subsequent CNN layer cannot capture the context information efficiently and resultantly the convolution layer loses the sequence information. Therefore, LSTM-CNN model performed less efficiently than the proposed BiLSTM-CNN model for rumor detection.

- ***BiLSTM-CNN (proposed) vs Attention based BiLSTM-CNN (Baseline):*** In this experiment, we evaluated the performance of our proposed model with the work performed by Zhu et al. (2018) for text classification.

  The basic theme of the attention mechanism is to pay attention to the significant words (Rocktäschel et al. 2015). The attention is an additional MLP mechanism, which is jointly trained with all the other components of the deep learning model (Attention-based BiLSTM-CNN). This mechanism determines which words should be given more attention, than the other words, in a sentence when predicting sentence class. The attention layer aims at creating a context vector for each word (Zhao and Wu 2016).

  Experimental results reported in Table 13 show that the Attention-based BiLSTM-CNN model has exhibited 1.46% less efficiency as compared to our proposed BiLSTM-CNN model. It is observed that the performance has not improved by using the attention layer, because the model has become more complex to be trained (Han et al. 2018).

We evaluate the performance of the proposed model with the two additional datasets (D2, D3) and their results are reported in Tables 14 and 15. Experimental results indicate that the proposed BiLSTM-CNN model perform better the other baselines models in terms of accuracy, precision, recall, and f-measure. These results show that by combining the BiLSTM with CNN, we are able to achieve better accuracy."

The aforementioned experiments show that the proposed BiLSTM-CNN model has performed improved performance (accuracy: 86.12%) as compared to the state-the-art deep learning-based studies (Duong et al. 2017; Ma et al. 2016; Ruchansky et al. 2017; Ajao et al. 2018), conducted on the rumor classification.

4.1.4 Why our results are better?

The major reason for the performance improvement of our model over state-of-the-art studies is that we used BiLSTM, which preserves information both from the past and future contexts, before making its input to the CNN model. The BiLSTM layer produces a new representation of the input it receives from the embedding layer in such a way that it captures information from both the current input as well as the previous inputs, abstaining from the information loss. Therefore, it is concluded that the BiLSTM model is able to retain the contextual information (current and previous) over a long period of time for making predictions efficiently. The BiLSTM model has shown efficient performance for the tweet classification into rumors and non-rumors in conjunction with the CNN layer, which extracts useful n-gram features and such richer representation assists in receiving better accuracy.

Table 16 Performance evaluation using pre-trained word Embeddings

Dataset	Pretrained embeddings (BiLSTM-CNN Model)	Accuracy	Precision	Recall	F-score
D1	Glove	78.10	0.80	0.78	0.79
	FastText	80.69	0.82	0.81	0.81
	Google Word2vec	83.53	0.83	0.84	0.83
D2	Glove	75.14	0.75	0.75	0.75
	Fasttext	76.50	0.77	0.76	0.77
	Google Word2vec	76.37	0.77	0.76	0.77
D3	Glove	77	0.77	0.76	0.76
	Fasttext	80	0.80	0.80	0.80
	Google Word2vec	81	0.82	0.82	0.82

4.2 Addressing few limitations of our proposed model

In this section, we address two limitations of our proposed model, the detail is given as follows:

4.2.1 Using 2 additional datasets

To address the limitation #1(see limitation section), we have applied the proposed model on two additional datasets, and their results are reported in Sect. 4. The implementation code is given in the supplementary material.

4.2.2 Applying sampling technique on dataset D1

One of the limitations of our system (see limitation section) is the use imbalance classes in the dataset (D1), which may result in performance degradation. To address this issue, different sampling techniques, such as under-sampling and oversampling can be applied. We applied oversampling technique, implemented in Python Anaconda-based framework, and achieved an accuracy improvement of 1%, i.e. 87% as compared to result reporter in Table 13. Oversampling can be defined as adding more copies of the minority class and we have used resampling module form Scikit-learn to randomly replicate sample from the minority class. The implementation code is given in supplementary material.

4.2.3 Using pre-trained word embedding

There are two ways of adding an embedding layer in Deep learning model, namely (i) Train your own embedding layer, and (ii) Use a pre-trained embedding. One more limitation (see limitation section) of our proposed system is that we used task-specific word embedding, based on the learning from scratch in the rumor dataset. To address this limitation, we also used three pre-trained word embedding models, namely Glove, FastText and Google Word2vec on the proposed BiLSTM-CNN, and the results are listed in below Table 16. The implementation code is given in supplementary material.

The performance of the proposed model on three Dataset D1, D2, D3 is computed using the pre-trained embeddings. The experimental results (Table 16) depict that Google Word-2vec performs better than the other pre-trained embeddings.

4.3 Significance test

We performed two experiments to investigate whether the proposed *BiLSTM-CNN* model with features, based on word embeddings, is statistically significant than that of Machine Learning classifier (Decision Tree) with classical features based on BOW, and does not occur by chance.

We randomly extracted (260) tweets from the corpus and each tweet was classified by both deep learning model (BiLSTM-CNN) vs Machine learning algorithm (Decision tree). The experimental setup is comprised of the following null and alternate hypothesis.

H_0: The error rate of the two classifiers is the same.

H_A: The error rate of both classifiers is significantly different.

The McNemar's test statistic ("Chi squared") can be computed as follows:

$$\chi^2 = \frac{(|b - c| - 1)^2}{(b + c)} \tag{21}$$

where the notation χ^2 used in Eq. 21 denotes the Chi squared statistic, b and c represent the number of discordant pair, and 1 represents the degree of freedom.

4.3.1 Discussion

In the first experiment, the performance of Decision tree with traditional feature set is reported in Table 12. The decision tree algorithm showed relatively poor performance in all four metrics (Acc, pre, recall, and f1-measure) (Habib et al. 2018). The results depict that the Decision tree with traditional features set i.e. BOW is shown poor performance to detect the rumors from tweets.

Table 17 Significance test to measure the performance difference between baseline (decision tree) and the proposed classifier (BiLSTM-CNN)

Proposed classifier (BiLSTM-CNN) with word embedding	Baseline classifier (DT) with traditional feature set (BOW)		Total
Correctly Classified	185	38	223
Misclassified	20	17	37
	205	55	260

McNemar 's Test: The Chi squared is 4.98 with 1 degree of freedom and the two-tailed P value is 0.026 reject the null hypothesis and accept alternative hypothesis (the two algorithms are statistically significant)

In the second experiment, the BiLSTM-CNN classifier with word embedding-based features has shown significantly better results (Table 13). The model predicted rumors in tweets with an accuracy of 86.12%.

In the significant test, we validated that there is a significant difference between Deep learning model (BiLSTM-CNN) with word embedding features than the Machine Learning classifier (Decision Tree) with classical features. Results in Table 17 show the tweets on which both classifiers disagreed are 58 (Classifier with a different type of features display different behavior to the misclassification). McNemar's test with continuity correction is applied to calculate the p value. The Chi squared value is 4.98 and the two-tailed p-value is 0.026 with one degree of freedom. The null hypothesis is rejected with small p-value < 0.5 and supports the alternative hypothesis: the proposed *BiLSTM-CNN* model with features, based on word embeddings, is statistically significant than that of Machine Learning classifier (Decision Tree) with classical features based on BOW.

The aforementioned statistical analysis shows that the word embedding features significantly improved the performance of the proposed model (BiLSTM-CNN) for tweet classification into rumors and non-rumors.

4.4 Limitations

The proposed approach has the following limitations.

1. In this work, only text-based features are used for rumor classification, whereas the inclusion of further types of the features may produce more robust results (Ma et al. 2016).
2. The work focused only on the English text representation.

4.5 Future directions

1. In addition to text-based features, other types of features, such as images and social context, can be investigated for obtaining more efficient results.
2. In future, more experiments will be performed on the textual content in language text.
3. In the future, we want to investigate other techniques in deep learning and multidimensional CNN for rumor classification (Ahmad et al. 2019; Khan et al. 2019).

5 Conclusion

In this work, we addressed the problem of tweet classification into rumor and non-rumor by using a Deep learning-based BiLSTM-CNN model. The proposed work consists of the following tasks: (i) dataset acquisition, (ii) preprocessing, (iii) feature representation, (iv) feature encoding, (v) feature extraction, and (vi) classification

The proposed BiLSTM-CNN model is a combination of BiLSTM and CNN. The BiLSTM, also called sequential layer, preserves the sequence information in both directions (forward and backward), while the CNN layer captures the features from the rich representation produced by the BiLSTM layer; and finally, the tweet is classified as rumor and non-rumor. We experimented with various machine learning and deep learning classifiers and reported their results on the benchmark Twitter dataset. Experimental results show that that the BiLSTM-CNN performed better than the other models by outperforming all other classifiers and achieving the best results in terms of improved accuracy (86.12%), precision (86%), recall (86%) and f-measure (86%).

Acknowledgements This research work was supported by Zayed University Cluster Research Award # R18038.

Compliance with ethical standards

Conflict of interest The authors declare that there are none of the authors have any competing interests in the manuscript.

References

Acharya A (2017) Comparative study of machine learning algorithms for heart disease prediction

Ahmad S, Asghar MZ, Alotaibi FM, Awan I (2019) Detection and classification of social media-based extremist affiliations using sentiment analysis techniques. Hum Centric Comput Inf Sci 9(1):24

Ahmed F, Abulaish M (2012) An mcl-based approach for spam profile detection in online social networks. In: 2012 IEEE 11th international conference on trust, security and privacy in computing and communications, pp 602–608, IEEE

Ajao O, Bhowmik D, Zargari S (2018) Fake news identification on twitter with hybrid cnn and rnn models. In: Proceedings of the 9th international conference on social media and society, ACM, pp 226–230

Alayba AM, Palade V, England M, Iqbal R (2018) A combined cnn and lstm model for arabic sentiment analysis. In: International cross-domain conference for machine learning and knowledge extraction, pp 179–191, Springer, Cham

Allport GW, Postman L (1947) The psychology of rumor

Alzanin S, Azmi A (2018) Detecting rumors in social media: a survey. Proc Comput Sci 142:294–300. https://doi.org/10.1016/j.procs.2018.10.495

Asghar MZ, Rahman F, Kundi FM, Ahmad S (2019a) Development of stock market trend prediction system using multiple regression. In: Computational and mathematical organization theory, pp 1–31

Asghar MZ, Ullah A, Ahmad S, Khan A (2019b) Opinion spam detection framework using hybrid classification scheme. In: Soft computing, pp 1–24

Ayutthaya TSN, Pasupa K (2018) Thai sentiment analysis via bidirectional lstm-cnn model with embedding vectors and sentic features. In: 2018 International joint symposium on artificial intelligence and natural language processing (iSAI-NLP), pp 1–6, IEEE

Barbosa L, Feng J (2010) Robust sentiment detection on twitter from biased and noisy data. In: Proceedings of the 23rd international conference on computational linguistics: posters, Association for Computational Linguistics, pp 36–44

Castillo C, Mendoza M, Poblete B (2011) Information credibility on twitter. In: Proceedings of the 20th international conference on world wide web, ACM, pp 675–684

Chang C, Zhang Y, Szabo C, Sheng QZ (2016) Extreme user and political rumor detection on twitter. In: International conference on advanced data mining and applications, Springer, Cham, pp 751–763

Chen T, Wu L, Li X, Zhang J, Yin H, Wang Y (2017a) Call attention to rumors: deep attention based recurrent neural networks for early rumor detection. arXiv:1704.05973

Chen T, Xu R, He Y, Wang X (2017b) Improving sentiment analysis via sentence type classification using bilstm-CRF and CNN. Expert Syst Appl 72:221–230

Chen W, Zhang Y, Yeo CK, Lau CT, Lee BS (2018) Unsupervised rumor detection based on users' behaviors using neural networks. Pattern Recogn Lett 105:226–233

Chiu JP, Nichols E (2016) Named entity recognition with bidirectional LSTM-CNNs. Trans Assoc Comput Linguist 4:357–370

Conroy NJ, Rubin VL, Chen Y (2015) Automatic deception detection: methods for finding fake news. Proc Assoc Inf Sci Technol 52(1):1–4

Duong CT, Nguyen QVH, Wang S, Stantic B (2017) Provenance-based rumor detection. In: Australasian database conference, Springer, Cham, pp 125–137

Edara DC, Vanukuri LP, Sistla V, Kolli VKK (2019) Sentiment analysis and text categorization of cancer medical records with LSTM. J Ambient Intell Hum Comput pp 1–17

Gupta A, Kumaraguru P, Castillo C, Meier P (2014) Tweetcred: real-time credibility assessment of content on twitter. In: International conference on social informatics, Springer, Cham, pp 228–243

Habib A, Akbar S, Asghar MZ, Khattak AM, Ali R, Batool U (2018) Rumor detection in business reviews using supervised machine learning. In: 2018 5th International conference on behavioral, economic, and socio-cultural computing (BESC), IEEE, pp 233–237

Hamidian, S., & Diab, M. T. (2015). Rumor detection and classification for twitter data. In: Proceedings of the 5th international conference on social media technologies, communication, and informatics (SOTICS), pp 71–77

Han H, Liu J, Liu G (2018) Attention-based memory network for text sentiment classification. IEEE Access 6:68302–68310

Hosseinimotlagh S, Papalexakis EE (2018) Unsupervised content-based identification of fake news articles with tensor decomposition ensembles

Huang H (2017) A war of (mis) information: the political effects of rumors and rumor rebuttals in an authoritarian country. Br J Polit Sci 47(2):283–311

Jaho E, Tzoannos E, Papadopoulos A, Sarris N (2014) Alethiometer: a framework for assessing trustworthiness and content validity in social media. In: Proceedings of the 23rd international conference on World Wide Web, ACM, pp 749–752

Jin Z, Cao J, Guo H, Zhang Y, Luo J (2017a) Multimodal fusion with recurrent neural networks for rumor detection on microblogs. In: Proceedings of the 2017 ACM on multimedia conference, ACM, pp 795–816

Jin Z, Cao J, Zhang Y, Zhou J, Tian Q (2017b) Novel visual and statistical image features for microblogs news verification. IEEE Trans Multimed 19(3):598–608

Khan A, Feng J, Liu S, Asghar MZ (2019) Optimal skipping rates: training agents with fine-grained control using deep reinforcement learning. J Robot 2019

Kimmey DL (2015) Twitter event detection

Kwon S, Cha M, Jung K, Chen W, Wang Y (2013). Prominent features of rumor propagation in online social media. In: 2013 IEEE 13th international conference on data mining, pp 1103–1108

Liang G, He W, Xu C, Chen L, Zeng J (2015) Rumor identification in microblogging systems based on users' behavior. IEEE Trans Comput Soc Syst 2(3):99–108

Liu B (2018) Text sentiment analysis based on CBOW model and deep learning in big data environment. J Ambient Intell Hum Comput, pp 1–8

Liu AN, Quanzhi L, Rui F, Sameena S (2015) Real-time rumor debunking on twitter. In: Proceedings of CIKM, 2015

Liu J, Yang Y, Lv S, Wang J, Chen H (2019) Attention-based BiGRU-CNN for chinese question classification. J Ambient Intell Hum Comput, pp 1–12

Long Y, Lu Q, Xiang R, Li M, Huang CR (2017) Fake news detection through multi-perspective speaker profiles. In: Proceedings of the 8th international joint conference on natural language processing (volume 2: short papers), pp 252–256

Lopez-Martin M, Carro B, Sanchez-Esguevillas A, Lloret J (2017) Network traffic classifier with convolutional and recurrent neural networks for internet of things. IEEE Access 5:18042–18050

Ma X, Hovy E (2016) End-to-end sequence labeling via bi-directional lstm-cnns-crf. arXiv:1603.01354

Ma J, Gao W, Wei Z, Lu Y, Wong KF (2015) Detect rumors using time series of social context information on microblogging websites. In: Proceedings of the 24th ACM international on conference on information and knowledge management, ACM, pp 1751–1754

Ma J, Gao W, Mitra P, Kwon S, Jansen BJ, Wong KF, Cha M (2016) Detecting rumors from microblogs with recurrent neural networks. In: IJCAI, pp 3818–3824

Morris MR, Counts S, Roseway A, Hoff A, Schwarz J (2012) Tweeting is believing? Understanding microblog credibility perceptions. In: Proceedings of the ACM 2012 conference on computer supported cooperative work, ACM, pp 441–450

Pham TT (2018) A study on deep learning for fake news detection

Qazvinian V, Rosengren E, Radev DR, Mei Q (2011) Rumor has it: identifying misinformation in microblogs. In: Proceedings of the conference on empirical methods in natural language processing, Association for computational linguistics, pp 1589–1599

Rani S, Kumar P (2018) Deep learning based sentiment analysis using convolution neural network. Arab J Sci Eng pp 1–10

Rocktäschel T, Grefenstette E, Hermann KM, Kočiský T, Blunsom P (2015) Reasoning about entailment with neural attention. arXiv :1509.06664

Ruchansky N, Seo S, Liu Y (2017) CSI: a hybrid deep model for fake news detection. In: Proceedings of the 2017 ACM on conference on information and knowledge management, ACM, pp 797–806

Seo E, Mohapatra P, Abdelzaher T (2012) Identifying rumors and their sources in social networks. In: Ground/air multisensor interoperability, integration, and networking for persistent ISR III, vol 8389, International Society for Optics and Photonics, p 83891I

Shao C, Ciampaglia GL, Varol O, Flammini A, Menczer F (2017) The spread of fake news by social bots. arXiv:1707.07592

Shen Q, Wang Z, Sun Y (2017) Sentiment analysis of movie reviews based on cnn-blstm. In: International conference on intelligence science, Springer, Cham, pp 164–171

Tripathy RM, Bagchi A, Mehta S (2010) A study of rumor control strategies on social networks. In: Proceedings of CIKM, ACM, pp 1817–1820

Veyseh APB, Ebrahimi J, Dou D, Lowd D (2017) A temporal attentional model for rumor stance classification. In: Proceedings of the 2017 ACM on conference on information and knowledge management, ACM, pp 2335–2338

Wu K, Yang S, Zhu KQ (2015) False rumors detection on sina weibo by propagation structures. In: 2015 IEEE 31st international conference on data engineering, IEEE, pp 651–662

Yang F, Liu Y, Yu X, Yang M (2012) Automatic detection of rumor on Sina Weibo. In: Proceedings of the ACM SIGKDD workshop on mining data semantics, ACM, Xiaomo, p 13

Yang Z, Wang C, Zhang F, Zhang Y, Zhang H (2015) Emerging rumor identification for social media with hot topic detection. In: Web information system and application conference (WISA), 12th, IEEE, pp 53–58

Yu F, Liu Q, Wu S, Wang L, Tan T (2017a) A convolutional approach for misinformation identification. In: Proceedings of the 26th international joint conference on artificial intelligence. AAAI Press, pp 3901–3907

Yu S, Li M, Liu F (2017b) Rumor identification with maximum entropy in micronet. Complexity

Zeng Y, Yang H, Feng Y, Wang Z, Zhao D (2016) A convolution bilstm neural network model for chinese event extraction. In: Natural language understanding and intelligent applications, Springer, Cham, pp 275–287

Zhang L, Xiang F (2018) Relation classification via bilstm-cnn. In: International conference on data mining and big data, Springer, Cham, pp 373–382

Zhang Y, Yuan H, Wang J, Zhang X (2017) YNU-HPCC at EmoInt-2017: using a cnn-lstm model for sentiment intensity prediction. In: Proceedings of the 8th workshop on computational approaches to subjectivity, sentiment and social media analysis, pp 200–204

Zhang Z, Zhang Y, Zhou T (2018) Medical knowledge attention enhanced neural model for named entity recognition in Chinese EMR. In: Chinese computational linguistics and natural language processing based on naturally annotated big data, Springer, Cham, pp 376–385

Zhao Z, Wu Y (2016) Attention-based convolutional neural networks for sentence classification. In: INTERSPEECH, pp 705–709

Zhou X, Zafarani R (2018) Fake news: a survey of research, detection methods, and opportunities. arXiv:1812.00315

Zhou P, Qi Z, Zheng S, Xu, J, Bao H, Xu B (2016) Text classification improved by integrating bidirectional LSTM with two-dimensional max pooling. arXiv:1611.06639

Zhu Y, Gao X, Zhang W, Liu S, Zhang Y (2018) A bi-directional lstm-cnn model with attention for aspect-level text classification. Fut Intern 10(12):116

Zubiaga A, Aker A, Bontcheva K, Liakata M, Procter R (2018) Detection and resolution of rumors in social media: a survey. ACM Comput Surv (CSUR) 51(2):32

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.