International Journal of Advanced Science and Technology
Vol.128 (2019), pp.45-58
http://dx.doi.org/10.33832/ijast.2019.128.05

DETECTION OF CONTEXT-VARYING RUMORS ON TWITTER THROUGH DEEP LEARNING

Mohammad Ahsan[1*] and Madhu Kumari[2]

[1,2]Computer Science and Engineering Department, National Institute of Technology, Hamirpur, India
[1]ahsan@nith.ac.in, [2]madhu@nith.ac.in

***Abstract* — The rapid exchange of information and large user base of online social networks such as Twitter or Facebook make them an ideal platform to gather the latest information. Peoples belonging to different parts of the world can easily share thoughts or updates on real-time events just by having an internet connected device. This low cost of information exchange and inadequacy of techniques which can check the veracity of shared information, gives birth to deliberate and the accidental spread of rumors i.e. pieces of information having uncertain truth at the time of posting. There exist techniques which detect rumors on online social networks by extracting patterns from pre-identified rumors, but these techniques are not sufficient to detect fast paced rumors (i.e. breaking news based rumors). Existing techniques require periodic update of rumor detecting patterns for identifying newly emerging rumors. In this paper, a deep learning model is proposed which required no periodic update of rumor related patterns to detect the rumorous information. The results clearly reveal how our approach outperformed state of the art methods of rumor detection.**

***Keywords* — Deep Learning, k-Neighbors, CNN, Online Social Networks, Rumor Detection, Twitter**

1. INTRODUCTION

There is a huge contribution of social media applications like Twitter and Facebook in shaping the current communication structure among people. A few decades back, information was shared through town criers and drums, but these utilities were not sufficient to convey messages beyond few miles. The advent of print media (i.e. books and newspapers) has extended information reach by delivering news to people staying at a distance of hundreds of miles. Next, there came electronic media (i.e. radio and televisions), which can broadcast a message to the whole nation at once. The limitation of electronic media is - restricting ordinary people from sharing their experiences and thoughts. In order to broadcast some information through news channels, one has to be a professional journalist. This boundary between journalists and ordinary citizens restricts normal persons from sharing local information, which required during extreme events like earthquakes and terrorist attacks. Social media remove this boundary and let everyone behaves as a reporter and news consumers at the same time. It broadens the diversity of information which people can receive from others.

Twitter is one of the most popular social media application. It is not only used to keep in touch with friends and family members, but also to get information regarding worldwide events. Twitter has 360 million monthly active users and all can share or consume the information depending on their needs. The cost of sharing information on

Received: April 20, 2019
Reviewed: May 4, 2019
Accepted: June 24, 2019

this platform is very low – it requires a device with internet connectivity. Due to this low cost of information sharing and huge user-base, information posted by a single user can reach millions of people within a few seconds [1]. These properties make this application an ideal platform of getting useful information as well as spreading fabricated contents to alter people's perspective about a political party or products and services of the companies.

The fast and broad delivery of information through Twitter incurs a cost in the form of credibility and accountability. In verbal communication (i.e. town criers), the speaker is accountable for his/her words, but on social media there is a lack of accountability. The information broadcasted by news channels is pre-checked for facts and veracity, but in case of social media, there is a post-checking of these parameters. This flexibility of Twitter allows its users to quickly spread an information without any pre-check regarding veracity. A timely provision of such local information is very much helpful in handling the post-event situation [2]. In many situations, Twitter is cited as a source of information by news channels due to its real-time reporting, for example, Haiti earthquake 2010 [3]. Despite these advantages, Twitter is also used for spreading rumors, lies, and misinformation. The rumor is a currently circulating story that seems important to its recipients and has an uncertain or doubtful truth. In 2010, after Haiti earthquake, rumors got spread that "UPS will ship any package under 50 lbs to Haiti for free". Apart from spreading these false information, rumors are also capable of doing harm in the form of reputation and lives [4]. So, there is a need to devise some methods for detecting and controlling the spread of rumors on Twitter.

The spread of rumors causes substantial harm to the society in the form of financial loss, reputation loss and even danger of lives during extreme cases i.e. natural disasters or terrorist attacks. This research can be helpful in limiting the spread of any information that may end up being false and reduce the risk of panic situations to communities or individuals [5]. The threat posed by rumors has prompted researchers to comprehend rumors' diffusion process and designing rumor controlling methods. The primary technique used by the government bodies is communicating anti-rumor messages to the whole population [6]. But it fails to debunk rumors in the situation where people start thinking that government has a personal stake (i.e. maintaining lawfulness) behind spreading anti-rumor messages. In 2003, rumors were spread in Nigeria that Polio vaccine is killing fertility of children and it took two years to control these rumors. Online social networks composed of many nations and to timely combat rumors on these platforms there is a need of more robust approach. In this research work, a deep learning based rumor detection model (Convolutional Neural Network) has been proposed to detect rumors on Twitter with a better accuracy.

The remaining paper is organized as follows. In Section 2, we discuss the literature to provide the insights of existing methods and approaches. The proposed scheme is described in Section 3. Results are discussed in Section 4. Finally, a brief conclusion is contained in Section 5.

2. RELATED WORK

In literature, there exists very little research on automatic detection of rumors [7]. Initially, rumor detection was limited to identify posts relevant with pre-identified rumors [8, 9, 10]. The identification of relevant posts needs human experts to manually identify early rumors so that the relevancy of new unseen posts can be checked. Manual identification of rumors needs experts to check multiple sources of information which take considerable time [11]. So, this method works well in a situation where new rumors keep emerging after manual identification of early rumors (i.e. long-standing rumors). But, it is not sufficient to detect new rumors during fast-paced rumors (i.e. breaking news based rumors) because by the time experts manually identify early rumors, new rumors of different context start emerging and previously identified rumors remain of no use. To overcome the challenge of detecting context-varying rumors there is a need of some generalizable pattern. The detection of rumors by utilizing generalized patterns is discussed in Section 2.1.

2.1. REGULAR EXPRESSIONS

There are several needs that force people to share information with others like verification or correction of a controversial information. These needs not only helpful in the quick spread of information on Twitter, but also provide clues for rumor detection. According to Allport and Postman [12], rumor spreading is a multiplicative function of informational ambiguity and its importance.

$$(Rumor \cong Ambiguity \times Importance) \quad (1)$$

It means a rumor post is always shrouded with ambiguity and seems important to its recipients. There is no incentive in sharing an unimportant story or a fact (unambiguous information) which needs no subjective interpretation. So, to understand the clear meaning of rumors people start commenting on these posts. Zhao et al. [13] have used these findings and manually curated a list of 5 regular expressions (as listed in Table I) to identify tweets of verification and correction patterns. To the extent of our knowledge, this is the only work in the literature which detect new rumors.

Table I. Regular Expressions to Identify Verification and Correction Patterns

Pattern	**Regular Expressions**
Verification	*is (that| this| it) true*
	*wh[a] * t [? !] [? 1]**
	(real? | really? | unconfirmed)
Correction	*(rumor | debunk)*
	(that | this | it) is not true

But all rumors do not contain these patterns and the ones having no such characteristics can easily bypass this method of rumor detection. These regular expressions require manual updates periodically to detect rumors of newly emerging patterns. This approach needs to be extended for automatic extraction of the latent patterns from rumor/non-rumor information.

2.2. TWEETS TIMELINE

Zubiaga et al. [14, 15] have developed a tool to display the timeline of event specific tweets. With this, they leverage the context of the tweets to determine whether a post constitutes a rumor or not. In the rumor detection process, a single tweet is not always enough to identify whether it is a rumor or not. They enlist the help of experienced annotators like journalists to detect rumors on Twitter. They examined tweets which have been retweeted a decent number of times because a rumor is used to be ambiguous and important to its recipients. People share rumors with their connections in order to get a clear meaning out of it. But not all rumors trigger a large number of enquiring tweets, and can easily bypass this method of rumor detection. So, there is a need of more robust method which can automatically learn rumor related pattern from tweets' text and detect rumors with a considerable accuracy. For this, a Convolutional Neural Network (CNN) model has been trained in this paper, where no manual extraction of rumor related patterns is required. This model automatically learns patterns from the text of rumor/non-rumor tweets.

On social media, annotation of users' posts as rumor and non-rumor is possible by employing certain filters like discard posts which have not been retweeted by the users a given number of times. But this annotation encounters certain challenges in the form of annotation quality and time. Social media data are usually unstructured, noisy and multi-modal (text, audio, video, pictures). The annotation of social media data needs deep knowledge of the domain under investigation and it is not always possible to have domain experts in a timely manner. It leads to low quality annotation of the social media posts.

In this research, a standard dataset of rumor/non-rumor tweets is used by the CNN model to classify social media posts as rumor and non-rumor. This work presents how deep learning based models are better in detecting fast-paced rumors (breaking news based rumors) than other existing machine learning methods of rumor detection like k-Neighbors [16] and SVM [17].

3. PROPOSED APPROACH

In order to detect rumors on Twitter, a CNN model is trained on PHEME dataset of rumors and non-rumors. First, tweets are pre-processed to remove irrelevant data attached in the metadata of tweets. The preprocessing steps are discussed in algorithm 1 (section 3.2.1). Then, features relevant to our study are extracted in section 3.2 (Table 3). Finally, a CNN model is trained and tested to capture the real world scenario of fast paced rumors on Twitter.

3.1. DATASET

The data used in this work contain rumor and non-rumor tweets belonging to various breaking news stories [18] i.e. *Charlie Hebdo shooting* (on 7th January 2015, Charlie Hebdo branch was attacked by two gunmen from Al-Qaeda. The attack was a reaction to the publication of Prophet Muhammad S.A.W. cartoon by this magazine), *Ebola-Essien* (in October 2014, rumors were spread that Michael Essien has contracted Ebola and later the footballer himself denied this information), *Ferguson unrest* (due to a killing of African-American teenager, Michael Brown, by a white police officer from Ferguson police department, United States witnessed an unrest from 9th August 2014 to 24th November 2014), *crash of Germanwings plane* (on 24th March 2015, a germanwings plane took off with 150 passengers from Barcelona but it was crashed mid-way before reaching the scheduled destination - Dusseldorf, Germany. This crash was a result of the co-pilot's mental illness, but social media was flooded with hijacking information), *recovery of artwork from Gurlitt* (on 22nd September 2010, Gurlitt was going from Zurich to Munich and he was investigated by the custom officers about the reason of crossing Swiss border. After interrogating him, officers came to know that he has a collection of Nazi-era artworks. The information presented by social media users stated that a Swiss museum is going to accept artworks from a Nazi-era art collector.), *Ottawa shooting* (on 22nd October 2014, a Canadian soldier was killed in a terrorist attack on the Parliament Hill in Ottawa, Canada), *Prince-show in Toronto* (On 5th November 2014, a tweet was posted at 6AM that Price would be performing a secret show in Toronto. This tweet was deleted by 8AM but a spread of this information on Twitter brought huge crowd to Toronto city.), *Putin missing* (from 5th March 2015 to 15th March 2015, Putin was out of public view and that disappearance has sparked rumors about his death, illness, and becoming father.), and Sydney siege (Haron Monis held hostage 10 customers and 8 employees of a chocolate café in Sydney, Australia from 15th December to 16th December 2014).

The annotation process of these tweets is discussed in [11] and statistics about the number of rumor and non-rumor tweets are presented in Table 2.

Table II. Dataset Statistics

S. No.	Events	Number of rumor tweets	Number of non-rumor tweets
1.	Charliehebdo	458	1621
2.	Ebola-Essien	14	0
3.	Ferguson	284	859
4.	Germanwings-crash	238	231
5.	Gurlitt	61	77
6.	Ottawashooting	470	420
7.	Prince-Toronto	229	4
8.	Putin missing	126	112
9.	Sydneysiege	522	699

During these events, rumors were sparked about more supposed attacks, illness and death of reputed personalities, and killing of one social group by other (African-American and white, Muslims and non-Muslims). In order to protect the society from getting panic during these breaking news, a rumor detection method is proposed in this research.

3.2. RUMOR DETECTION

Due to the serious consequences of rumors on people's reputation and their lives, rumor detection becomes very important. Extreme events like natural disasters and terrorist attacks create chaos among citizens and individuals try to comprehend the situation as soon as possible. Natural disaster and terrorism provide the optimal condition for rumor spread by creating an anxious environment and generating ambiguous information [12, 19, 20]. People initially turn to the news channels to get some credible information, but fact-checking protocols of these channels slow down their information broadcasting process. This delay in information release by mainstream media diverts people to social media, which allow its users to share information without pre-checking of facts. Social media applications like Twitter are very fast in delivering information, but people have to invest their own knowledge or to refer other sources of information for knowing the veracity of received information. There is no rumor or non-rumor label on the spreading information at Twitter and even a rumorous information seems credible. So, to protect our society from information that may end up being false, there is a need of timely detection of such information.

In this research, a CNN model is trained to classify tweets as rumors and non-rumors. It requires no manual extraction of regular expressions and capable of identifying new rumors even when their contexts are different from events of training posts. The features listed in Table 3 are utilized in this task.

Deep learning models do not process raw text and only work with numeric tensors like other neural networks. There are two major ways of representing the textual data as numeric tensors: one-hot encoding and word embedding. One-hot encoding associates each word with a sparse (most of the entries as 0s) and high-dimensional (same as the number of vocabulary words) vector whereas word embedding is dense and relatively low dimensional. In this work, word embedding is used for tweets' vectorization as it can pack more information into smaller vectors. Generally, word embedding is of two types: pretrained embedding and task specific embedding. Pretrained embedding is precomputed on different tasks of machine learning and helps in solving the problems where no sufficient training data is available to learn the features. Google and Stanford have released some pretrained word embeddings, for example, **Word2vec** (developed by Google in 2013) [21] and **GloVe** (developed by Stanford researchers in 2014) [22]. Task specific embedding is learned jointly with the main task a user is solving and usually outperforms pretrained embedding if sufficient training data is available. Here, task specific embedding is used for representing text with numeric tensors as it gave us a higher accuracy of rumor detection than other alternatives (word2vec or glove).

Table III. Features for Detecting Rumors on Twitter

S.No.	Features	Descriptions
1.	text	The actual text of tweets posted by the users
2.	n_urls	The number of URLs included in a tweet's text
3.	n_hashtags	The number of hashtags attached to a tweet
4.	n_usermentions	The number of Twitter users mentioned in a tweet's text
5.	n_emoticons	The number of emoticons included in a tweet
6.	ul_ratio	The ratio of uppercase characters to lowercase characters in a tweet
7.	verified	It indicates whether a user has verified account or not. The accounts of public interests including politics, fashion, government, religion, sports, media, journalism and other key areas are verified by Twitter authorities.
8.	n_statuses	Total number of tweets issued by a user in the account's lifetime.
9.	account_age	Lifetime of users' account, in months
10.	tweet_frequency	Total number of tweets monthly issued by a user
11.	n_followers	Total number of Twitter accounts that currently following a user
12.	n_followees	Total number of accounts a user is currently following on Twitter
13.	n_favorites	Total number of tweets liked by a user in his/her account's lifetime

3.2.1. TWEETS VECTORIZATION: It is a process of converting tweets into numeric tensors. Twitter users are bound to post their messages within 280 characters and get a functionality of attaching their tweets with other users and topics. So, users choose to add abbreviations, creative spellings, @ (user mentions), urls, and # (hashtags) in their tweets. Due to these quirks, first, tweets are preprocessed by following the steps described in Algorithm 1 and then vectorization is done through task specific embedding.

Algorithm 1: Tweet pre-processing
Input: Original text of tweets
Step 1: Replace https://..... or www…. with <url>
Step 2: Replace user mention (@some_user) with <user>
Step 3: Remove multiple occurrences of characters except A-Z, a-z, and 0-9.
Step 4: Replace emoticons with their names
Step 5: Transform words to their normal form
Step 6: Remove stopwords and digits
Output: Preprocessed text of tweets

Tweets are informal in nature and comes with abbreviations, shortenings, creative spellings, urls, emoticons, etc. To add meaning of the included emoticons, into tweets, these are converted from symbolic form to their names. URLs and user mentions are event specific, so these are replaced them with simple tags as <url> and <user> to get a generic representation. Stopwords and digits do not contribute much in adding meaning to a textual information, so, these are removed from the tweets.

After preprocessing, tweets are tokenized into words and each word is assigned an integer index. These indices are used to learn word embedding with the help of embedding layer provided by Keras [23].

3.2.2. MODEL ARCHITECTURE: A CNN model is designed by adding different layers in a pipelined fashion where the output of one layer is passed as input to the next layer. These layers are data processing units where meaningful representations are extracted from the input data and a more useful data is released as output. The proposed model consists of an embedding layer, convolution layer, flatten layer, input layer, and dense layers (with *relu* and *sigmoid* activation functions) as shown in Fig. 1. Embedding layer process tweets' text and generate 3-dimensional numeric tensors as output. Flatten layer transforms the output of embedding layer to 2-dimensional vectors as a dense layer can only accept the input in 2D form. Dense layer apply $relu$ activation function $(relu(x) = max(0, x))$ to learn the relations of input features and target class. The other choices of activation function like $sigmoid$ and $tanh$ face vanishing gradient problem when used in hidden layers [24]. The features listed in Table 3 except text are received by the input layer to pass them further to the next dense layer. The output of these dense layers which have $relu$ as their activation functions are combined together and passed as input to the last dense layer. Last layer, apply a sigmoid function on the received input and produce a probability score between 0 and 1. This score indicates how likely a tweet is to belong a class 1 (‘rumor’ in this work).

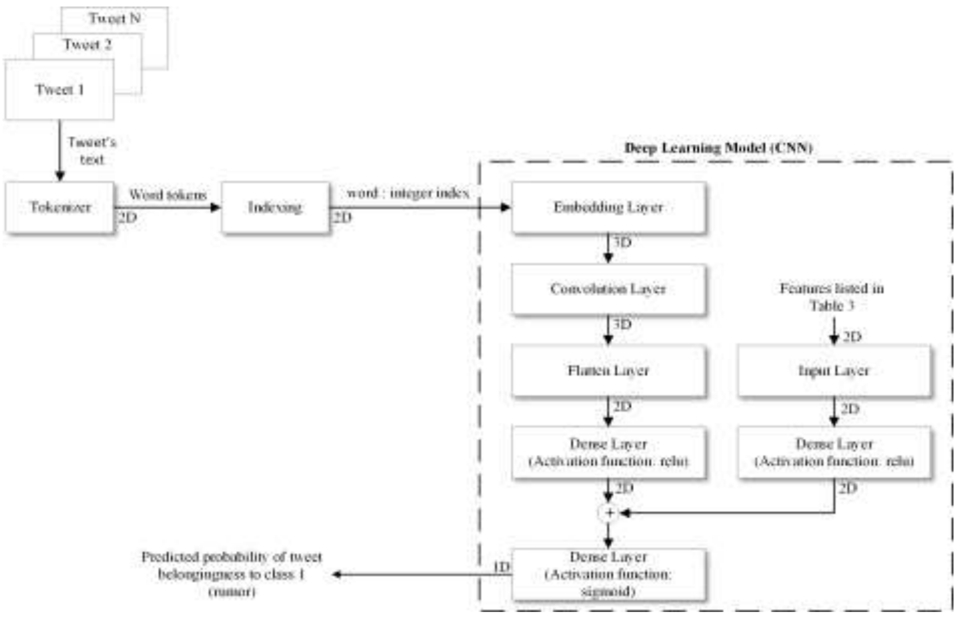
Tweet N
Tweet 2
Tweet 1
Tweet's text
Tokenizer
Word tokens
2D
Indexing
word : integer index
2D
Deep Learning Model (CNN)
Embedding Layer
3D
Convolution Layer
3D
Flatten Layer
2D
Dense Layer (Activation function: relu)
2D
Features listed in Table 3
2D
Input Layer
2D
Dense Layer (Activation function: relu)
2D
+
Dense Layer (Activation function: sigmoid)
1D
Predicted probability of tweet belongingness to class 1 (rumor)

Fig. 1 Predicting how likely the considered tweets belong to a rumor class

The choice of activation function for last layer varies with the type of problem i.e. binary classification - *sigmoid*, and multi-classification - *softmax*. In multi-class classification, if the total number of possible classes is n then the softmax function returns an array of n probability scores with a summation of 1. Each score represents the probability with which an input sample belongs to one of n possible classes. Fig. 1 represents the whole process of how a probability score is predicted for collected tweets using the sigmoid activation function.

3.2.3. MODEL TRAINING: After predicting the probability scores of tweets, the network is trained to minimize the dissimilarity between true and predicted probabilities of these tweets. It requires the network to be configured with some loss function and optimizer. Loss (or objective) function computes a score that has to be minimized during training and an optimizer determines how to update the network according to the given loss. The choice of loss function must correlate with the success of tasks otherwise it will not give required results. For example, an AI entity trained on a loosely defined objective function – ‘maximize the average well-being of people alive’ may choose to focus on the well-being of a few people and kill the rest, which was not intended. Therefore, a wise selection of loss function is must for the success of a task. As the problem of rumor detection falls under the category of binary classification, ‘binary crossentropy’ is selected as a loss function and rmsprop (a variant of stochastic gradient descent) as an optimizer. Crossentropy is derived from information theory and widely used as a loss function in machine learning. It measures the dissimilarity between true and predicted probabilities of samples belongingness to their classes.

After applying the designed CNN model on the collected tweets, the predicted probability of tweets' belongingness to class 1(rumor) and class 0(non-rumor) is computed by using equation 2 and equation 3, respectively.

$$p_{y=1} = \hat{y} = sigmoid\,(W.X) = \frac{1}{1+e^{-WX}} \tag{2}$$

$$p_{y=0} = (1-\hat{y}) \tag{3}$$

In a similar manner, the true probability of tweets' belongingness to class 1 and class 0 can be expressed by equation 4 and equation 5.

$$q_{y=1} = y \tag{4}$$

$$q_{y=0} = (1-y) \tag{5}$$

By applying cross-entropy on these true and predicted probabilities, a dissimilarity score (or loss) is calculated by using equation 6.

$$\begin{aligned} H(p,q) &= -\sum_i p_i \log q_i \\ &= -y\log\hat{y} - (1-y)\log(1-\hat{y}) \end{aligned} \tag{6}$$

The final loss function is computed by taking the average of all cross-entropies of a sample. For N tweets in a sample, each indexed from $n = 1\ to\ N$, the final loss function is computed through equation 7.

$$J(w) = \frac{1}{N}\sum_{n=1}^{N} H(p_n, q_n) = -\frac{1}{N}\sum_{n=1}^{N}[y_n \log\widehat{y_n} + (1-y_n)\log(1-\widehat{y_n})] \tag{7}$$

The score computed by equation 6 is minimized during training in order to increase the accuracy of predicting true labels for collected tweets. In the proposed rumor detection model, rmsprop is used as an optimizer which finds a set of parameters W that minimize loss score. Rmsprop is a variant of stochastic gradient descent, which updates network's weights by finding the gradient of the loss function. For each parameter, it updates the weight as:

$$V_t = \gamma\, V_{t-1} + (1-\gamma) \times g_t^2 \tag{8}$$

$$\Delta w_t = -\frac{\eta}{\sqrt{V_t + \varepsilon}} \times g_t \tag{9}$$

$$w_{t+1} = w_t + \Delta w_t \tag{10}$$

where, V_t = exponential average for current update vector,
γ = a constant value as 0.9,
V_{t-1} = exponential average computed till last update vector,
g_t = gradient of final loss function with respect to weight,
Δw_t = change in weight vector,

η = learning rate,
ε = a small constant, to avoid divide-by-zero exception, and
w_t = weight vector of the network.

Initially, the weight vectors of this CNN model are filled with random values and during training get updated to minimize a gap between predicted and target values. After configuration of the model, numeric tensors given by word embedding, are passed to it for further processing.

4. RESULTS

To analyze the performance of the proposed CNN model, its accuracy is measured on three types of data split: i) training set and test set contain data (tweets) from all events (_m), ii) training set and test set contain data from same event (_s), and iii) training set and test set having data from different events (_d). The performance on these choices of training set and test set are represented in Table IV, Table V, and Table 6 respectively.

In the first choice of data split, collected rumor/non-rumor tweets of all events are shuffled before getting divided into a 70/30 ratio of training and test dataset. In means the proposed CNN model is tested on similar tweets as contained in the training dataset.

Table IV. Rumor Detection Accuracy When Training and Test Set is a Mix of all Events (_m)

Features / **Training and test events**	**Accuracy of rumor detection**				
	Features	**Original tweets**	**Original tweets + features**	**Processed tweets**	**Processed tweets + features**
Data from all events	0.640893	0.837571	0.832382	0.841204	0.827193

In the second strategy of data splitting, the data of each event is divided into 70/30 ratio rather than mixing data of all events and splitting further. Here, CNN model is trained and tested on all events individually. The resulted performance is listed in Table V.

Table V. Rumor Detection Accuracy When Training and Test Set Contain Tweets of the Same Event (_s)

Features / **Training and test events**	**Accuracy of rumor detection**				
	Features	**Original tweets**	**Original tweets + features**	**Processed tweets**	**Processed tweets + features**
Charlie Hebdo	0.791332	0.829856	0.861958	0.865169	0.871589
Ebola-Essien	1	1	1	1	1
Ferguson	0.722222	0.821637	0.847953	0.862573	0.850877
Germanwings-crash	0.614286	0.8	0.8	0.742857	0.728571
Gurlitt	0.487805	0.658537	0.658537	0.780488	0.439024
Ottawa-shooting	0.625468	0.850187	0.868914	0.910112	0.917603
Prince-Toronto	0.956522	0.971014	0.971014	0.971014	0.971014
Putin-missing	0.507042	0.661972	0.577465	0.732394	0.676056
Sydney-siege	0.631148	0.789617	0.795082	0.81694	0.819672
Average accuracy	**0.703981**	**0.820313**	**0.820103**	**0.853505**	**0.808267**

In both first and second method of data split, events of test set tweets are used in the training phase. But in actual scenario of online social networks i.e. Twitter, the context of rumors changes very rapidly and sometimes we do not have pre-identified rumor tweets, similar to newly emerging rumors. To model this situation, the data of all events are divided in such a manner that test set contains tweets of one event and training set contain tweets of the remaining events. Here, the events of the test set are not shared by the training set. The accuracy of rumor detection through this third approach is presented in Table VI.

Table VI. Rumor Detection Accuracy When Tweets of Training Set and Test Set Belong to Different Events (_d)

Features / **Test Events**	**Accuracy of rumor detection**				
	Features	**Original tweets**	**Original tweets + features**	**Processed tweets**	**Processed tweets + features**
Charlie Hebdo	0.685426	0.505051	0.519481	0.60558	0.646364
Ebola-Essien	0.114928	0.428571	0.5	0.397143	0.478571
Ferguson	0.748906	0.565179	0.524934	0.622047	0.637922
Germanwings-crash	0.49467	0.460554	0.466951	0.522388	0.486141
Gurlitt	0.594203	0.514058	0.534783	0.42029	0.454783
Ottawa-shooting	0.52809	0.464045	0.494382	0.503371	0.534719
Prince-Toronto	0.138627	0.318884	0.353219	0.450644	0.512146
Putin-missing	0.470588	0.542017	0.521008	0.546218	0.576227
Sydney-siege	0.599509	0.519247	0.521704	0.628256	0.638084
Average accuracy	**0.486105**	**0.479734**	**0.492940**	**0.521771**	**0.551662**

After having three splits of the collected dataset, the accuracy of the proposed model (CNN) is checked with five different inputs i.e. $features$ (listed in Table 3), $otweets$ (tweets' text in original form), $otweets + features$ (combination of otweets and features), $ptweets$ (tweets' text after preprocessing) and $ptweets + features$ (combination of ptweets and features). Fig. 2 represents accuracy scores of our CNN model.

In the first and second method of data split, highest rumor detection accuracy is achieved by using $ptweets$ as input feature whereas in the third method of data split $ptweets + features$ gives highest accuracy. It means, if events are shared between training set and test set, then $ptweets$ are enough to achieve best possible accuracy. Otherwise, a combination of $ptweets$ and $features$ is required to get the best accuracy. The accuracy of the first and the second split (85% and 84%) is very high compared to the performance of third split (55%). But, the third method of data split capture real scenario of rumors spread on online social networks where the context of rumors changes rapidly.

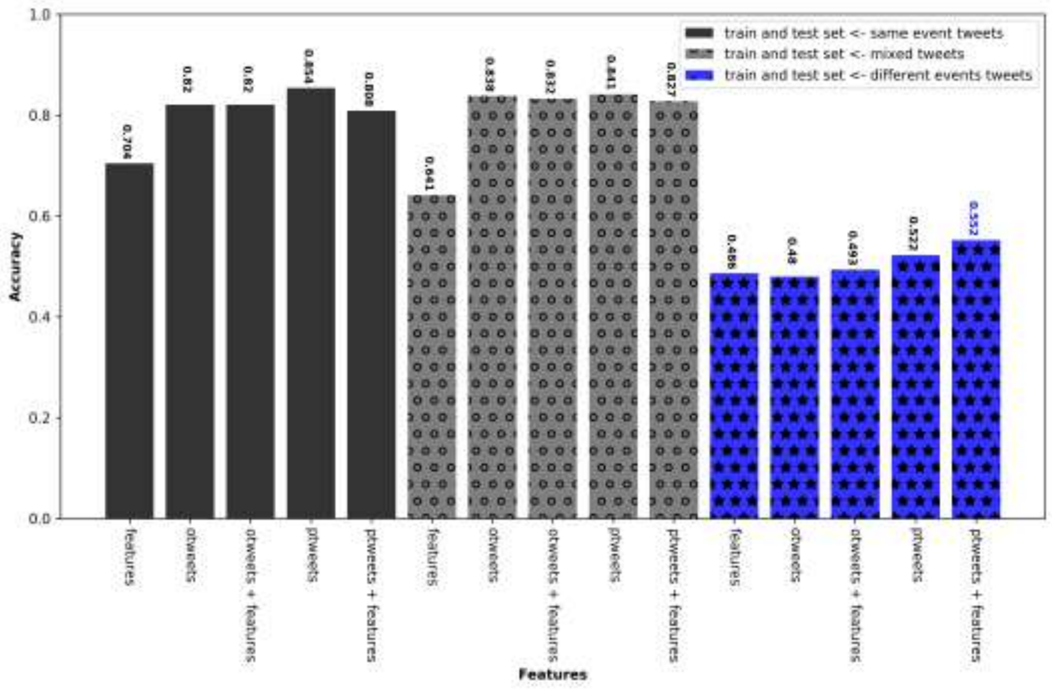

Fig. 2 Rumor detection accuracy of CNN: i) black bars show the accuracy when training data and test data belong to same event, ii) gray bars present the accuracy when training data and test data is a mix of multiple events, and iii) accuracy shown by blue bars is the case when events of training data and test data are mutually exclusive

The performance of the proposed CNN model is also compared with other state of the art methods of rumor detection like k-Neighbors [16], SVM [15, 17], Decision Tree, Random Forest [15], and AdaBoost. The performance of these models is checked on the third method of data split where tweets from same event are not present in both training and test set. Table VII contains the accuracy scores of all shallow learning models where k-Neighbors outperformed all other models with 53.6% rumor detection accuracy.

Table VII. Accuracy of Rumor Detection with Shallow Learning Models

Classifiers / **Test events**	**Accuracy of rumor detection**					
	k-Neighbors	Linear SVM	RBF SVM	Decision Tree	Random Forest	AdaBoost
Charlie Hebdo	0.610870	0.779701	0.684944	0.579605	0.659932	0.665223
Ebola-Essien	0.428571	0.0	0.0	0.357142	0.285714	0.214285
Ferguson	0.660542	0.751531	0.746281	0.602799	0.712160	0.709536
Germanwings-crash	0.539445	0.492537	0.537313	0.573560	0.526652	0.550106
Gurlitt	0.543478	0.557971	0.608695	0.594202	0.536231	0.572463
Ottawa-shooting	0.562921	0.471910	0.552808	0.559550	0.575280	0.559550
Prince-Toronto	0.321888	0.017167	0.047210	0.424892	0.240343	0.188841
Putin-missing	0.554621	0.470588	0.483193	0.478991	0.550420	0.533613
Sydney-siege	0.606060	0.572481	0.633087	0.609336	0.624078	0.631449
Average accuracy	**0.536488**	**0.457098**	**0.477059**	**0.531119**	**0.523423**	**0.513896**

Fig. 3 shows the comparison of shallow learning models (linear SVM, rbf SVM, adaBoost, random forest, decision tree and k-Neighbors) with deep learning based model (Keras). It represents that CNN based model outperformed k-Neighbors model with approx. 2% accuracy in detecting context-varying rumors on Twitter. This 2% margin in correctly classifying the spreading rumors on Twitter is a huge gain as there generated millions of tweets on a daily basis (500M/day [25]).

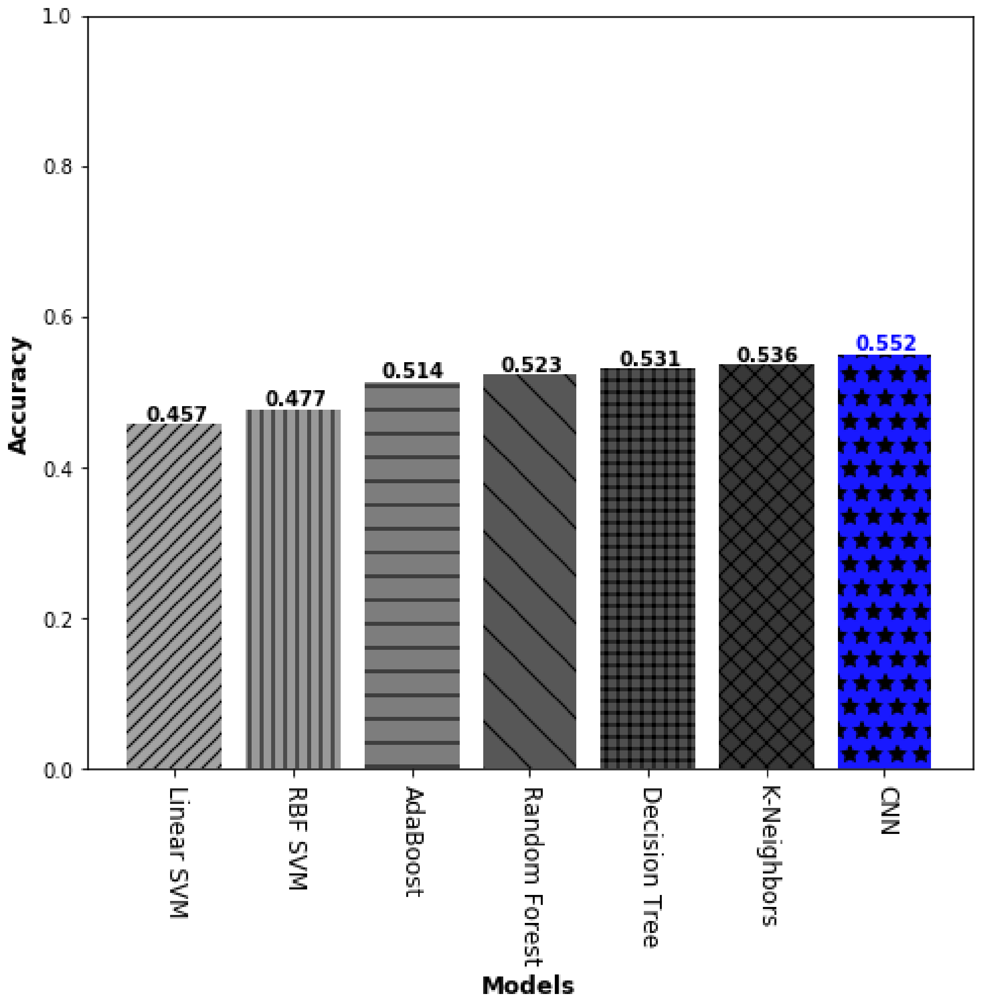

Fig. 3 Comparison of rumor detection accuracy.

5. CONCLUSION

This paper proposes a method of rumor detection on the Twitter social network. Identifying rumors on social media platforms is a very challenging task as rumors and non-rumors seem identical to the users. The state of the art methods for rumor detection either detect posts which are relevant to pre-identified rumors or detect new rumors. The detection of new rumors is done by using a manually curated list of regular expressions. This list needs a periodic update for the effective detection of newly emerging rumors. In this research, a deep learning based model (convolutional neural network) is proposed for rumor detection. This model automatically learns rumor related patterns from users' tweets and no manual extraction or update of regular expressions is required. The effectiveness of the proposed method is shown by extensive simulation on a publicly available dataset of rumors and non-rumors tweets, PHEME. The results clearly indicate that the proposed model outperformed other state of the art methods of rumor detection i.e. support vector machine and random forest. This research protects society from panic situations by more efficiently detecting rumors on Twitter.

ACKNOWLEDGMENT

This work was supported by the Ministry of Electronics and Information Technology (MEITY), Government of India. We would like to thank the editor and the anonymous reviewers for their insightful comments.

REFERENCES

[1] Pinar Ozturk, Huaye Li, and Yasuaki Sakamoto. "Combating Rumor Spread on Social Media: The Effectiveness of Refutation and Warning." 48th Hawaii International Conference on System Sciences (HICSS), Kauai, HI, USA, 5-8 January 2015, pp. 2406-2414, IEEE. DOI: 10.1109/HICSS.2015.288.

[2] Jessica Li and H. Raghav Rao. "Twitter as a Rapid Response News Service: An Exploration in the Context of the 2008 China Earthquake." The Electronic Journal of Information Systems in Developing Countries, Volume 42, Issue 1, pp. 1-22, December 2017. DOI: https://doi.org/10.1002/j.1681-4835.2010.tb00300.x.

[3] Rory O' Connor. "Facebook and Twitter are Reshaping Journalism As We Know It." https://www.alternet.org/2009/01/facebook_and_twitter_are_reshaping_journalism_as_we_know_it. (Accessed Dec 2018).

[4] Rosa Sicilia, Stella Lo Giudice, Yulong Pei, Mykola Pechenizkiy, and Penchenizkiy Soda. "Twitter Rumour Detection in the Health Domain." Expert Systems with Applications, Volume 110, pp. 33-40, November 2018. DOI: https://doi.org/10.1016/j.eswa.2018.05.019.

[5] Helena Webb, Pete Burnap, Rob Procter, Omer Rana, Bernd Carsten Stahl, Matthew Williams, William Housley, Adam Edwards, and Marina Jirotka. "Digital Wildfires: Propagation, Verification, Regulation, and Responsible Innovation." ACM Transactions on Information Systems (TOIS), Volume 34, Issue 3, Article no. 15, May 2016. DOI: http://dx.doi.org/10.1145/2893478.

[6] Rudra M. Tripathy, Amitabha Bagchi, and Sameep Mehta. "A Study of Rumor Control Strategies on Social Networks." Proceedings of the 19th ACM International Conference on Information and Knowledge Management, Toronto, Canada, 26-30 October 2010, pp. 1817-1820, ACM. DOI: 10.1145/1871437.1871737.

[7] Arkaitz Zubiaga, Ahmet Aker, Kalina Bontcheva, Maria Liakata, and Rob Procter. "Detection and Resolution of Rumours in Social Media: A Survey." ACM Computing Surveys (CSUR), Volume 51, Issue 2, Article no. 32, June 2018. DOI: 10.1145/3161603.

[8] Vahed Qazvinian, Emily Rosengren, Dragomir R. Radev, and Qiaozhu Mei. "Rumor has it: Identifying Misinformation in Microblogs." Proceedings of the Conference on Empirical Methods in Natural Language Processing, Edinburgh, United Kingdom, 27-31 July 2011, pp. 1589-1599, Association for Computational Linguistics, Stroudsburg, USA.

[9] Sardar Hamidian, and Mona T. Diab. "Rumor Detection and Classification for Twitter Data." Proceedings of the Fifth International Conference on Social Media Technologies, Communication, and Informatics (SOTICS), Barcelona, Spain, 15-20 November 2015, pp. 71-77.

[10] Sardar Hamidian, and Mona Diab. "Rumor Identification and Belief Investigation on Twitter." Proceedings of the 7th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis, San Diego, California, June 12-17 2016, pp. 3-8. Association for Computational Linguistics.

[11] Arkaitz Zubiaga, Maria Liakata, Rob Procter, Kalina Bontcheva, and Peter Tolmie. "Towards Detecting Rumours in Social Media." Workshops at the Twenty-Ninth AAAI Conference on Artificial Intelligence, Austin, Texas, US, 25-26 January 2015.

[12] Gordon W. Allport and Leo J. Postman. "The Psychology of Rumor." American Psychological Association, New York, NY: Holt, Rinehart, & Winston, October 1947. DOI: https://doi.org/10.1002/1097-4679(194710)3:4<402::AID-JCLP2270030421>3.0.CO;2-T.

[13] Zhe Zhao, Paul Resnick, and Qiaozhu Mei. "Enquiring Minds: Early Detection of Rumors in Social Media from Enquiry Posts." Proceedings of the 24th International Conference on World Wide Web, Florence, Italy, 18-22 May 2015, pp. 1395-1405, International World Wide Web Conferences Steering Committee, Geneva, Switzerland. DOI: 10.1145/2736277.2741637.

[14] Arkaitz Zubiaga, Maria Liakata, and Rob Procter. "Learning Reporting Dynamics during Breaking News for Rumour Detection in Social Media." 24 October 2016. arXiv preprint arXiv:1610.07363.

[15] Arkaitz Zubiaga, Maria Liakata, and Rob Procter. "Exploiting Context for Rumour Detection in Social Media." Springer International Conference on Social Informatics, September 2017, pp. 109-123, Springer, Cham. DOI: https://doi.org/10.1007/978-3-319-67217-5_8.

[16] Raveena Dayani, Nikita Chhabra, Taruna Kadian, and Rishabh Kaushal. "Rumor Detection in Twitter: An Analysis in Retrospect." IEEE International Conference on Advanced Networks and Telecommunications Systems (ANTS), Kolkata, India, 15-18 December 2015, pp. 1-3, IEEE. DOI: 10.1109/ANTS.2015.7413660.

[17] Hardeo K. Thakur, Anand Gupta, Ayushi Bhardwaj, and Devanshi Verma. “Rumor Detection on Twitter Using a Supervised Machine Learning Framework.” International Journal of Information Retrieval Research (IJIRR), Volume 8, Issue 3, July 2018, pp. 1-13. DOI: 10.4018/IJIRR.2018070101.
[18] Elena Kochkina, Maria Liakata, and Arkaitz Zubiaga. “PHEME dataset of Rumour Detection and Veracity Classification.” (Accessed December 2018). https://figshare.com/articles/PHEME_dataset_for_Rumour_Detection_and_Veracity_Classification/6392078.
[19] James M. Dahlhamer and Joanne M. Nigg. “An Empirical Investigation of Rumoring: Anticipating Disaster under Conditions of Uncertainty.” Disaster Research Centre, University of Delaware, April 1994. http://udspace.udel.edu/handle/19716/622.
[20] Onook Oh, Kyounghee H. Kwon, and H. Raghav Rao. “An Exploration of Social Media in Extreme Events: Rumor Theory and Twitter during the Haiti Earthquake 2010.” Thirty First International Conference on Information Systems (ICIS), Saint Louis, Missouri, USA, 15-18 December 2010, Volume 231, pp. 7332-7336.
[21] Word2Vec – Vector representation of words. (Accessed January 2019). https://code.google.com/archive/p/word2vec.
[22] Jeffrey Pennington, Richard Socher, Christopher D. Manning. “GloVe: Global Vectors for Word Representation.” Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), Doha, Qatar, 25-29 October 2014. (Accessed January 2019). https://nlp.stanford.edu/projects/glove.
[23] Keras – The Python Deep Learning Library. (Accessed Feb 2019). https://keras.io.
[24] Rohan Kapur. “The Vanishing Gradient Problem.” March 2016. (Accessed February 2019). https://ayearofai.com/rohan-4-the-vanishing-gradient-problem-ec68f76ffb9b.
[25] Internet Live Stats - Twitter Usage Statistics. (Accessed February 2019). http://www.internetlivestats.com/twitter-statistics.