*Journal of Computer and Knowledge Engineering, Vol. 5, No. 2, 2022. (21-30)*
*DOI: 10.22067/cke.2022.75685.1054*


# Efficient and Deception Resilient Rumor Detection in Twitter*

Research Article

Milad Radnejad[1] Zahra Zojaji[2] Behrouz Tork Ladani[3]

**Abstract:** Social networks have become a central part of our lives these days and have real effects on the world's events. However, social networks greatly boost spreading misinformation and rumors that are becoming more and more dangerous each day. As fighting rumors first requires detecting them, several researchers tried to propose novel approaches for automatic early detection of rumors. However, most of them rely on handcrafted content features which makes them prone to deception and threats the adaptability of the model. Furthermore, a great deal of work have concentrated on event-level rumor detection while it faces early detection with serious challenges. There are also deficiencies in proposed methods in terms of time and resource complexity. This study proposes a deep learning approach to automate the detection of rumors on Twitter. The proposed method relies on automatically extracted features through word and sentence embeddings along with profile and network-based features. It then uses Recurrent Neural Networks (RNN) leveraging Gated Recurrent Units (GRU) for detecting the veracity of a tweet. The proposed method also improves time efficiency. The achieved experimental evaluation results on RumorEval2019 dataset demonstrate that the proposed method outperforms other rival models on the same dataset in terms of both performance and time complexity. By the way, the proposed method is more resilient to deception by avoiding the use of handcrafted content features and leveraging features that are out of the control of the user.

**Keywords:** Deception, Deep Learning, Rumor Detection, Social Network, Twitter

## 1. Introduction

The explosive growth of online social media is an evidence for their crucial role in spreading news in the modern society. Nowadays, a large number of users actively engage in producing or propagating news about different trending topics. The convenience of publishing news in online social networks causes also the spreading of misinformation and rumors.

There have been numerous definitions for rumor in the literature, each offering its interpretation. However, the definition provided in [1] seems to be more popular which defines rumor as "a story or statement in general circulation without confirmation or certainty." Another essential research on rumor has been undertaken by [2], which defines three characteristics for rumors: 1) Rumors have a distinct mode of transmission, 2) Rumors always provide information about some particular person, happening, or condition, and 3) Rumors satisfy audiences. The second and the third characteristics refer to the fact that people feel unsafe in the absence of information, and rumors satisfy them by providing information. Spreading rumors imposes potentially harmful effects on public perception and behavior. One can point to the alleged Russian interference in the 2016 US Presidential Election with the spread of rumors and misinformation through social media [3–5]. Online social networks facilitate rapid propagation of fake news and rumors which thereby greatly amplify the impact of harmful effects.

The most effective and operational approach in rumor detection and debunking todays is manual detection, which is done by authoritative centers and websites like Snopes (www.snopes.com) and Politifact (www.politifact.com). However, although this approach seems to be very accurate, it is slow and ineffective with the nature of fast-spreading rumors in social networks. Another approach used these days is automatic detection using artificial intelligence, which leverages machine learning techniques to detect social network rumors. Although the performances of the proposed systems are lower than manual detection, the upside is the constant innovations that are making this approach a likely candidate to replace manual detection. Automatic rumor detection facilitates detecting and preventing rumors in early stages of spreading prior to affecting the public opinion.

Although several researches have been conducted for detecting rumors, the previous methods mostly rely on handcrafted content features. Along with dynamic changing of social network conversations, the content of rumors and the signs of fake or verified news also change. Therefore, feature extraction process should be also dynamic in order to reflex the specifications of the rumors, which is not achieved in the case of developing detection model based on the handcrafted features. Furthermore, handcrafted features make the rumor detection system more susceptible to deception. Employing these features provides more chance to design fake news with appearance similar to verified news. In addition, handcrafted features could bias the prediction model without any explicit improvement in the performance. Again, most of the prior works operate at the event-level, meaning that it can only detect whether a general topic is a rumor or not and cannot decide about the veracity of a single post. Moreover, event-level rumor detection requires an extensive set of messages in each topic which is not available in the first stages of rumor propagation. Hence event-level models are hardly applicable for early detection. Moreover, the scalability of previously presented rumor detection systems is low due to extensive computational complexity. In this work, due to mentioned shortcomings of the previous works, we propose a deep learning approach for detecting rumors on Twitter. The proposed method operates at the

* Manuscript Received: 06 March 2022; Revised, 22 September 2022. Accepted, 26 September 2022.
[1]. MSc. Of Information Technology, Faculty of Computer Engineering, University of Isfahan, Isfahan, Iran,
[2]. Corresponding author, Assistant Professor, Faculty of Computer Engineering, University of Isfahan, Isfahan, Iran.
**Email**: z.zojaji@eng.ui.ac.ir.
[3]. Professor, Faculty of Computer Engineering, University of Isfahan, Isfahan, Iran.

tweet level i.e., detecting rumor using a tweet and its responses. While handcrafted features can improve a deep learning system in doing its task, we do not directly use handcrafted content features and let the system extract them by itself through feeding it the raw text of tweets. Instead, word and sentence level embedding for content feature extraction are utilized which makes the model more resilient to user deception, more scalable to social network dynamics, and less susceptible to model biasing. Our insight also is that analyzing social and profile features is a rich information source for developing deception resilient models. This is because these features are independent from the content of the claims and are out of the direct control of the user that was neglected in most of researches. Therefore, we use social and profile features in our model. Lastly, we emphasize on the system's performance, taking the time efficiency and scalability of the system into account.

The main contributions of this work can be summarized as:

1. Avoiding the use of handcrafted content features for better dealing with the dynamic nature of social networks and providing more resiliency against the deception;
2. Using profile and network features that provide the system with valuable information about the users and propagation state;
3. Proposing a new deep learning model with RNN architecture leveraging GRU cells for automatic rumor tweet detection;
4. Detection of rumors at the tweet level in order to facilitate early detection;
5. Emphasis on performance, especially in the training phase, that make the system more scalable in comparison to the similar works.

The proposed method is evaluated and compared using the RumorEval 2019 dataset. The overall performance of the system is first compared to the state of the art methods in terms of Macro F-score. The achieved results show that the proposed method outperforms nearly all similar methods. The experimental results also show the superiority of the proposed method in terms of time efficiency comparing the baseline. Furthermore, some experiments are conducted in order to prove the resiliency of the proposed method against the intended content alteration with the aim of deception. We believe that the proposed rumor detection model has enough capabilities to be applied efficiently in early, tweet-level rumor detection task with remarkable tolerance to deception.

Section 2 of the paper will briefly introduce the concepts that were used in our research. Section 3 will discuss the researches that are similar and related to our work. The problem statement will be described in Section 4. Section 5 is devoted to the proposed method description and its details. The evaluation process and the experimental results and comparisons are presented in Section 6, and finally Section 7 will conclude the paper.

## 2. Background

In this section some preliminary concepts about RNNs and text embedding methods are provided.

### *2.1. Recurrent neural network*

As described in [6], recurrent neural networks are a family of neural networks for processing sequential data. These networks arise from the idea of sharing parameters across different parts of a sequence, making them very efficient and effective in processing sequential data as well as in extraction and learning of sequential features. One of the most exciting features of RNNs is that they can process data of different length as an ability not seen in other types of neural networks which require fixed-size inputs. Another feature of these networks is the concept of memory, which arises from the fact that by sharing parameters and processing the data in sequence, each input will contribute to the model's output in a later stage, which acts as a memory. As also described in [7], one significant shortcoming of conventional RNN cells is that by applying Backpropagation through time, they cannot learn or extract dependencies in a long sequence due to the problem of vanishing or exploding gradients; This can be described as a sort of memory loss, which means conventional RNNs have a very short term memory.

### *2.2. Bidirectional RNNs*

RNNs usually process data in a feed-forward approach, meaning at each timestep, the output is calculated using the information from the past, which is the hidden state and the current input [6]. However, in some cases giving the network information about the whole sequence (past and future timesteps) will help solving the problem. Bidirectional RNNs are the combination of two RNNs, one moving forward through time from the start of the sequence and the other moving backward through the time from the end of the sequence. In this way, the output at each timestep is calculated using the information from the past and the future, but more dependent on the data nearest to the current timestep.

### *2.3. Long Short-Term Memory (LSTM)*

To combat the memory loss in conventional RNN cells, LSTM was proposed, which defines a pathway for long dependencies, which acts as long-term memory [6]. This pathway can be seen as a cell inside the LSTM cell with its parameters which can add data to the cell and remove unnecessary data when needed. By giving the network, the option of adding data to and removing it from this pathway, the network can memorize essential data in the sequence and forget unnecessary information, which has made LSTM cells a very successful architecture for solving problems, where the input is a sequence.

### *2.4. Gated recurrent unit*

Although LSTMs are considered the go-to architecture when dealing with sequential data, they have a crucial shortcoming that arises from its too many parameters. These parameters burden the model, which has to do the standard calculations and with learning the parameters of the LSTM. Moreover, due to the high number of parameters, LSTMs are more susceptive to overfitting, which is a prevalent problem in neural networks and deep learning tasks.

To combat the mentioned shortcomings, the GRU cell was proposed in [8], which is very similar to the LSTM cell but differs in that it combines some parts of the LSTM cell into a unified part and causes a reduction in the number of parameters compared to the LSTM cell. This modification has two benefits: 1) it gives the model less space for overfitting compared to LSTM, and 2) it puts less burden on

the model in terms of calculations that in turn makes it run faster.

### *2.5. Text embedding*

In the field of natural language processing, which deals with human language, there is no straightforward way of using words in a neural network. One possible solution might be using one-hot encoding on a lengthy dictionary of words, but this approach has two problems: 1) it wastes memory and processing resources, which can be used elsewhere; (2) by giving the one-hot code based on the alphabetic order, this approach might give close codes to words with different meaning and remote codes to words with similar meaning (the distances are in terms of points in a hyperspace), which might give the model the wrong impression about the similarity of the words.

Word embeddings were proposed to overcome the enumerated challenges by using dense vectors for word representation, reducing memory, and processing the needed resources. The proposed embeddings also put similar words in respect to their concepts and meaning in close vectors in terms of distance, giving the model the ability to understand those words' meanings. Google's word2vec [9] and Stanford's Glove [10] are examples of word embeddings. Although word embeddings are very useful, since the words in a language get their meaning in a sentence, they are not an optimal solution for sentence-level embeddings. One trivial solution for sentence-level embedding is using arithmetic operations to combine the word vectors of a sentence, but this approach can alter the sentence's meaning. In response to these challenges, sentence-level embeddings were proposed to turn a sentence into a dense vector preserving the meaning of sentences in the terms that similar sentences will be given close vectors. Universal sentence encoder (USE) [11] and Fast Sentence Embedding (FSE) are examples of popular sentence-level embeddings.

## 3. Review of related works

There are two different objectives in the automatic rumor detection literature, including *event-level* and *tweet-level* rumor detections. The purpose of event-level approach is to identify the veracity of a general topic related to an event represented by a set of conversations with similar topic. Formally, given an event E containing conversations $C_1$ to $C_n$ (i.e. $E = \{C_1, \ldots, C_n\}$) the label $L(E)$ indicates whether the whole event is rumor or not. In contrast, in tweet-level approach, given a source tweet of each conversation, its responses, and some metadata about the tweets and users, the model should be able to decide whether the source tweet is rumor, non-rumor, or unverified. In fact, in tweet level view, for each conversation $C$, $L(C)$ specifies the veracity of its single source tweet.

One of the earliest attempts to automate rumor detection was undertaken by [12], in which the effectiveness of different feature categories was studied for identifying rumors. The proposed system can track known rumors but cannot detect new rumors on Twitter.

The first attempt to detect new rumors has been performed in [13], which uses the fact that users respond to rumors by asking questions about them, which was also reported by [14]. The proposed system utilizes conventional machine learning for detection of rumors. However, their system relies on handcrafted content features. Moreover, it operates at the event-level mode.

Another interesting work on automatic rumor detection based on conventional machine learning on Twitter is [15], which states that although users' stances used in [13] offers an indicator for detecting rumors, but detecting these stances itself is a big challenge. The proposed system leverages a few interesting and lesser-used features; however, it detects unverified stories and does not detect rumors in the context of false information. Moreover, the proposed system operates at the event-level, which was discussed before.

One of the first works in detecting rumors leveraging deep learning techniques is [7], which proposes to use RNN architecture, containing LSTM and GRU cells in detection of rumors. The proposed system considers content data, but it operates at the event level.

Yu et al. [16] have also employed deep learning techniques for rumor detection. They used Convolutional Neural Network (CNN) architecture, reasoning that the proposed system will be more suited due to the fact that RNN architecture is more biased towards the last elements of input while the indicators of rumor are not necessarily in the last elements of the input. They also point out that RNN architecture requires a lengthy input for reliable detection, while many microblog posts are short. Although their work is innovative in that few works are using CNN to detect rumors, but their system, like the ones before, works at the event level.

The closest research to our work is [17], which was later refined and presented as the baseline for RumourEval 2019 [18], and we also consider this research as the baseline for our work. Furthermore, working on this base code and the RumourEval framework makes evaluation of the work more straightforward and clear. In their work, the authors propose a system based on RNN architecture leveraging LSTM cells. One significant contribution of their work is that the proposed system uses some novel features as the feature vector, and it also detects rumors at the tweet level. However, their proposed system uses many handcrafted content features which makes it more susceptible to deception. In addition, it neglects social and profile features which can be potentially used for efficient rumor detection.

Another crucial work similar to our work is [19], which is the winner of SemEval-2019 and the state of the art system. This system is trained with Twitter data and has an exciting innovation that is using fine-tuned word-level embedding specific for the task of rumor detection. Unfortunately, due to the unavailability of the source code and development details of this system, few comments on this work can be made. The proposed system again uses many handcrafted features that makes their system more susceptible to deception. Furthermore, their proposed system relies on some machine learning systems that are still in R&D phase and are considered as open problems in the field of machine learning and natural language processing. To be more specific, this work relies on: 1) a system for detection of parts of sentences like named entities, verbs, etc.; 2) a system for detection of user stances; 3) a system to detect the topic area of the rumor. Hence the overall performance of the proposed method significantly depends on the performance of these underlying systems.

Again, a research related to our work is [15] in which we used some of the features that they have proposed for the task of detection. However, our work is different from [15] in some issues. We are trying to detect false rumors contrary to their work, which can only detect unverified stories. Also we work at the tweet level while their research detects rumors at the event-level.

Another interesting works in automatic rumor detection is [20], which takes a novel approach to detect rumors by leveraging spatial-temporal rumor aspects in social media. Other work is [21] that leverages multi-loss bidirectional RNNs for rumor detection. The work reported in [22] is also another exciting work in rumor detection that has utilized ensemble method for rumor detection.

Table 1 summarizes some of the most important aspects of the more relevant researches to our work. As it can be inferred from Table 1, most of the previous important researches developed rumor detection models in event level, hence they cannot judge about the veracity of each individual tweet. Many papers employed handcrafted features which results in low generalization in detecting new rumor forms and make these systems more prone to deception. Some systems used user's stance as a representative of the crowd wisdom about the specified tweet. Just one research has used the network and user profile features in detecting rumors. As a consequence, the proposed method is designed so that it operates in tweet level, it uses nearly all information sources including content, profile and network features along with users' stances. This is while the proposed method does not inherit the weaknesses of using handcrafted features.

## 4. Statement of the problem

In this research we aim to automatically detect rumors in Twitter. We attempt to develop a rumor detection system which is resilient to deception. Moreover, the system should detect rumors at tweet-level. The metadata includes profile and network features. Profile features are used to determine the user's credibility, while network features are used to show the state of the rumor propagation in the social network.

While automatic rumor detection has attracted the attention of many researchers over the past few years, a huge bulk of studies rely on handcrafted features which leads the developed models susceptible to deception. In psychological studies, deception is defined as an intentional and knowing attempt of the writer of a message to create a false deduction or belief in the reader's mind [23, 24]. Humans often do not detect fake contents, in most situations. It has been proved that people can distinguish a truthful statement from a lie with the accuracy of 54% which is just a bit above the random decision [25]. This fact highlights the role of automatic rumor detection under the intended deception process. When a statement is created with the aim of deceiving people, its content appearance should mimic a legitimate statement.

Thus a rumor detection system should detect the veracity of a message regardless of its appearance in order to have resiliency to deception. The appearance of the message can be defined in terms of punctuations, letters cases, image inclusion, and so on. Since most of the handcrafted features used in rumor detection task are describing the message appearance, the resulting models are prone to deception. In this study we attempt to propose a model for rumor detection which can detect rumors efficiently, while neglecting the appearance based features.

Furthermore, event-level approaches require large volume of messages in each topic which is not available in the first stages of rumor propagation. Thus, the aim of this research is developing a tweet-level rumor detection system that will be applicable for early detection. In Twitter, after a user posts a tweet, others can reply to it, and it is also possible to post a reply to a previous reply, and so on. This results in a tree structure of tweets and replies that is called a conversation. Each conversation can be broken up into several branches, each starting from the source tweet and ending at a tree leaf. It is possible to break the conversation into its branches by running a depth-first search on the tree, and each time the algorithm reaches a leaf, the current branch can be extracted by backtracking the steps.

To better understand the concepts of branch and conversation, Figure 1 shows an example, in which a conversation is represented in two branches. One branch containing the source, User1 and User2 posts and another including the source plus User3 and User4 posts.

Table 1. Important aspects of related researches

| Research | Operation level | Using handcrafted content features | Using user's stance | Using profile features | Using network features |
|---|---|---|---|---|---|
| Yu et al. [1] | Event | Low | Not used | Not used | Not used |
| Zhao et al. [2] | Event | Medium | Low(only inquires) | Not used | Not used |
| Ma et al. [3] | Event | Low | Not used | Not used | Not used |
| Li et al. [4] | Tweet | High | High (all possible stances) | Yes | Yes |
| Kochkina et al. [5] | Tweet | High | High (all possible stances) | Not used | Not used |
| Huang et al. [6] | Event | None | Not used | Not used | Yes |
| Sujana et al. [7] | Event | None | Not used | Not used | Not used |
| Mouli Madhav Kotteti et al. [8] | Event | None | Not used | Not used | only tweet time stamps |
| Proposed method | Tweet | None | High (all possible stances) | Yes | Yes |

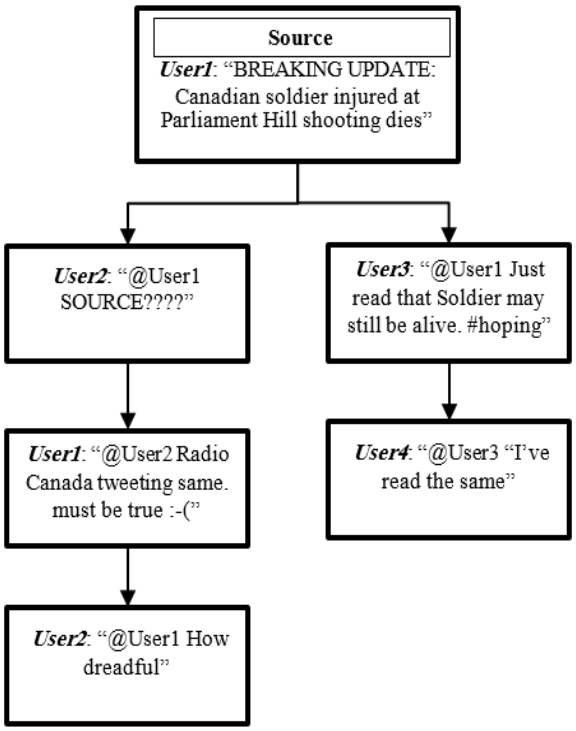


Figure 1. Branches of a conversation

## 5. The proposed method

In this section, we introduce the proposed method in detail. The section begins with the explanation of the system architecture and continues with the detail description of each individual phase.

### *5.1. System architecture*

The proposed method is composed of three main phases: preprocessing, feature extraction, and modeling. The overall architecture of the proposed system is depicted in Figure 2. As it is shown in this figure, having a row conversation *C*, it is decomposed initially into the set of branches {$b_1$, $b_2$, … $b_k$} where each branch $b_i$ is composed of the sequence of tweets $t_{i,1}$, $t_{i,2}$ , …., $t_{i,|b_i|}$ in which $t_{i,j}$ is the *i*th tweet of the *j*th branch. Then different features are extracted from the tweet contents and metadata. After that feature vectors can be used to train the model in the training phase and predicting the veracity as label *L*(*C*) in the testing phase.

### *5.2. Preprocessing*

Due to the tree's nature, the conversation data processing with a neural network is particularly challenging. To combat this challenge, some researches, such as [5, 9], suggested representation of conversation in terms of its branches. Therefore, the conversation is fed to the network branch by branch. Figure 3 shows the prepressing phase, in which branches $b_1$ to $b_k$ are first extracted and tweets of each branches are then extracted in terms of $t_{i,1},\ldots,t_{i,|b_i|}$ for each $b_i$ where $1 \le i \le k$. In this notation, $k$ is the number of different branches in $C$ and $|b_i|$ is the number of tweets in branch $b_i$.

### *5.3. Feature extraction*

Feature extraction phase is illustrated in Figure 4. For each tweet $t_{i,j}$, the corresponding network, profile and content features are extracted respectively and concatenated to form the overall tweet feature vector $\bar{t}_{i,j}$. The network and profile features are characterizing the social context of the tweet and content features are representing the text of the tweet itself. The feature vector associated with tweets of a branch are then concatenated to form the branch feature vector $\bar{b}_i = \{\bar{t}_{i,1},\ldots,\bar{t}_{i,|b_i|}\}$ and a conversation is finally represented as a set of branch feature vectors (i.e. $C = \{\bar{b}_1,\ldots,\bar{b}_k\}$). One of the innovations of our work is the novel feature set proposed for detection of rumors. Although many of these features have been used before in rumor detection, we have not seen them used together in other previous works. The proposed method also relies on user's stances based on the fact that the users' reactions to rumors are different from non-rumors, which was first pointed out in [10].

An important aspect of the proposed feature set is that we use word and sentence level embeddings for content feature extraction which makes the model more resilient to user deception.

The feature set we use can be described in the following three categories:

1. Profile features (the features of the user who posted the tweet):
- Number of followers
- Number of followings
- Whether the account is verified or not
- Number of total tweets
2. Network features (the features related to state of the propagation):
- Number of retweets of the tweet
- Number of likes of the tweet
- Whether the tweet is the source or response
- The stance of each tweet towards the source tweet with values of Supporting, Denying, Querying, and Commenting
3. Content features (the features of the tweet itself by dense vectors leveraging word and sentence level embeddings)
- Avg2Vec, used in [5, 9], uses Google's word2vec to create a sentence level embedding for tweets by averaging between the word level embedding vectors of the words in the tweet.
- Universal Sentence Encoder (USE), which is a sentence level embedding also introduced by Google.

It is also worth mentioning that the features used in our work can be categorized in two set:

1. Manually extracted features containing profile and network features;
2. Automatically extracted features containing content features.

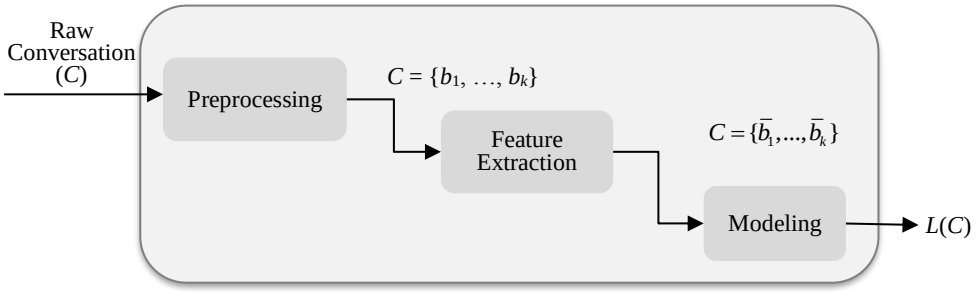


Figure 2. System architecture

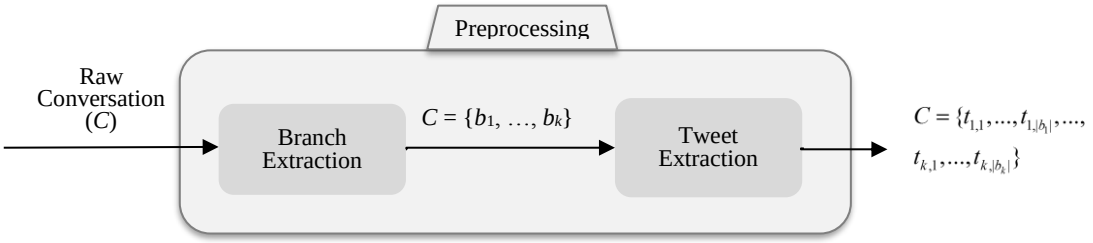


Figure 3. Preprocessing phase

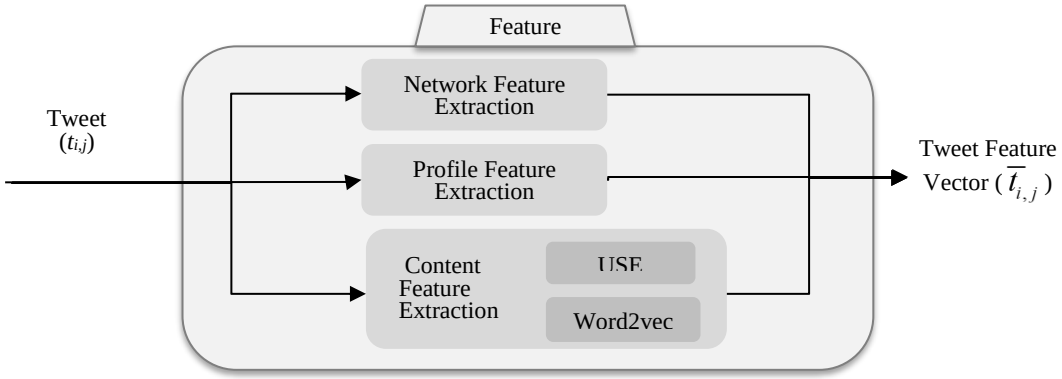


Figure 4. Feature extraction phase

**6. Modeling**

The core of the proposed system is the modeling phase that is comprised from training and testing phases. In training phase, the branch feature vectors along with the source tweet label are feed to RNN for learning. After training, the learnt model can be deployed in automatic rumor detection system as it is demonstrated in Figure 5. In this phase, given a conversation feature vector, the feature vector of each branch is extracted and inputted to RNN model. After predicting the corresponding labels for individual branches, a voting module is used to determine the final label for the conversation as the majority label predicted by its branches.

We employed RNN deep learning architecture because the nature of the input data in the underlying system is a sequence. It means that we want the learning system to recognize the patterns and relations between consecutive words, sentences and tweets in processing a conversation. Since RNN is memory-based architecture and learns sequences well, it is appropriate for our purpose. Furthermore, RNNs support learning sequences with variant lengths which is the case in rumor detection systems for branches. Since the tweet branches may form as long sequences, memory based unites such as LSTM and GRU are needed for learning these sequences. Using GRUs is more preferable because of their speed and efficiency and also to give the model less space for overfitting, which contributes to the overall model performance. We also leverage bidirectional GRUs for two reasons: 1) giving the model more information at each time step; 2) reducing the model's bias towards the end of sequence by processing the sequence from both directions.

The detail architecture of RNN units are revealed in Figure 6. The model is comprised of one bidirectional GRU Layer, and the output of this layer is passed to two dense layers with ELU (exponential linear units) function as their activation [26]. It is worth mentioning that before each layer, the input of that layer is normalized with the batch normalization layer. This has two effects:

1. The data is scaled and the training phase's noise is reduced, where in turn makes the training phase faster and more stable;
2. By feeding data in different batches, it has a slight regularization effect on the model, which reduces the chance of overfitting.

For improving the generalization of the network and avoid overfitting, L2-regularization mechanisms are adopted. The complexity of the network and subsequently the overfitting issue are controlled in this way.

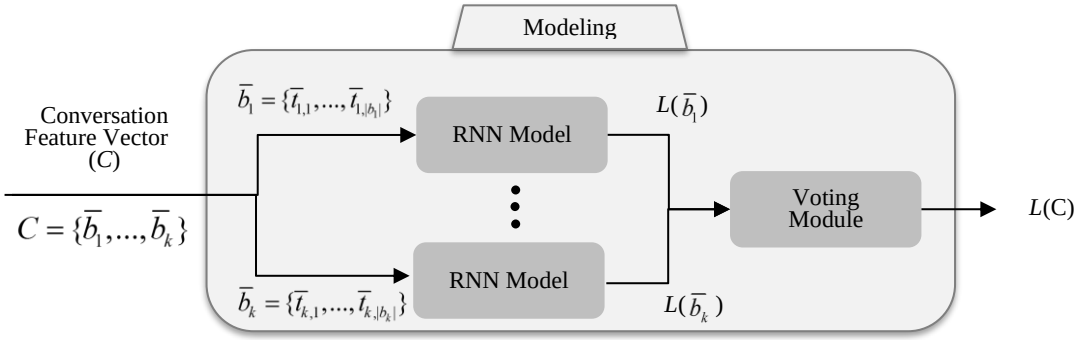


Figure 5. Modeling phase

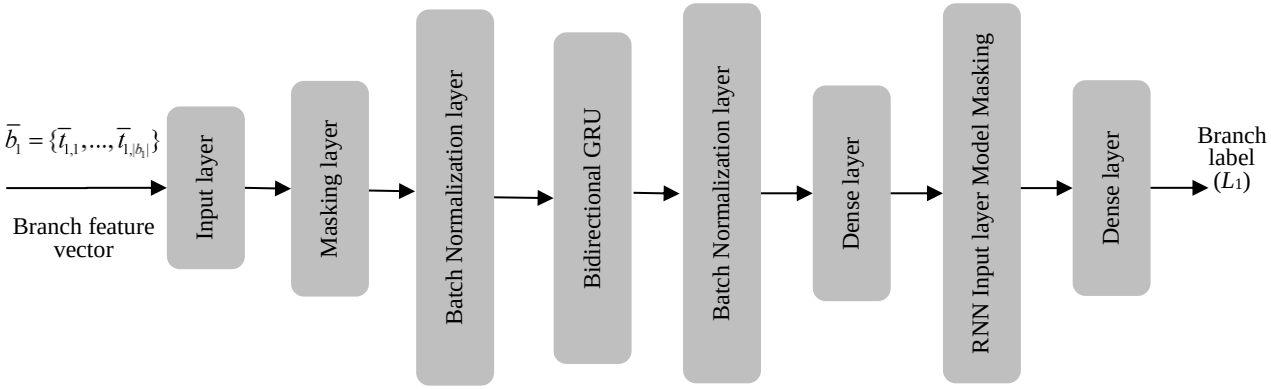


Figure 6. The architecture of the RNN unit

Due to social networks' dynamic nature, the rumor detection model should be adapted and retrained many times after deployment. To achieve this goal, we have leveraged CuDNN libraries in designing the model in order to improve the time complexity in the training phase. The use of CuDNN has one big downside: the loss of some layer features like dropout in GRU layers, but it helps a lot in the model's generalization. Therefore, there is a tradeoff between the time complexity and the generalization and hence we took the middle way through utilizing CuDNN.

## 7. Experimental evaluations and results

In this section after describing the evaluation configurations including the dataset specification and the system setup, the proposed rumor detection system is evaluated in terms of macro F-Score, time efficiency and deception resiliency. The experimental results are also compared to the results of the state of the art methods.

### *7.1. Dataset*

The dataset used in this research is associated with RumorEval 2019 competitions which is a refined and updated version of the Pheme dataset [27]. This dataset is comprised of conversations categorized into topics, each topic containing conversations with one of the below labels:

1. True: Conversations that are spreading verified information;
2. False: Conversations that are spreading rumors;
3. Unverified: Conversations that are spreading unverified information that was neither verified nor denied up to the time of their retrieval.

Table 2 shows the distribution of conversations and branches between the training, development, and testing sets, while Table 3 and Table 4 show the distribution of labels in conversations and branches in different sets. Note that the development dataset in RumoreEval context is equivalent to validation dataset known in machine learning literature.

Table 2. Distribution of conversations and branches

| | Train | Development | Test |
|---|---|---|---|
| Conversation | 297 | 28 | 56 |
| Branch | 3245 | 768 | 1010 |

Table 3. Label distribution in conversations

| | Train | Development | Test |
|---|---|---|---|
| True | 137 | 8 | 22 |
| False | 62 | 12 | 30 |
| Unverified | 98 | 8 | 4 |

Table 4. Label distribution in branches

| | Train | Development | Test |
|---|---|---|---|
| True | 1470 | 124 | 341 |
| False | 549 | 514 | 558 |
| Unverified | 1226 | 130 | 111 |

### *7.2. Setup*

All of the experiments were run on a single system with the same hardware setting for all of them. Table 5 shows the system details.

Table 5. System setup

| | |
|---|---|
| Processor | Intel Core i7 6700HQ |
| RAM | 16GB DDR4 2133MHz |
| GPU | Nvidia GeForce GTX 980M |

One of the most important parts of every deep learning model is the hyper parameters setting presented in Table 6. Hyper parameters include parameters that define the model and cannot be learnt by the model during the training phase. In order to find the optimal hyper parameters, we leveraged Tree-structured Parzen Estimator (TPE) algorithm [11] implemented in python library called Hyperopt which helps automating some of the tasks in hyper parameter search and model tuning. TPE is a sequential model-based optimization approach, which sequentially estimates the conditional probability density function of the objective function based on hyperparameters. In each iteration, the next set of hyperparameters are configured based on their evaluation on the estimated probability model and the model is refined accordingly. Sequential model-based optimization is a formalization of Bayesian optimization which is more efficient than random or grid search in finding the best set of hyperparameters [11]. Table 6 shows the best hyper parameters found by TPE for the proposed system.

Table 6. Hyper parameter setting

| Hyper Parameter | Value |
|---|---|
| Number of GRU Layers | 1 |
| Number of GRU Units | 400 |
| Number of hidden dense layers | 2 |
| Number of dense units in 1st dense layer | 600 |
| Number of dense units in 2nd dense layer | 400 |
| Training Steps | 50 |
| L2 Regularization Parameter in 1st dense layer | 1e-4 |
| L2 Regularization Parameter in 2nd dense layer | 1e-4 |
| L2 Regularization Parameter in output layer | 1e-6 |
| L2 Regularization Parameter in GRU layer | 1e-6 |
| Minibatch size | 64 |
| Optimization Algorithm | Adam |

### *7.3. Overall performance*

Table 7 shows the results of evaluation metrics of the proposed method as well as those in [5, 9] as the baseline. The performance is measured in terms of precision, recall, and F1-score. It can be deduced from the table that our model outperforms the baseline in the overall metric used by the RumourEval 2019 competitions (i.e., Macro-F1 Avg.). A more detailed look shows that the proposed method outperforms the baseline in the rue class but slightly lags behind it in the other classes. It is due to low false negative rate of the proposed method which is a critical necessity of a rumor detection system. The performance of the baseline can be attributed to many features, but as we will show later, this gives their model a significant disadvantage regarding resilience to deception.

The results of all RumorEcval 2019 participants can be found in [9]. As it can be inferred from the table, the overall performance of the proposed method is better than other models. There are also some works like [4] that uses some auxiliary data for training. Utilizing auxiliary dataset gives the model some advantages and not only makes the comparison a little unfair, but also we believe it threatens the scalability of the method. When the model is trained and evaluated based on the auxiliary datasets, its performance is not guaranteed for rumor detection in other environments in which this data volume is not available.

Table 7. Comparison to the baseline

| | Class | Precision | Recall | F1 |
|---|---|---|---|---|
| Baseline [5, 9] | True | - | - | 0.31 |
| | False | - | - | 0.53 |
| | Unverified | - | - | 0.17 |
| | Macro Avg. | - | - | 0.33 |
| Proposed method | True | 0.85 | 0.37 | 0.51 |
| | False | 0.48 | 0.45 | 0.47 |
| | Unverified | 0.05 | 0.25 | 0.08 |
| | Macro Avg. | 0.46 | 0.36 | 0.35 |

Table 8 shows the performance comparison of the proposed method with the most successful related models, which operate on RumorEval 2019 dataset.

Table 8. Comparison to other models

| Model Name | Macro-F1 score |
|---|---|
| Baseline [5, 9] | 0.33 |
| VANTA and Aono[12] | 0.32 |
| WeST (CLEARumor) [13] | 0.28 |
| GWU NLP LAB [14] | 0.26 |
| BLCU NLP [15] | 0.25 |
| FINKI NLP (reported in [9]) | 0.33 |
| EventAI [4] | 0.58 |
| Proposed method | **0.35** |

### *7.4. Resilience to deception*

Since many rumors are created with the aim of user deception, the appearance of the claim is designed to mimic a legitimated news post. A successful rumor detection system should not be sensitive to simple apparent signs. In the proposed method, we tried to develop a model that is resilient to these changes. To evaluate the models' resilience to deception, we propose changes to the tweet text, keeping in mind that it is entirely in the user's control and can be changed easily without changing the tweet's overall meaning. The applied changes, enumerated below, are minimal and do not affect the meaning of the text:

1. Removing the periods or adding one if does not exists any;
2. Removing question marks or adding one if does not exists any;
3. Removing exclamation mark or adding one if does not

exists any;
4. Removing pictures or adding one if does not exists any;
5. Changing the capital ratio of the characters to a random value.

We assume that these changes can simulate the changes that a rumor creator made intentionally in the rumor content so that it looks like a normal verified claim. These changes are chosen regarding the experiences reported in [5, 9], which is the only model that shared its details and code. After applying the changes on the test set, both models trained on the original training set (the proposed model and the baseline model) were rerun on the modified test set, and the results were compared to the original run. Since our model ignores all the mentioned handcrafted content features due to the use of embeddings for extracting features from the text, these changes do not affect the model's performance. In contrast, the predicted labels in works reported in [5, 9] were changed in 34% of the conversations after applying the modifications. It can be inferred from the experiment that even simple text changes can easily mislead the baseline model.

Since the described issue arises due to the employment of handcrafted features, it seems that other researches that model the rumor based on these features (e.g., [4]) also suffer from the similar weaknesses. In fact, current experiment compares the resiliency of two categories of approaches to deception, the models based on automatic feature extracted and the models based on handcrafted features. To this end, the proposed method and the baseline are selected as representatives for these two categories of approaches, in the absence of source code of other related methods.

This test shows the downside of handcrafted features, especially for content features, since they are in the control of the user and can be changed easily. For that reason, all the content features used in the proposed method are 1) the ones that are determined by the network and are not controlled by the user, or 2) the ones that are extracted using methods like embedding that focuse on the meaning instead of the looks which minimizes the user's influence on the model.

### *7.5. Time efficiency*

Table 9 shows the training time of the proposed method and the baseline model. It can be seen that our model outperforms the baseline significantly in training time, giving it a valuable advantage for deployment. This means it saves much time in training, which leads to savings in resources and capital making it more suitable for deployment.

Although, as discussed before, our model cannot use dropout in the GRU layer, which can give it a significant advantage in generalization, we showed that it outperforms the baseline while being much faster in training.

Table 9. Time efficiency

| Model | Training Time (seconds) |
|---|---|
| Baseline [5, 9] | 2406.13 |
| Proposed method | 40.25 |

### *7.6. The role of profile and network features*

Another exciting aspect that needs discussing is the proposed feature set. Regarding content features, we have already shown that using embeddings instead of handcrafted features gives our model a significant advantage regarding resilience to deception. Regarding other features, it can be deduced that all of the profile and network features are a part of the social network and they are out of the control of the user, especially for trending topics, and one or a group of users cannot meaningfully change them to mislead the model.

We can also show that the proposed non-content features are needed to achieve the results shown in Table 7. Table 10 compares the model's performance using the proposed feature set to the model using only the content features. It shows that the full feature set outperforms the content features, which in turn shows that network and profile features provide essential information for rumor detection.

Another important aspect of our model is the use of GRU cells instead of LSTM. Although LSTMs are more common in RNN architecture, as discussed before, the use of GRU leads to reducing the training time and increasing the generalization.

Table 10. The role of different feature sets

| Feature Set | Macro-F1 Avg. |
|---|---|
| Content Features only | 0.31 |
| Full Feature Set | 0.35 |

Table 11 compares the proposed method with the same model with LSTMs instead of GRUs. It can be seen that the GRU model slightly outperforms the LSTM with fewer parameters and much faster run time.

Table 11. The comparison of GRU and LSTM unites

| Model | Macro-F1 |
|---|---|
| LSTM based model | 0.34 |
| Proposed method (GRU) | 0.35 |

## 8. Conclusion

While a considerable research effort has been done recently to develop automatic rumor detection models, most of prior approaches have had the problem of relying on handcrafted features. Using these features make the model more susceptible to deception and reduces the scalability of the system. Moreover, a great deal of work is devoted to event level rumor detection which is not applicable for early detection and prevention in real world. This research proposed a rumor detection system based on RNN model and GRU cells for specifying the veracity of tweets in Twitter network. One of the most important innovations of this research is a novel feature set that avoids the extraction of handcrafted content features and uses network and profile features that are out of users' control. Considering these features makes the model more resilient against deception. We focused on efficiency and scalability, especially in the training phase, keeping in mind that social networks' dynamic nature requires the model to be retrained many times to adapt to the users and network behavioral changes, making our model more suitable for deployment.

A number of experiments were conducted to analyze the effectiveness of the proposed rumor detection system. Experimental results show that the proposed method outperforms most similar research in terms of macro F-score.

It also revealed that the proposed system is less prone to deception. Furthermore, the results indicate the superiority of the proposed method comparing the baseline in terms of time efficiency. Consequently, the proposed rumor detection system is suitable for being applied efficiently in early tweet-level rumor detection task with remarkable tolerance to deception.

As the future work in our research direction, we tend to use pertained pre-trained contextual deep neural networks for both content embedding and tweet classification tasks in order to improve the overall performance.

**9. References**

[1] Yu, F., Liu, Q., Wu, S., et al., "A Convolutional Approach for Misinformation Identification", *In: IJCAI International Joint Conference on Artificial Intelligence*, **pp.** 3901-3907**,** 2017.

[2] Zhao, Z., Resnick, P., Mei, Q., "Enquiring minds: Early detection of rumors in social media from enquiry posts", *In: WWW 2015 - Proceedings of the 24th International Conference on World Wide Web*, **pp.** 1395-1405**,** 2015.

[3] Ma, J., Gao, W., Mitra, P., et al., "Detecting rumors from microblogs with recurrent neural networks", *In: IJCAI International Joint Conference on Artificial Intelligence*, **pp.** 3818-3824**,** 2016.

[4] Li, Q., Zhang, Q., Si, L., "eventAI at SemEval-2019 Task 7: Rumor Detection on Social Media by Exploiting Content", User Credibility and Propagation Information, 2019.

[5] Kochkina, E., Liakata, M., Augenstein, I., Turing at SemEval-2017 Task 8: Sequential Approach to Rumour Stance Classification with Branch-LSTM, 2018.

[6] Huang, Q., Zhou, C., Wu, J., et al., "Deep spatial–temporal structure learning for rumor detection on Twitter. Neural Comput Appl", https://doi.org/10.1007/s00521-020-05236-4, 2020.

[7] Sujana, Y., Li, J., Kao, H-Y., "Rumor Detection on {T}witter Using Multiloss Hierarchical {B}i{LSTM} with an Attenuation Factor", Aacl, 2020.

[8] Kotteti, C. M. M., Dong, X., Qian, L., "Ensemble deep learning on time-series representation of tweets for rumor detection in social media", Appl Sci 10:. https://doi.org/10.3390/app10217541, 2020.

[9] Gorrell, G., Kochkina, E., Liakata, M., et al., "SemEval-2019 Task 7: RumourEval", Determining Rumour Veracity and Support for Rumours, 2019.

[10] Mendoza, M., Poblete, B., Castillo, C., Twitter under crisis: Can we trust what we RT? In: SOMA 2010 - Proceedings of the 1st Workshop on Social Media Analytics, 2010.

[11] Bergstra, J., Bardenet, R., Bengio, Y., Kégl, B., "Algorithms for hyper-parameter optimization", *In: Advances in Neural Information Processing Systems 24: 25th Annual Conference on Neural Information Processing Systems 2011*, NIPS, 2011.

[12] Vanta, T., Aono, M., "Stance Classification and Rumor Analysis: Using New Dialog-Act Features and Augmenting Input Tweets", *In: 2020 7th International Conference on Advance Informatics: Concepts, Theory and Applications (ICAICTA).* IEEE, pp 1–6, 2020.

[13] Baris, I., Schmelzeisen, L., Staab, S., CLEARumor at SemEval-2019 Task 7: ConvoLving ELMo Against Rumors, 2019.

[14] Hamidian, S., Diab, M., GWU NLP at SemEval-2019 Task 7: Hybrid Pipeline for Rumour Veracity and Stance Classification on Social Media,2019.

[15] Yang, R., Xie, W., Liu, C., Yu, D., BLCU_NLP at SemEval-2019 Task 7: An Inference Chain-based GPT Model for Rumour Evaluation,2019.