applied sciences

MDPI

Article

Ensemble Deep Learning on Time-Series Representation of Tweets for Rumor Detection in Social Media

Chandra Mouli Madhav Kotteti †, Xishuang Dong † and Lijun Qian *,†

Center of Excellence in Research and Education for Big Military Data Intelligence (CREDIT Center), Prairie View A&M University, Texas A&M University System, Prairie View, TX 77446, USA; ckotteti@student.pvamu.edu (C.M.M.K.); xidong@pvamu.edu (X.D.)

* Correspondence: liqian@pvamu.edu

† These authors contributed equally to this work.

Received: 16 September 2020; Accepted: 19 October 2020; Published: 26 October 2020

Abstract: Social media is a popular platform for information sharing. Any piece of information can be spread rapidly across the globe at lightning speed. The biggest challenge for social media platforms like Twitter is how to trust news shared on them when there is no systematic news verification process, which is the case for traditional media. Detecting false information, for example, detection of rumors is a non-trivial task, given the fast-paced social media environment. In this work, we proposed an ensemble model, which performs majority-voting scheme on a collection of predictions of neural networks using time-series vector representation of Twitter data for fast detection of rumors. Experimental results show that proposed neural network models outperformed classical machine learning models in terms of micro F1 score. When compared to our previous works the improvements are 12.5% and 7.9%, respectively.

Keywords: social media; rumor detection; time-series data; machine learning; deep learning; Twitter

1. Introduction

Over the past few decades social media has emerged out as the primary means for news creation as well as for news consumption. Given the speed at which information travels on social media, it is very easy to propagate any type of news and it can be consumed instantly across the globe at the early stages of its propagation process. However, the biggest challenge for news spreading on social media is how to verify whether that news is correct or not. Even though social media outperforms traditional media in many aspects, the key difference between them is that the news is verified for its truthfulness before it gets proliferated in traditional media, while it is not the case for social media. Thus, any piece of information can be easily spread on social media regardless of its truthfulness.

Furthermore, information shared on social media propagates rapidly and increases the difficulty in verifying its credibility in near real time. A rumor is defined as a "circulating story of questionable veracity, which seems credible but hard to verify [1], and produces sufficient skepticism and anxiety", and it could have three values such as true, false or unverified [2]. It is difficult to directly determine whether a social media statement is a rumor or not. Thus, ontologies of rumors can be helpful in modeling rumors, for example, the Pheme ontology. In the Pheme ontology, a statement that is expressed in the texts is considered as a Pheme [3]. A rumor is considered as a direct subclass to Pheme, which has four sub-classes. They are speculation, controversy, misinformation, and disinformation. Detection of rumors in social media has a lot of importance among research communities because unverified information may be easily disseminated over a large network, and rumors may spread misinformation or disinformation (misinformation means information that is incorrect in its nature

and disinformation means information that is used to deceive its consumers), which are forms of false information [4,5].

If the spread of false information is not stopped early it may cause turmoil in the society. In case of time critical events, the effects may be dreadful. So detecting rumors in social media must be done in a timely fashion. Recently machine learning and deep learning gained huge popularity in addressing rumor detection in social media [6], and they typically applies trained classification models to predict new data samples as rumors or non-rumors [7]. One of the main concerns for applying these techniques is finding a dataset with good quality. On the other hand, performing extensive feature engineering on the dataset to extract a variety of useful features, for example, content-based and context-based features, for the rumor identification problem may help in improving a classification model's performance. However, it will significantly slow down the training procedure since employing complex features in training process is cumbersome in terms of computational complexity and availability of hardware resources to deal with extremely large sized feature set [8]. Hence, extensive feature engineering may not be suitable for timely rumor detection.

In this paper, we explore the temporal features of Twitter data for timely detection of rumors in social media. Tweet creation timestamp can readily be extracted from tweets, and there is no time delay to collect timestamp features and no sophisticated data pre-processing is required to convert them into useful features to train a classification model. Based on this observation, we proposed an ensemble-based multiple time-series analysis model using deep learning models for timely detection of rumors in social media. Specifically, we generated time-series data by transforming Twitter conversations, where each conversation contains a list of tweets, into times-series vectors that contain reaction counts (i.e., the total number of reactions per time step along a conversation) as features, and fed them as input to deep learning models. The contributions of our proposed method are:

- With the proposed method, computational complexity can be significantly reduced as we just need timestamps of tweets rather than their contents or user social engagements to perform feature extraction. Moreover, the extracted feature set is of numeric type, which is amicable to classification models.
- Our proposed ensemble model improves the performances of classification models since it uses the majority-voting scheme on multiple neural networks that are part of the ensemble model and takes advantage of their individual strengths.
- We validated our proposed method on the PHEME (https://figshare.com/articles/PHEME_dataset_for_Rumour_Detection_and_eracity_Classification/6392078) dataset and the performance results demonstrate the effectiveness of the proposed scheme.

In summary, the proposed method explores the temporal features of Twitter data through proposing an ensemble-based classification model for the fast detection of rumors in social media.

2. Problem Formulation

2.1. Rumor Detection

Rumor detection involves identifying whether a data sample is a rumor or not. In machine learning, this kind of problem is termed as a classification task, in which the classification model gets trained with adequate number of training samples and tries to classify a never before seen testing sample as rumor or not. Therefore, the problem is given by $\hat{y} = f(X)$, where f is the classification model and X is a completely new data sample (a Twitter conversation sample that is transformed into a time-series vector) to it, and $\hat{y}$ is the prediction of the classification model and it has only two values since the PHEME dataset has two classes. In our work, we used 0s and 1s to represent non-rumor and rumor samples, respectively, i.e., $\hat{y} \in \{0, 1\}$.

2.2. General Features of Tweets

Typically, for a classification task using machine learning or deep learning requires extraction of useful features from the dataset. A variety of features can be extracted from Twitter data, for example, four types of features are extracted from Twitter data for the study on spread of anomalous information in social media [9]: user profile features (users' friends and followers count), user network features (users' EgoNet features), temporal features (retweet count), and content features (e.g., whether a tweet has question mark). In the case of content-based features, word embeddings and n-gram model are well known techniques applied for natural language processing tasks. A word embedding represents words in the text in a way that words sharing the similar meaning have a similar representation. An n-gram model transforms text into a sequence of N words. However, based on the theories of rumor propagation, authors in [10] considered temporal features as one of the key properties for studying spread of rumors since rumormongers have a short attention [11]. In this work, for the fast detection of rumors on social media, we solely focused on the temporal features of Twitter data, which are the creation timestamps of tweets. These timestamps can be readily fetched, and our work strictly relies on them for generation of time-series data, which involves simple calculations, i.e., counting of number of tweets for a given time interval limits.

2.3. Feature Extraction

In general, for Twitter data we use a parser to read and extract required information from it by depending up on its data type. In our work, the Twitter data we utilized is available in *JSON* format and we used pandas and dateutil's parser module to read that information and to extract our required features, which are the creation timestamps of tweets. For instance, the format of the creation timestamp value of a tweet is Thu Nov 22 20:45:24 +0000 2012. We parse it into a date-time format, i.e., 22 November 2012 20:45:24+00:00.

3. Ensemble Learning

3.1. Overview of Ensemble Learning

Ensemble learning is a concept in which many weak or base learners try to solve a single problem. An ensemble contains a number of base learners and its generalization ability is powerful than that of the base learners [12]. Ensemble methods work on a set of hypotheses derived from training data rather than relying on one hypothesis. Constructing ensembles is a two-step process. At first, required number of base learners are produced. Secondly, all the base learners are grouped and typically majority voting is applied for classification problems, and weighted averaging combination schemes are used for regression problems. Popular ensemble methods are boosting [13], bagging [14], and stacking [15]. Boosting method focuses on fitting multiple weak learners sequentially, where each model in a sequence gives more emphasis to the data samples that were badly treated by its previous model. AdaBoost [13] algorithm is a good example of boosting, which is simple and can be applied to data that is numeric, textual, etc. In the bagging method, multiple bootstrap samples are generated from the training data, and an independent weak learner is fitted for each of these samples. Finally, all the predictions of weak learners are aggregated to determine the most-voted class. The Random Forests [16] algorithm is good example of the bagging method, which is one of the most accurate learning algorithms and runs efficiently on large databases. In the stacking method, by using different learning algorithms, multiple first-level individual learners are created, and these learners are grouped by a second-level learner (meta-learner) to output a prediction [15].

3.2. Bagging Learning

Bagging learning has been studied extensively in the literature. Bagging, also known as bootstrap aggregation, is a popular ensemble method that is useful in reducing the high variance of machine learning algorithms. In the bagging technique, several datasets are derived from the original training

data set by employing sampling with replacement strategy, which means some observations in the derived datasets may be repeated. These datasets are used to train classification or regression models, and outputs of them are typically weighted averaged for regression cases or majority voted for classification problems.

The majority voting grouping technique is used in [17,18]. In [17], the bagging method of ensemble is used with REPTree as base classifier for intrusion detection systems, and compared to other traditional machine learning techniques. It is shown that the ensemble bagging method achieved high classification accuracy by employing NSL_KDD dataset. The authors of [18] proposed to use dictionary learning with random subspace and bagging methods, and introduced Random Subspace Dictionary Learning (RDL) and Bagging Dictionary Learning (BDL) algorithms. Their experimental analysis concluded that ensemble based dictionary learning methods performed better than that of single dictionary learning.

The weighted averaging grouping technique is employed in [19,20]. In [19], a Neural Network Ensemble (NNE) approach is proposed to improve the generalization ability of neural networks, and to reduce the calculation errors of Density Functional Theory (DFT). It is shown that both simple averaging and weighted averaging grouping techniques helped in improving DFT calculation results. The authors of [20] proposed a method for improving image classification performance using SVM ensembles. Optimal weights for the base classifiers in the SVM ensemble are estimated by solving a quadratic programming problem. These weights are then used to combine the base classifiers to form an SVM ensemble.

Optimization of a generic bagging algorithm is studied in [21]. The authors added an optimization process into the bagging algorithm that focuses on selecting better classifiers, which are relatively efficient, and proposed a Selecting Base Classifiers on Bagging (SBCB) algorithm. Experimental results proved that their SBCB algorithm performed better than the generic bagging approach.

3.3. Deep Bagging Learning

Because deep neural networks are nonlinear methods and have high variance, ensemble learning can combine the predictions of multiple neural network models in order to achieve less variance among the predictions and to decrease the generalization error. The ensemble method is applied to neural networks mainly by (1) varying training data (data samples used to train models in the ensemble are varied), (2) varying choice of the models in the ensemble, and (3) varying the combination techniques that determine how outputs of ensemble members are combined.

In [22], the authors proposed a method that uses Convolutional Neural Network (CNN) and deep residual network (ResNET) ensemble-based classification methods for Hyperspectral Image (HSI) classification. Their proposed method uses deep learning techniques, random feature selection, and a majority voting strategy. Moreover, a transferring deep learning ensemble is also proposed to make use of the learned weights of CNNs. In [23], two cooperative algorithms namely NegBagg (bagging is used) and NegBoost (boosting is used) are proposed for designing neural network (NN) ensembles. These algorithms use negative correlation algorithms while training NNs in the ensemble. Applying these models to well-known problems in machine learning showed that with a lesser number of training epochs, compact NN ensembles with good generalization are produced.

In [24], a bagging ensemble is proposed to improve the prediction performance of artificial neural networks (ANN) to tackle the bankruptcy prediction problem. Experimental results showed that the proposed method improved performance of ANNs. The bagging technique using an ANN is proposed to address imbalance datasets on clinical prediction in [25], and experimental results showed that this method improved the prediction performance.

3.4. Overview of the Proposed Model

Our proposed model has two key components: the data pre-processing method and ensemble model. Firstly, raw Twitter conversations are processed to transform them into the required data

format and then the transformed data are supplied to the ensemble model to perform classification. The ensemble model consists of six different neural networks (base learners) that are trained using the generated time-series data and their predictions are grouped such that a majority voting scheme is applied on them to determine the outcome as rumor or non-rumor.

4. Methodology

The structure of our proposed model is shown in Figure 1. The model takes Twitter conversations as an input, where each conversation is a stream of tweets that contains source-tweet and its corresponding reactions. In the data pre-processing stage, we parsed every tweet and extracted its creation timestamp value. Once all tweets were parsed, we generated time-series data for different time intervals and conducted data cleaning on it. We pre-processed the data by reducing time-series data sparsity, normalizing the data, and removing duplicate data samples. Then we fed that cleaned data as input to the ensemble model. The ensemble model has n base learners, which are n different neural networks that are represented as $m_1, m_2, \cdots, m_n$, where each of them yields its individual prediction results (i.e., $r_1, r_2, \cdots, r_n$). Finally, we performed the majority-voting process on all the predictions of those base learners, i.e., summing up all the prediction results and deciding the final prediction result as 0 (non-rumor) if the total sum is less than $\lfloor n/2 \rfloor + 1$ or as 1 (rumor) otherwise.

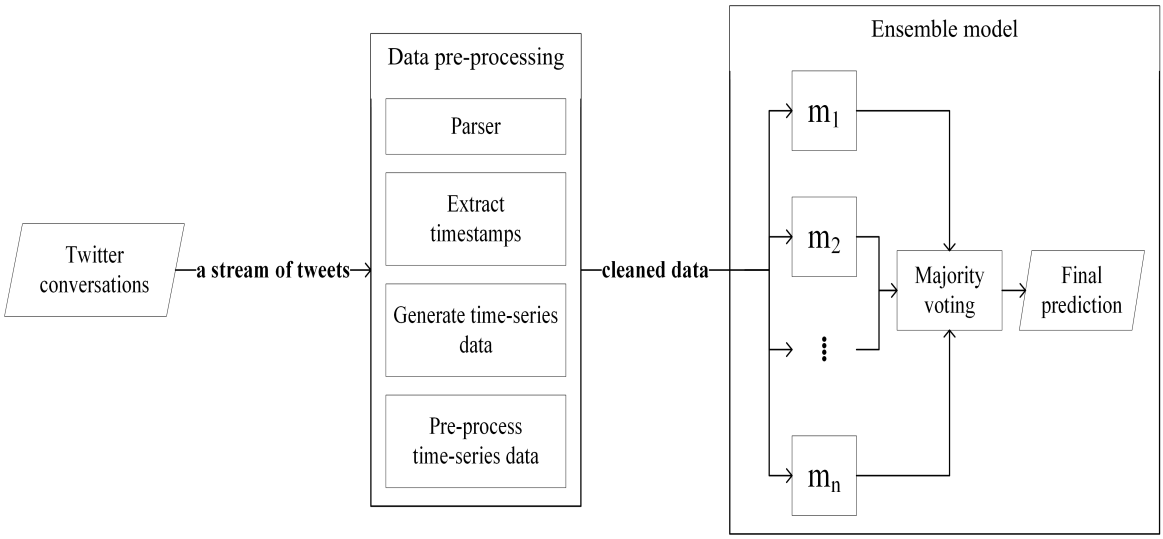

Figure 1. Proposed model for rumor classification taking Twitter conversations as input, which are cleaned in the data pre-processing block and fed as input to the ensemble model that performs the majority voting to determine the final prediction.

4.1. Neural Networks Models Considered

The ensemble model constitutes base learners designed using Recurrent Neural Network (RNN), Long Short-Term Memory (LSTM), Gated Recurrent Unit (GRU), and Bi-directional Recurrent Neural Network (Bi-RNN). Six base learners are designed in this work: BiGRU, BiLSTM, GRU, LSTM, LG (a neural network designed using a combination of LSTM and GRU layers), and RNN.

4.1.1. RNN

An RNN is a type of neural network that processes sequences by iterating through the sequence elements [26]. Typically, it consists of a hidden state $\mathbf{h}$, and an optional output $\mathbf{y}$ for a given variable length input sequence $\mathbf{x} = (x_1, \cdots, x_T)$. At each time t, the hidden state $\mathbf{h}_{(t)}$ is given by [27]:

$$\mathbf{h}_{(t)} = f(\mathbf{h}_{(t-1)}, x_t), \tag{1}$$

where f is a non-linear activation function. We used *Keras'* `SimpleRNN` [28] layer in our experiments.

4.1.2. LSTM

It is a special type of RNN and has been developed by Hochreiter and Schmidhuber in 1997 [29]. It consists of four major components, which are called cell, forget gate, input and output gate. The component cell functions to memorize values over arbitrary time intervals and three gates regulate flow of information into or our cell [26]. Each jth LSTM unit has a memory c_t^j at time t and the output h_t^j is given by [30]:

$$h_t^j = o_t^j \tanh(c_t^j), \tag{2}$$

where o_t^j is an output gate.

4.1.3. GRU

Chung et al. in 2014 [30] developed Gated Recurrent Unit, which has architecture similar to LSTM. There is no output gate in GRU, which means it has lesser number of parameters than LSTM. To control flow of information it uses update and reset gates, these gates decide how much of past information should be passed along to future or discarded [26]. Linear interpolation between h_{t-1}^j and $\tilde{h}_t^j$, which are previous activation and candidate activation respectively at time t is the activation h_t^j [30]:

$$h_t^j = (1 - z_t^j)h_{t-1}^j + z_t^j \tilde{h}_t^j, \tag{3}$$

where z_t^j is an update gate.

4.1.4. Bi-RNN

A traditional RNN processes the time-steps in order, whereas Bi-RNN [31] exploits the order sensitivity present in RNN and the input sequence can be processed in forward and reverse directions. It may have overfitting issues as it has twice the number of parameters of a traditional RNN, however, overfitting problem can be controlled by employing good regularization techniques [26]. We employed RNN variants GRU and LSTM layers in our experiments. The forward and backward hidden sequences (i.e., $\overrightarrow{h}$ and $\overleftarrow{h}$) for Bi-RNNs are given by:

$$\overrightarrow{h}_t = \mathcal{H}(W_{x\overrightarrow{h}} x_t + W_{\overrightarrow{h}\overrightarrow{h}} \overrightarrow{h}_{t-1} + b_{\overrightarrow{h}}) \tag{4}$$

$$\overleftarrow{h}_t = \mathcal{H}(W_{x\overleftarrow{h}} x_t + W_{\overleftarrow{h}\overleftarrow{h}} \overleftarrow{h}_{t+1} + b_{\overleftarrow{h}}), \tag{5}$$

where the W terms denote weight matrices, the b terms denote bias vectors, and $\mathcal{H}$ is the hidden layer function [32].

Once the base learners $(m_1, m_2, \cdots, m_n)$ complete their training procedures, the ensemble model combines all of their predictions, and performs majority voting procedure on them to determine the ensemble model's evaluation metrics. At first, we created our proposed ensemble model that consists of six base learners. Then we experimented on the proposed model by tuning its hyperparameters such as its batch input size and learning rate, and also created new ensemble models using RNN, LSTM, and GRU layers to obtain a comprehensive set of results to analyze and determine the effectiveness of each ensemble model in efficiently detecting rumor Twitter conversations. Specially, we examine hyper-parameter affection to model performance including batch size and learning rate. In addition, variants of the ensemble model will also have six base learners. Hence, in total there are three implementations of the proposed ensemble model, where some of the base learners are chosen based on the experiments we performed in our previous work [33].

4.2. Implementation-1

In implementation 1, each of five base learners (BiGRU_1, BiLSTM_1, GRU_1, LSTM_1, and simple RNN_1) has one hidden layer and the sixth based learner (LG_1) has two hidden layers, followed by one output Dense layer. For all the base learners, the number of hidden layer units is determined based on the integer value obtained from $(seq\_len + 2)/2$, where *seq_len* is the length of the feature set (i.e., vector length of the time-series data) and constant 2 is used because number of classification outputs are two (rumor and non-rumor). We considered this approach by following one of rule-of-thumb methods, which states that the number of hidden layer neurons should be between the size of the input layer and the size of the output layer [34]. $RandUniform$ kernel initializer is used for all the hidden layers with values $(-0.5, 0.5)$. *sigmoid* activation is applied only to the RNN model's hidden layer, and Flatten layer is applied only to BiGRU and BiLSTM models to flatten the data before the final output Dense layer that is activated using $softmax$ function. Adam optimizer is used with learning rate 1×10^{-5} along with *categorical cross-entropy* loss function. Batch input size is set to 32 and number of epochs is 300. We did not use the Dropout technique with these models since their architectures are simple, and using it may cause under-fitting issues. The variants of the proposed model follow the same neural network design except for the hyperparameter that is tuned, for example, batch input size and learning rate.

4.3. Implementation-2

Six base learners (RNN_1, RNN_2, RNN_3, GRU_1, GRU_2, and GRU_3) have been used in this implementation. To create new ensembles with new base learners, we used RNN, LSTM, and GRU layers. For instance, for base learners designed using RNN layer, we reused the RNN_1 base learner designed for implementation 1, and created new base learners by adding extra hidden layers with increasing (RNN_2) and decreasing (RNN_3) number of hidden layer units. The configurations of the base learners are shown in Table 1. All these base learners are having final output dense layer with $softmax$ activation and loss function as *categorical cross-entropy*. $RandUniform$ kernel initializer with values $(-0.5, 0.5)$. The number of training epochs is set to 300. For RNN_1, GRU_1, and LSTM_1 base learners in Table 1, *seq_len* is the length of the feature set.

Table 1. Configurations of NN models.

NN Model	# of Hidden Layers	Hidden Layer Units	Dropout
RNN_1	1	$(seq\_len + 2)/2$	N/A
GRU_1			
LSTM_1			
RNN_2	3	16, 32, 64	0.25
GRU_2			
LSTM_2			
RNN_3	2	64, 32	
GRU_3			
LSTM_3			

4.4. Implementation-3

Similar to implementation 2, six base learners (RNN_1, RNN_2, RNN_3, LSTM_1, LSTM_2, and LSTM_3) are employed in implementation 3. The hyperparameters have been set similarly.

Figure 2 shows the ensemble models designed using the above three implementations.

Figure 2. Ensemble models designed using three different implementations.

5. Dataset

5.1. PHEME Dataset

In this work, we used the PHEME [35] dataset of rumors and non-rumors, which consists of Twitter conversations for nine different newsworthy events. The distribution of the dataset is shown in Table 2. The basic structure of conversation samples is shown in Figure 3. Each conversation sample has a source-tweet and a set of reactions along time, where reactions express their opinions towards the claim contained in the source-tweet.

As shown in Table 2, this dataset exhibits a severe event-wise and class-wise unbalanced nature. For example, the Charlie Hebdo event is dominant over all other events present in the dataset in terms of number of samples causing event-wise unbalance. In general, the number of non-rumor class samples are way more than the number of rumor class samples, which is a class-wise imbalance in the dataset.

Table 2. The PHEME dataset with nine events.

Event	Rumors	Non-Rumors	Total
Charlie Hebdo	458	1621	2079
Ferguson	284	859	1143
Germanwings crash	238	231	469
Ottawa shooting	470	420	890
Sydney siege	522	699	1221
Gurlitt	61	77	138
Putin missing	126	112	238
Prince Toronto	229	4	233
Ebola Essien	14	0	14
Total	2402	4023	6425

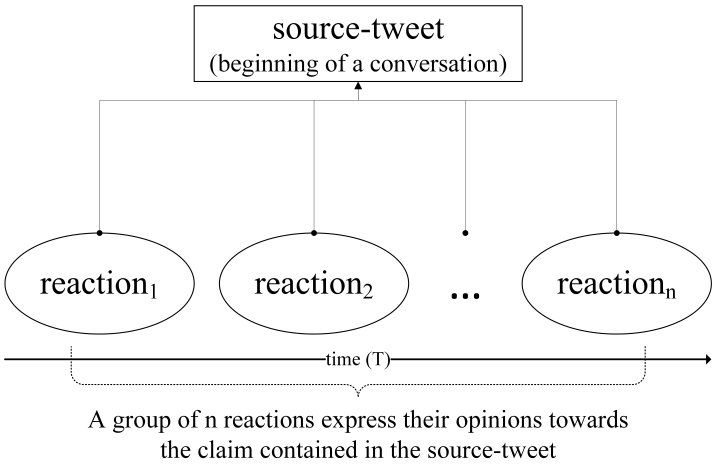

Figure 3. Structure of a Twitter conversation sample.

In our analysis, we removed events Prince Toronto and Ebola Essien as they have extremely unbalanced proportions of rumors and non-rumors, and trimmed down the dataset to seven events. For example, the Ebola Essien event has zero number of non-rumor class samples. The basic statistics of the PHEME dataset with seven events are shown in Table 3. Overall, the PHEME seven events dataset has 6178 data samples, in which non-rumor class samples are almost double the number of rumor class samples.

Table 3. Distribution of the PHEME dataset with seven events.

Event	Rumors	Non-Rumors	Total
Charlie Hebdo	458 (22.03%)	1621 (77.97%)	2079
Ferguson	284 (24.85%)	859 (75.15%)	1143
Germanwings crash	238 (50.75%)	231 (49.25%)	469
Gurlitt	61 (44.20%)	77 (55.80%)	138
Ottawa shooting	470 (52.81%)	420 (47.19%)	890
Putin missing	126 (52.94%)	112 (47.06%)	238
Sydney Siege	522 (42.75%)	699 (57.25%)	1221
Total	2159 (34.95%)	4019 (65.05%)	6178

5.2. Generation of Time-Series Data

In this paper, we explore the temporal features of Twitter data for timely detection of rumors in social media. Specifically, we generated time-series data by transforming Twitter conversations, where each conversation contains a list of tweets, into times-series vectors that contain reaction counts as features, and fed them as input to deep learning models. We transformed each of the Twitter conversation sample present in the PHEME seven events dataset into time-series vector for each time interval T, where $T = \{2, 5, 10, 30, 60\}$ min. After successful transformation of all conversations into time-series data, each vector represents one whole conversation and each of its values are the total reaction counts with respect to T.

Denote $E = \{e_i\}$ the set that contains data of seven events present in the dataset, then for each event data e_i, c_{ij} is a conversation sample related to that event. As the dataset has conversations separated by event, we iterated over all the events one-by-one. In each iteration, for every conversation sample present in them, we extracted timestamp of its source-tweet *timeSource* (starting point of the conversation) and its $timeReactions = \{tr_1, tr_2, \cdots, tr_n\}$, which is a set of timestamps of all the reactions corresponding to that source-tweet. For a conversation sample, its length $N(c)$ is determined by,

$$N(c) = \lceil \frac{\max(timeReactions) - timeSource}{T} \rceil \tag{6}$$

Assume c represents a conversation sample, if $(a, b]$ is the time interval limit for *k-th* interval, where $k = 1, 2, \cdots, N(c)$ then the total reactions count for that time interval is given by,

$$count_k = \mathbf{card}(Q) \tag{7}$$

where $Q \subset timeReactions$ and $Q = \{x \mid x > a \wedge x \leq b\}$, here x is the timestamp of a reaction (tweet) and cardinality is the measure of the size of set Q, and the transformed vector representation is as follows:

$$V(c) = [count_k \quad count_{k+1} \quad \cdots \quad count_N] \tag{8}$$

The final vector representation of all conversation samples for each event is given by,

$$e_i = \begin{bmatrix} V(c_1) \\ V(c_2) \\ \vdots \\ V(c_n) \end{bmatrix} \tag{9}$$

The flow chart of transforming Twitter conversations into time-series vectors for all combinations of E and T is given in Figure 4.

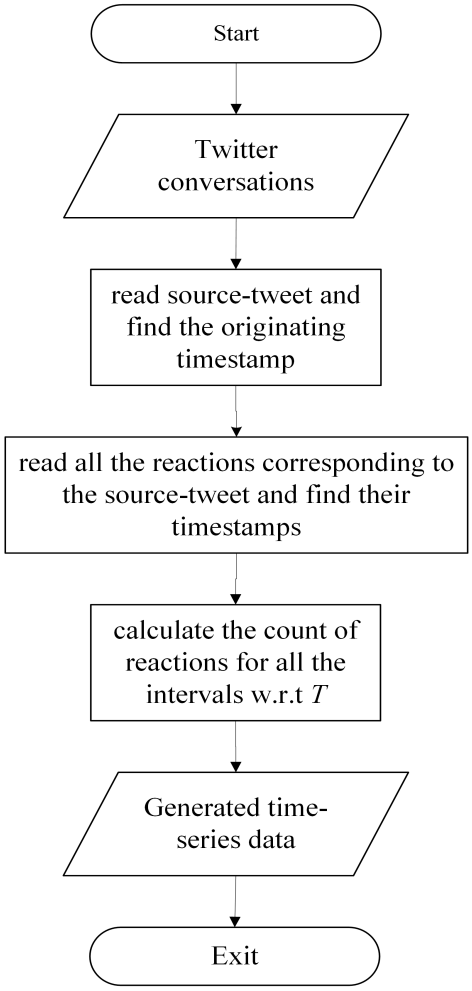

Figure 4. The flow chart for transforming Twitter conversations into time-series vectors.

5.3. Data Pre-Processing

The second step in our data preparation is to reduce the data sparsity of the time-series data since the vector length of all data samples are decided by the longest conversational sample with respect to T. To tackle this problem, we applied Sklearn's dimensionality reduction method called `TruncatedSVD` [36]. Next, we normalized the time-series data using sklearn's `MinMaxScaler` [36], and removed duplicate data samples that are having the same features with different ground truth values.

Finally, we calculated class weights by using sklearn's `class_weight` [36] library with *balanced* scheme since the PHEME dataset exhibits unbalance class nature. Class weights are used in weighting loss functions during the training process, which means higher weight is given to minority classes and lower weight to majority classes. The class weights are computed using the equation below (10).

$$class\ weights = \frac{n\_samples}{(n\_classes \times bincount(y))} \tag{10}$$

where y represents the actual class labels per sample, $n\_classes$ is the count of unique class label values existing in the dataset, $n\_samples$ is the number of data samples, and $bincount(y)$ counts number of occurrences of each value in y of non-negative integers.

6. Results and Discussions

6.1. Evaluation Metrics

We used F1-score, which is the weighted average of Precision and Recall scores as the ensemble model's evaluation metric. We considered F1-score metric with micro and macro averaging schemes for evaluating the performances of the ensemble classification models. In general, we calculate F1-score by using Equation (11).

$$F1 = 2 \times \frac{precision \times recall}{precision + recall} \tag{11}$$

where precision and recall scores tell the strength of a classifier.

In the macro averaging scheme, F1-score is calculated using Equation (13). The macro F1-score uses precision and recall scores for each class label, and finds their unweighted mean. In the micro averaging scheme, F1-score is determined using Equation (15), and the micro F1-score uses global metrics that means precision and recall scores are calculated by counting all the true positives (TP), false positives (FP), and false negatives (FN) across all classes.

$$\begin{aligned} P_{macro} &= \frac{\sum_{i=1}^{n} p_i}{n} \\ R_{macro} &= \frac{\sum_{i=1}^{n} r_i}{n} \end{aligned} \tag{12}$$

$$F1_{macro} = 2 \times \frac{P_{macro} \times R_{macro}}{P_{macro} + R_{macro}} \tag{13}$$

$$\begin{aligned} P_{micro} &= \frac{\sum_{i=1}^{n} TP_i}{\sum_{i=1}^{n} TP_i + FP_i} \\ R_{micro} &= \frac{\sum_{i=1}^{n} TP_i}{\sum_{i=1}^{n} TP_i + FN_i} \end{aligned} \tag{14}$$

$$F1_{micro} = 2 \times \frac{P_{micro} \times R_{micro}}{P_{micro} + R_{micro}} \tag{15}$$

In the above equations, P and R represent precision and recall values for a given averaging scheme (macro or micro), i represents a class label, p_i and r_i are the precision and recall scores for ith class label. TP_i, FP_i, and FN_i are the true positives, false positives, and false negatives respectively for ith class label. n is the total number of classes.

6.2. Experimental Results

In Table 4, we compared our current work's best micro-averaged scores of Precision, Recall, and F1 with our previous works' best micro-averaged results. Clearly, we improved the rumor classification performance by a decent margin with our proposed ensemble based deep learning model in terms of micro-F1. The improvements over Kotteti et al., 2018 [8] and Kotteti et al., 2019 [33] are 12.5% and 7.9%, respectively. The rest of this section discusses the influence of hyperparameters such as batch input size and learning rate on the classification model's performance.

Table 4. Comparison of current work to our previous works.

Metric	Previous Work		Current Work
	Kotteti et al., 2018 [8]	**Kotteti et al., 2019 [33]**	
Micro-Precision	0.949	0.564	0.643
Micro-Recall	0.374	0.564	0.643
Micro-F1	0.518	0.564	0.643

6.2.1. Fixed Batch Input Size

The testing results when the batch input size is fixed are shown in Tables 5 and 6. These testing results are the mean micro and macro averaged F1 scores of all events that are obtained using leave-one-event-out cross-validation along T by varying learning rate.

Table 5. Mean micro averaged F1 testing results of all events that are obtained using leave-one-event-out cross-validation along T by varying learning rate.

Time Interval	Learning Rate	Micro-F1		
		I-1	I-2	I-3
2 min	5×10^{-6}	0.53986	0.52801	0.52835
	1×10^{-5}	0.55231	0.53131	0.53537
	1.5×10^{-5}	0.56656	0.53399	0.50439
5 min	5×10^{-6}	0.43673	0.43128	0.3936
	1×10^{-5}	0.43809	0.4199	0.39489
	1.5×10^{-5}	0.44764	0.41844	0.40408
10 min	5×10^{-6}	0.43347	0.45515	0.46869
	1×10^{-5}	0.43594	0.4086	0.41396
	1.5×10^{-5}	0.42631	0.40358	0.41814
30 min	5×10^{-6}	0.55092	0.43766	0.5524
	1×10^{-5}	0.54492	0.42296	0.53995
	1.5×10^{-5}	0.53428	0.45717	0.5394
60 min	5×10^{-6}	0.55717	0.58966	0.6116
	1×10^{-5}	0.61769	0.56448	0.62146
	1.5×10^{-5}	0.619	0.55943	0.59565

Table 6. Mean macro averaged F1 testing results of all events that are obtained using leave-one-event-out cross-validation along T by varying learning rate.

Time Interval	Learning Rate	Macro-F1		
		I-1	I-2	I-3
2 min	5×10^{-6}	0.44878	0.38891	0.37119
	1×10^{-5}	0.46329	0.39504	0.38406
	1.5×10^{-5}	0.49849	0.38397	0.39108
5 min	5×10^{-6}	0.41544	0.35804	0.29844
	1×10^{-5}	0.42368	0.35859	0.3125
	1.5×10^{-5}	0.42362	0.371	0.34597
10 min	5×10^{-6}	0.34527	0.33345	0.33153
	1×10^{-5}	0.35471	0.31419	0.31294
	1.5×10^{-5}	0.34021	0.32128	0.32075
30 min	5×10^{-6}	0.37669	0.32084	0.34954
	1×10^{-5}	0.38528	0.31908	0.36389
	1.5×10^{-5}	0.38588	0.31528	0.38077
60 min	5×10^{-6}	0.42367	0.39506	0.45095
	1×10^{-5}	0.48757	0.39201	0.45083
	1.5×10^{-5}	0.48163	0.38864	0.43825

Micro Scores

From Table 5, for $T = 2$ and 5 min, the micro-F1 scores of the ensemble Implementation-1 (I-1) are better than that of the ensemble Implementation-2 (I-2) and Implementation-3 (I-3) across the chosen learning rates. This is due to the fact that it has more ensemble diversity compared with other ensembles, i.e., the presence of base learners designed using Bi-directional RNNs and a model with hybrid architecture that contains a pair of LSTM and GRU layers. In these time intervals, the best scores for the ensemble I-1 are obtained for learning rate 1.5×10^{-5}.

When $T = 10$ min and $T = 30$ min, the micro scores performances are mixed. For instance, the ensemble I-1 outperformed ensembles I-2 and I-3 for learning rates 1×10^{-5} and 1.5×10^{-5} when $T = 10$ min. For $T = 30$ min, the ensemble I-3 achieved maximum micro-F1 score for learning rates 5×10^{-6} and 1.5×10^{-5}.

For $T = 60$ min, the ensemble I-3 outperformed other ensembles in terms of maximum micro-F1 score for learning rates 5×10^{-6} and 1×10^{-5}. It is this time interval where all ensembles obtained their maximum micro-F1 scores across T for all chosen time intervals. The overall best micro-F1 score of 62.1% is achieved by the ensemble I-3 for learning rate 1×10^{-5}. In this time interval, ensembles I-1 and I-3 are better than that of the ensemble I-2. Again, this is due to more diversity of ensemble I-1 and the base learners in ensemble I-3 with LSTM have better representational power than GRU in ensemble I-2.

Macro Scores

From Table 6, for $T = 2, 5, 10$ and 30 min, the macro-F1 scores of the ensemble I-1 are better than that of the ensembles I-2 and I-3 across the chosen learning rates. Again, this is due to the presence of more diversified base learners in ensemble I-1 that helped to surpass other ensembles. It is also noticed that when $T = 10$ and 30 min, the performance of the ensemble I-1 drops down across the learning rates compared to $T = 2$ and 5 min. This is because for longer time intervals, the lengths of time-series data sequences become shorter thus may overlook small propagation patters presented in the time-series data.

For $T = 60$ min, the ensemble I-1 outperformed others in terms of best macro-F1 score for learning rates 1×10^{-5} and 1.5×10^{-5}. When the learning rate is 5×10^{-6}, the ensemble I-3 surpasses other ensembles. Moreover, in this time interval, for ensembles I-1 and I-2, the results are almost on par with the results that they achieved when $T = 2$ min. In this time interval, the ensemble I-3 achieved its overall best performance across T. The overall best macro-F1 score is obtained by the ensemble I-1 when $T = 2$ min and learning rate of 1.5×10^{-5}.

General Observations

Furthermore, ensembles I-1, I-2, and I-3 better performed in terms of both micro-F1 and macro-F1 scores when $T = 60$ min over other time intervals w.r.t the chosen learning rates. The only exception is that the ensemble I-1 performed well in terms of macro-F1 score when $T = 2$ min over other time intervals w.r.t the chosen learning rates. In general, both the results are showing us the fact that the performances of ensembles are better when T is either low (2 min) or high (60 min). This provides a guidance for us to select time interval based on the requirement. For example, if early detection is important, we can pick a low time interval value. In the case of effective prediction, we can go for a higher time interval value.

It is also noted that the 10 min time interval caused most of the ensemble implementations, particularly, the ensemble I-1 to achieve low performance in both micro and macro scores. This may be due to the propagation patterns extracted using this time interval value do not have necessary variations, such that it is harder for classification. Another interesting observation is that ensemble I-3 performs poorly with a 5 min time interval in both micro and macro scores. In this case, using a

5 min time interval caused high data sparsity, which in turn caused LSTM-based ensemble I-3 to perform poorly.

6.2.2. Fixed Learning Rate

In the case of a fixed learning rate, the testing results are shown in Tables 7 and 8. These testing results are the mean micro and macro averaged F1 scores of all events that are obtained using leave-one-event-out cross-validation along T by varying batch input size.

Table 7. Mean micro averaged F1 testing results of all events that are obtained using leave-one-event-out cross-validation along T by varying batch input size.

Time Interval	Batch Input Size	Micro-F1		
		I-1	I-2	I-3
2 min	16	0.51013	0.48588	0.50378
	32	0.55231	0.53131	0.53537
	64	0.54089	0.51473	0.52323
5 min	16	0.48062	0.4534	0.42757
	32	0.43809	0.4199	0.39489
	64	0.44371	0.46473	0.41045
10 min	16	0.44006	0.43332	0.48341
	32	0.43594	0.4086	0.41396
	64	0.43322	0.42206	0.42
30 min	16	0.51484	0.4542	0.50053
	32	0.54492	0.42296	0.53995
	64	0.55006	0.45109	0.54926
60 min	16	0.5789	0.5619	0.46469
	32	0.61769	0.56448	0.62146
	64	0.57902	0.58571	0.64331

Table 8. Mean macro averaged F1 testing results of all events that are obtained using leave-one-event-out cross-validation along T by varying batch input size.

Time Interval	Batch Input Size	Macro-F1		
		I-1	I-2	I-3
2 min	16	0.44335	0.39693	0.38611
	32	0.46329	0.39504	0.38406
	64	0.43664	0.3692	0.36674
5 min	16	0.4367	0.37274	0.35665
	32	0.42368	0.35859	0.3125
	64	0.38173	0.32	0.32925
10 min	16	0.34352	0.32805	0.35568
	32	0.35471	0.31419	0.31294
	64	0.35004	0.31881	0.32688
30 min	16	0.37424	0.30853	0.3749
	32	0.38528	0.31908	0.36389
	64	0.38205	0.30122	0.34899
60 min	16	0.4266	0.37702	0.3486
	32	0.48757	0.39201	0.45083
	64	0.39252	0.38015	0.47661

Micro Scores

From Table 7, for $T = 2$ and 30 min, the micro-F1 scores of ensembles I-1 and I-3 are very similar and better than that of the ensemble I-2. This is due to the presence of LSTM layers in both ensembles I-1 and I-3, where in ensemble I-2, there is no base learner with a LSTM layer. In these intervals, w.r.t the chosen batch input sizes, ensemble I-1 achieved the best performance.

When $T = 5$ min, the ensemble I-1 outperformed other ensembles for batch input sizes 16 and 32. In this time interval, the ensemble I-2 performed better than that of other ensembles for batch size 64. It is this time interval, where the ensemble I-3 achieved its least micro-F1 scores across all the batch input sizes and T, which is the same when batch input size is fixed under micro-averaging scheme. For $T = 10$ min, the ensemble I-1 obtained the best micro-F1 scores for batch input sizes 32 and 64, and the ensemble I-3 achieved better micro-F1 score over other ensembles for batch input size 16. In this time interval, the ensembles I-1 and I-2 obtained their least micro-F1 scores across all the batch input sizes and T.

When $T = 60$ min, the ensemble I-3 outperformed other ensembles for batch input sizes 32 and 64, and the ensemble I-1 performed better for batch input size of 16. It is this time interval, where all ensembles obtained their maximum micro-F1 scores. The overall best micro-F1 score of 64.3% is achieved by the ensemble I-3 for batch input size of 64. In this case, the higher time interval helped the ensembles to surpass their lower time interval micro-F1 scores for almost all of the combinations of batch input size and T. Once again, the results show that LSTM-backed ensemble I-3 outplayed other ensembles given the advantages of LSTM such as its good gating mechanism and ability to learn long-term dependencies.

Macro Scores

From Table 8, for $T = 2$ min, the ensemble I-1 achieved better macro-F1 scores than that of ensembles I-2 and I-3 across all the batch input sizes. In this time interval, the ensemble I-2 obtained its maximum macro-F1 score. When $T = 5$ min, the ensemble I-1 outperformed other ensembles in terms of macro-F1 score. Lower time intervals have longer time-series sequences that can better represent variations in propagation patterns of rumors and non-rumors than for higher time interval values. However, lower time intervals may have more data sparsity.

For $T = 10$ and 30 min, the ensemble I-1 achieved better performance than that of other ensembles for batch input sizes 32 and 64. However, its performance is significantly dropped compared to lower time interval values, and the ensemble I-3 obtained better performance for batch input size 16. The ensemble I-2 became weak when $T = 30$ min, and ensembles I-1 and I-3 start to show some improvement in their performances compared to $T = 10$ min.

In the time interval $T = 60$, the ensemble I-1 better performed over other ensembles for batch input sizes 16 and 32, and the ensemble I-3 obtained the best macro-F1 score for batch input size 64. In this time interval, the ensembles I-1 and I-3 obtained their overall maximum macro-F1 scores (i.e., 48.7% and 47.6% respectively) across T. Overall, the ensembles support extreme time intervals such as $T = 2$ min and $T = 60$ min in order to achieve good performance.

General Observations

In case of micro-F1 score, the ensembles I-1, I-2, and I-3 obtained their best micro-F1 scores for $T = 60$ min w.r.t the chosen batch input sizes. The only exception is where the micro-F1 score of the ensemble I-3 is lower than its own micro-F1 scores when $T = 2, 10$ and 30 min when batch input size set to 16. This means that $T = 60$ min is appropriate for effective detection of rumors. In case of macro-F1 score, the best performances of the ensembles I-1, I-2, and I-3 are varied for each batch input size across T, which means based on the need we can choose an ensemble model and select an appropriate time interval.

As discussed earlier, we have seen the same behavior for the 10 min time interval, which caused most of the ensemble implementations to perform poorly for both micro and macro averaging schemes. In addition to that, ensemble I-3 again showed low performance in 5 min time intervals under both averaging schemes.

By observing the above results, varying the hyperparameters batch input size and learning rate resulted in producing similar kinds of behavior in the ensembles. In general, when micro-averaging is used, both hyperparameter variations supported higher time interval values for better performance. In the case where a macro-averaging scheme is employed, time intervals 2 and 60 min helped ensembles I-1 and I-2 to perform well. However, ensemble I-3 still achieved better performance when $T = 60$ min. As all ensembles are performing well with 60 min time interval, it is a good choice to achieve decent performance regardless of variations in chosen batch input sizes and learning rates. For $T = 60$, the generated time-series data will have lesser data sparsity than that of other values of T that make the feature space short for the conversation samples. This may be the reason for all ensembles to perform better at higher time intervals, especially ensembles with base learners designed using LSTM layers.

Another key observation is that, for all ensembles, 2 and 60 min time intervals are shown to have good performance. However, there is no sweet spot for the ensembles for other values of T. This observation is critical in applying the proposed model depending on the goal. For instance, if early detection is needed we can pick a small time interval value such as $T = 2$ min by sacrificing a little amount of prediction performance. In case of effective prediction is important, we can set time interval to a higher value, for example, $T = 60$ min.

6.3. Discussions

As the PHEME dataset is imbalanced (i.e., non-rumor samples almost double the number of rumor samples), adding more rumor samples to the dataset will help in improving its class balance, and may help classification models to perform better classification. When compared to [33], we noticed that increase in maximum micro and macro averaged F1 scores with addition of two extra events (Gurlitt and Putin Missing events) to the dataset. In case of fixed batch input size, the improvement is 5.7% and 0.4% for micro and macro averaging schemes, respectively. When the learning rate is constant, the improvement is 7.9% for micro averaging scheme. However, the maximum macro F1 score is dropped by 0.7%. Moreover, even though the Gurlitt and Putin missing events are included in the seven events PHEME dataset, only the Putin missing event contributed in adding slightly a greater number of rumor samples to the dataset than Gurlitt event, which is also a supporter of the non-rumor group.

In addition to this, our data pre-processing method combined with the proposed model helped in improving our previous best score in [33] and achieved 64.3% micro F1 score, which is almost 8% improvement. The performance improvement may seem small, but it is non-trivial to gain huge performances using this dataset, for instance, in [37], extensive feature engineering was conducted for the rumor detection problem on social media using the PHEME dataset with five events. The authors focused on extracting complex features such as content-based and social features, and their best F1 scores are 0.606 and 0.339 for content-based and social features respectively, and when both feature sets are jointly used, the F1 score reached up to 0.607, which is 0.1% improvement. Again, extensive feature engineering needs long time to be completed as some of the features may not be readily available, having complex feature sets challenge hardware resources, which also increases computational complexity that directly impacts training times of classification models. Nevertheless, given the condition that information spreads rapidly on social media, time-taking labor-intensive feature engineering may not be appropriate.

As our work is solely based on the temporal property (i.e., the tweet creation timestamp value) of tweets we believe using other higher-level features, for example, content-based and context-based features may help in building a more effective classification model. Since our work is intended for the early detection of rumors, this can be used as a primary model and coupled with other models that give

more accurate predictions, for building a robust classification model overall. Furthermore, the proposed model's efficiency should be validated with other datasets that are similar to the PHEME dataset. As the PHEME dataset is imbalanced in its nature a balanced dataset may help in improving the efficiency of the proposed model. Finally, the time interval step size can be reduced (i.e., 1 min or half-minute) and/or increased (i.e., 90 min or 2 h) for more deep analysis on the propagation patterns of rumors and non-rumors.

7. Related Work

Rumor detection on social media is an existing problem in the literature. Many researchers have experimented to find a good solution to this problem. In [38], the authors developed a two-layer network to model the interaction between epidemic spreading and information diffusion. Their results showed that knowledge diffusion can eradicate both rumors and epidemics, where the penetration intensity of knowledge into rumor plays a crucial role. It increases the thresholds for rumors and epidemics to break out. According to [39], a complex network can be modeled as a graph, which usually consists of nodes and edges. Identifying the most influential node in a complex network has real world applicability, for instance, rumor spreading in social networks. The authors presented a survey on the identification of influential spreaders in complex networks by analyzing and comparing major variations of k-shell based methods along with representative network topology based hybrid techniques. The coreness of the nodes is considered in a typical k-shell method by dividing the network into shells.

In [40], the authors proposed a distributed computing approach to calculate the network centrality value for each user in a social network. They used the MapReduce approach in the Hadoop platform, which allows better computational performance than that of conventional implementation. Their results showed that with the distributed approach they improved the calculation performance of degree centrality, closeness centrality, and eigenvalue centrality on average by significant margins over the conventional approach. The authors of [41], proposed a hybrid recommendation model to improve recommendation accuracy. Their model is based on users' ratings, reviews, and social data. By conducting a variety of experiments their results showed that their proposed model helps improve the recommendation accuracy.

In [42], authors have explored user-specific features along with content characteristics of social media messages and proposed an information propagation model based on heterogeneous user representation to observe distinctions in the propagation patterns of rumors and credible messages and using it to differentiate them, and their study identifies that rumors are more likely to spread among certain user groups. To predict a document in a social media stream to be a future rumor and stop its spread Qin et al. [43] used content-based features along with novelty-based features and pseudo feedback. In [7], a sentiment dictionary and a dynamic time series algorithm based Gated Recurrent Unit model is proposed, that identifies fine-grained human emotional expressions of microblog events and the time distribution of social events to detect rumor events.

By treating microblog users' behaviors as hidden clues to detect possible rumormongers or rumor posts, Liang et al. [44] proposed a user behavior-based rumor identification schemes, which focuses on applying traditional user behavior-based features as well as authors' proposed new features that are extracted from users' behaviors to rumor identification task and concluded that rumor detection based on mass behaviors is better than detection based on microblogs' inherent features. In [10], temporal, structural, and linguistic features of social media rumors were explored for rumor classification task and using those features together helped in identifying rumors more accurately. Wu et al. [45] proposed a graph-kernel based hybrid SVM classifier that can capture high-order (message) propagation patterns as well as semantic features, for example, the topics of the original message for automatically detecting false rumors on Sina Weibo.

As discussed above, most of the works focus on medium to heavy weight feature extraction processes, which makes them slow in identifying false information on social media since the fast-paced

environment of social media allows a very little amount of time to analyze a piece of information before it propagates all over the network. Our proposed data pre-processing method and ensemble model are capable for this challenge because of the nature of our generated time-series data, and simplicity of classification models' architectures that are part of the ensemble model, and feature extraction process is almost near real-time since our features are creation timestamps of Twitter tweets, which can be extracted and processed without any time delay.

8. Conclusions

We proposed a data pre-processing method and ensemble model for fast detection of rumors on social media. Our data pre-processing method transforms Twitter conversations into time-series vectors using the tweet creation timestamps, which can be extracted and processed without any wait time, which is a key requirement for the defined problem, and generated time-series data that are of the pure numeric type, which simplifies feature set complexity that in turn helps in reducing the computational complexity of classification models during their training process, and our ensemble model contains several classification models designed using deep learning techniques that have simplistic yet effective architectures. By combining data pre-processing with the ensemble model, we improved the classification performance in terms of micro F1 score compared to the baselines (Kotteti et al., 2018 [8] and Kotteti et al., 2019 [33]). The performance improvements are 12.5% and 7.9%, respectively.

Author Contributions: Conceptualization, C.M.M.K. and X.D.; methodology, C.M.M.K.; software, C.M.M.K. and X.D.; validation, C.M.M.K., X.D. and L.Q.; formal analysis, L.Q.; investigation, L.Q.; resources, X.D. and L.Q.; writing—original draft preparation, C.M.M.K.; writing—review and editing, X.D. and L.Q.; supervision, L.Q.; funding acquisition, L.Q. All authors have read and agreed to the published version of the manuscript.

Funding: This research work is supported by the U.S. Office of the Under Secretary of Defense for Research and Engineering (OUSD(R&E)) under agreement number FA8750-15-2-0119.

Acknowledgments: The U.S. Government is authorized to reproduce and distribute reprints for governmental purposes notwithstanding any copyright notation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the Office of the Under Secretary of Defense for Research and Engineering (OUSD(R&E)) or the U.S. Government.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Zubiaga, A.; Liakata, M.; Procter, R.; Bontcheva, K.; Tolmie, P. Towards detecting rumours in social media. In Proceedings of the Workshops at the Twenty-Ninth AAAI Conference on Artificial Intelligence, Austin, TX, USA, 25–30 January 2015.
2. Shu, K.; Sliva, A.; Wang, S.; Tang, J.; Liu, H. Fake news detection on social media: A data mining perspective. *ACM SIGKDD Explor. Newsl.* **2017**, *19*, 22–36. [CrossRef]
3. Declerck, T.; Osenova, P.; Georgiev, G.; Lendvai, P. Ontological modelling of rumors. In *Workshop on Social Media and the Web of Linked Data*; Springer: Berlin/Heidelberg, Germany, 2015; pp. 3–17.
4. Qazvinian, V.; Rosengren, E.; Radev, D.R.; Mei, Q. Rumor has it: Identifying misinformation in microblogs. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*; Association for Computational Linguistics: Edinburgh, Scotland, UK, 2011; pp. 1589–1599.
5. Kshetri, N.; Voas, J. The economics of "fake news". *IT Prof.* **2017**, *19*, 8–12. [CrossRef]
6. Reshi, J.A.; Ali, R. Rumor proliferation and detection in Social Media: A Review. In Proceedings of the 2019 5th International Conference on Advanced Computing & Communication Systems (ICACCS), Coimbatore, India, 15–16 March 2019; pp. 1156–1160.
7. Wang, Z.; Guo, Y.; Wang, J.; Li, Z.; Tang, M. Rumor Events Detection from Chinese Microblogs via Sentiments Enhancement. *IEEE Access* **2019**, *7*, 103000–103018. [CrossRef]

8. Kotteti, C.M.M.; Dong, X.; Qian, L. Multiple Time-Series Data Analysis for Rumor Detection on Social Media. In Proceedings of the 2018 IEEE International Conference on Big Data (Big Data), Seattle, WA, USA, 10–13 December 2018; pp. 4413–4419.
9. Zhao, J.; Cao, N.; Wen, Z.; Song, Y.; Lin, Y.R.; Collins, C. # FluxFlow: Visual analysis of anomalous information spreading on social media. *IEEE Trans. Vis. Comput. Graph.* **2014**, *20*, 1773–1782. [PubMed]
10. Kwon, S.; Cha, M.; Jung, K.; Chen, W.; Wang, Y. Prominent features of rumor propagation in online social media. In Proceedings of the 2013 IEEE 13th International Conference on Data Mining, Dallas, TX, USA, 7–10 December 2013; pp. 1103–1108.
11. Shibutani, T. *Improvised News: A Sociological Study of Rumor*; Ardent Media: London, UK, 1966.
12. Zhou, Z.H. Ensemble learning. *Encycl. Biom.* **2015**, 411–416.
13. Freund, Y.; Schapire, R.E. A decision-theoretic generalization of on-line learning and an application to boosting. *J. Comput. Syst. Sci.* **1997**, *55*, 119–139. [CrossRef]
14. Breiman, L. Bagging predictors. *Mach. Learn.* **1996**, *24*, 123–140. [CrossRef]
15. Wolpert, D.H. Stacked generalization. *Neural Netw.* **1992**, *5*, 241–259. [CrossRef]
16. Breiman, L. Random forests. *Mach. Learn.* **2001**, *45*, 5–32. [CrossRef]
17. Gaikwad, D.; Thool, R.C. Intrusion detection system using bagging ensemble method of machine learning. In Proceedings of the 2015 International Conference on Computing Communication Control and Automation, Pune, India, 26–27 February 2015; pp. 291–295.
18. Tuysuzoglu, G.; Moarref, N.; Yaslan, Y. Ensemble based classifiers using dictionary learning. In Proceedings of the 2016 International Conference on Systems, Signals and Image Processing (IWSSIP), Bratislava, Slovakia, 23–25 May 2016; pp. 1–4.
19. Li, H.; Wang, J.; Gao, T.; Lu, Y.; Su, Z. Accurate prediction of the optical absorption energies by neural network ensemble approach. In Proceedings of the 2010 Fifth International Conference on Frontier of Computer Science and Technology, Changchun, China, 18–22 August 2010; pp. 503–507.
20. Linghu, B.; Sun, B. Constructing effective SVM ensembles for image classification. In Proceedings of the 2010 Third International Symposium on Knowledge Acquisition and Modeling, Wuhan, China, 20–21 October 2010; pp. 80–83.
21. Zeng, X.D.; Chao, S.; Wong, F. Optimization of bagging classifiers based on SBCB algorithm. In Proceedings of the 2010 International Conference on Machine Learning and Cybernetics, Qingdao, China, 11–14 July 2010; Volume 1, pp. 262–267.
22. Chen, Y.; Wang, Y.; Gu, Y.; He, X.; Ghamisi, P.; Jia, X. Deep Learning Ensemble for Hyperspectral Image Classification. *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens.* **2019**, *12*, 1882–1897. [CrossRef]
23. Islam, M.M.; Yao, X.; Nirjon, S.S.; Islam, M.A.; Murase, K. Bagging and boosting negatively correlated neural networks. *IEEE Trans. Syst. Man Cybern. Part B (Cybern.)* **2008**, *38*, 771–784. [CrossRef] [PubMed]
24. Shi, L.; Xi, L.; Ma, X.; Hu, X. Bagging of artificial neural networks for bankruptcy prediction. In Proceedings of the 2009 International Conference on Information and Financial Engineering, Singapore, 17–20 April 2009; pp. 154–156.
25. Fakhruzi, I. An artificial neural network with bagging to address imbalance datasets on clinical prediction. In Proceedings of the 2018 International Conference on Information and Communications Technology (ICOIACT), Yogyakarta, Indonesia, 6–7 March 2018; pp. 895–898.
26. Chollet, F. *Deep Learning with Python*, 1st ed.; Manning Publications Co.: Greenwich, CT, USA, 2017.
27. Cho, K.; Van Merriënboer, B.; Gulcehre, C.; Bahdanau, D.; Bougares, F.; Schwenk, H.; Bengio, Y. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *arXiv* **2014**, arXiv:1406.1078.
28. The Keras Special Interest Group. Keras. 2015. Available online: https://keras.io (accessed on 8 January 2019).
29. Hochreiter, S.; Schmidhuber, J. Long short-term memory. *Neural Comput.* **1997**, *9*, 1735–1780. [CrossRef]
30. Chung, J.; Gulcehre, C.; Cho, K.; Bengio, Y. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv* **2014**, arXiv:1412.3555.
31. Schuster, M.; Paliwal, K.K. Bidirectional recurrent neural networks. *IEEE Trans. Signal Process.* **1997**, *45*, 2673–2681. [CrossRef]

32. Graves, A.; Jaitly, N.; Mohamed, A. Hybrid speech recognition with deep bidirectional LSTM. In Proceedings of the 2013 IEEE Workshop on Automatic Speech Recognition and Understanding, Olomouc, Czech Republic, 8–12 December 2013; pp. 273–278.
33. Kotteti, C.M.M.; Dong, X.; Qian, L. Rumor Detection on Time-Series of Tweets via Deep Learning. In Proceedings of the 2019 IEEE Military Communications Conference (MILCOM 2019), Norfolk, VA, USA, 12–14 November 2019; pp. 1–7.
34. Heaton, J. *Introduction to Neural Networks with Java*; Heaton Research, Inc.: Chesterfield, UK, 2008.
35. Kochkina, E.; Liakata, M.; Zubiaga, A. All-in-one: Multi-task learning for rumour verification. *arXiv* **2018**, arXiv:1806.03713.
36. Pedregosa, F.; Varoquaux, G.; Gramfort, A.; Michel, V.; Thirion, B.; Grisel, O.; Blondel, M.; Prettenhofer, P.; Weiss, R.; Dubourg, V.; et al. Scikit-learn: Machine Learning in Python. *J. Mach. Learn. Res.* **2011**, *12*, 2825–2830.
37. Zubiaga, A.; Liakata, M.; Procter, R. Learning reporting dynamics during breaking news for rumour detection in social media. *arXiv* **2016**, arXiv:1610.07363.
38. Huang, H.; Chen, Y.; Ma, Y. Modeling the competitive diffusions of rumor and knowledge and the impacts on epidemic spreading. *Appl. Math. Comput.* **2020**, *388*, 125536. [CrossRef]
39. Maji, G.; Mandal, S.; Sen, S. A systematic survey on influential spreaders identification in complex networks with a focus on K-shell based techniques. *Expert Syst. Appl.* **2020**, *161*, 113681. [CrossRef]
40. Kumar Behera, R.; Kumar Rath, S.; Misra, S.; Damaševičius, R.; Maskeliūnas, R. Distributed centrality analysis of social network data using MapReduce. *Algorithms* **2019**, *12*, 161. [CrossRef]
41. Ji, Z.; Pi, H.; Wei, W.; Xiong, B.; Woźniak, M.; Damasevicius, R. Recommendation based on review texts and social communities: A hybrid model. *IEEE Access* **2019**, *7*, 40416–40427. [CrossRef]
42. Liu, Y.; Xu, S. Detecting rumors through modeling information propagation networks in a social media environment. *IEEE Trans. Comput. Soc. Syst.* **2016**, *3*, 46–62. [CrossRef]
43. Qin, Y.; Dominik, W.; Tang, C. Predicting future rumours. *Chin. J. Electron.* **2018**, *27*, 514–520. [CrossRef]
44. Liang, G.; He, W.; Xu, C.; Chen, L.; Zeng, J. Rumor identification in microblogging systems based on users' behavior. *IEEE Trans. Comput. Soc. Syst.* **2015**, *2*, 99–108. [CrossRef]
45. Wu, K.; Yang, S.; Zhu, K.Q. False rumors detection on sina weibo by propagation structures. In Proceedings of the 2015 IEEE 31st International Conference on Data Engineering, Seoul, Korea, 13–17 April 2015; pp. 651–662.