Research Article

International Journal of
Distributed
Sensor Networks

International Journal of Distributed Sensor Networks
2019, Vol. 15(6)
© The Author(s) 2019
DOI: 10.1177/1550147719856492
journals.sagepub.com/home/dsn
SAGE

Real-time event detection using recurrent neural network in social sensors

Van Quan Nguyen, Tien Nguyen Anh and Hyung-Jeong Yang

Abstract
We proposed an approach for temporal event detection using deep learning and multi-embedding on a set of text data from social media. First, a convolutional neural network augmented with multiple word-embedding architectures is used as a text classifier for the pre-processing of the input textual data. Second, an event detection model using a recurrent neural network is employed to learn time series data features by extracting temporal information. Recently, convolutional neural networks have been used in natural language processing problems and have obtained excellent results as performing on available embedding vector. In this article, word-embedding features at the embedding layer are combined and fed to convolutional neural network. The proposed method shows no size limitation, supplementation of more embeddings than standard multichannel based approaches, and obtained similar performance (accuracy score) on some benchmark data sets, especially in an imbalanced data set. For event detection, a long short-term memory network is used as a predictor that learns higher level temporal features so as to predict future values. An error distribution estimation model is built to calculate the anomaly score of observation. Events are detected using a window-based method on the anomaly scores.

Keywords
Social data, neural network, multiple word embedding, event detection, long short-term memory, real-time

Date received: 14 February 2018; accepted: 12 May 2019

Handling Editor: Gianluigi Ferrari

Introduction

Social network services (SNSs) such as Twitter and Facebook have been widely used in recent years, and they have become a potential data source for data mining as social sensor. In fact, social networks have come to dominate the daily activities of people (entertainment, online shopping, distance learning, etc.). People use social networks as a means to express their opinions and emotions about what they are experiencing. Due to the natural real-time characteristic of social data, users can be considered social sensors and posted messages (tweets) can be considered the response signals called social data. Although SNS messages posted by Twitter users are short sentences of less than 140 characters, they contain useful information in many contexts, especially in cases of emergency situations. Therefore, these big data from social media can be used as social sensors to explore vital knowledge.

Although social data are the vital source of data for mining events, opinions, or trends, we need to filter out noise or irrelevant data. Any automatic classification system of social media data to extract useful information is faced with a number of challenges, such as the

Department of Electronics and Computer Engineering, Chonnam National University, Gwangju, South Korea

Corresponding author:
Hyung-Jeong Yang, Department of Electronics and Computer Engineering, Chonnam National University, Gwangju 61186, South Korea.
Email: hjyang@jnu.ac.kr

fact that the sentences are short; it is hard to understand the content; and the messages contain abbreviations, lingo, and spelling mistakes. Due to the requirement of noise filtering, many studies introduce the classification of social media messages. There have been several works on detecting or identifying informative messages in a pre-processing phase. Naïve Bayer is an easy technique for text classification that is commonly used in spam detection,[1] recommendation systems,[2] topic detection,[3] finding trending topics,[4] and summarizing social media blogs.[5] In the social context, the support vector machine (SVM) has been used to solve many problems such as opinion mining for the purpose of reviewing user satisfaction with a product[6] or classifying tweet messages to infer trending topics.[7] The main problem involves extracting the features of the linguistic unit and conducting composition over variable-size sequences.[8–10]

Neural network–based models naturally enable the learning of features that make them more efficient than traditional feature-based methods because of the fact that they have no requirement of complicated feature engineering. Convolutional neural network (CNN)-based methods have achieved impressive results[8,11] in sentence level classification. CNN is based on learning distributed representation that is derived by projecting words to lower dimensions and dense vector space. It encodes the semantic and syntactic features of words.[12]

An event can be generally defined as a real-world occurrence when it requires unfolding over space and time aspects.[13] Social media–based event detection is associated with increasing numbers of messages related to some topics. Therefore, events can be derived from anomaly detection methods. In order to discover temporal features, collected data are processed through two approaches: contents of message[14–18] or features (handcraft generator or learned feature).[19–21] In particular, keyword-based pre-processing is often used to focus on event data. Several studies have attempted to use discrete signals to find high frequency or "burst" features in time series data.[19,20,22–24] He et al.[25] use features associated with power spectrum and periodicity to group keyword signals. A discrete Fourier transformation (DFT)-based method involves finding the peak in the frequency domain, while Weng and Lee[26] use a wavelet transformation. For situations involving disasters, Avvenuti et al.[22] and Nguyen et al.[24] use a burst detection algorithm based on the occurrence of a larger number of events within some time window. Sakaki et al.[27] use a temporal model considering the number of event-related tweets as an exponential distribution. The exponential distribution occurs naturally when describing the lengths of the inter-arrival times in a homogeneous Poisson process.

In this article, we propose an approach that automatically identifies informative messages on social sensors by taking advantage of neural network techniques. It learns text features from embedding vectors. An improved version of the standard CNN on diverse word embedding is introduced in order to overcome the shortcomings of CNN-based methods, which use multichannel word embedding. Standard multiple word embeddings appear as separated channels red, green, and blue (RGB) in image processing. Since the standard approach of diverse word embeddings requires the same dimension, it is not convenient when we use pre-trained word embeddings from different corpora. As we directly work with the noisy environment as the social network, our approach of filtering "non-informative data" to obtain "informative data" for event detection is necessary. In the scope of event detection, there are many traditional methods for this task, but feature-based methods require complicated feature engineering.[19,20,22,27] In this article, we introduce a way to take advantage of both CNN on word embedding and variable sizes of multiple embeddings under the classifier in order to identify informative messages.

In our work, there is no limit to either the types of embeddings or the number of embeddings. The contributions from the improved CNN are as follows: (1) it overcomes the limitation of multichannel embeddings requiring the same dimension. Therefore, it is easy to implement in terms of training models and practical applications due to the availability of pre-trained word embeddings for use. (2) The multi-word embeddings-based approach is investigated in order to explore sentence features. These show an improved performance of the classifier as compared to the baseline methods. (3) It does not require complicated feature engineers, as do traditional machine learning methods.

In order to detect the temporal occurrence of the considered event, we accumulate topics related to informative messages in order to transform them into time series data. The time series data is analyzed by long short-term memory (LSTM)-based[28] event detection approach. An error distribution estimation model is employed to calculate the anomaly score of the observations. A window-based method is used to detect events with the anomaly score. This approach also performs well on many context applications, where balanced data are not always available and real-time disaster event detection is required. We conducted experiments on earthquake data set to obtain the time response.

The rest of the article is organized as follows: the section "Related works" provides a brief discussion of the background and related works, including text classification and event detection in social network data. Section "Proposed approach" focuses on our proposed approach with multiple word embeddings for sentence-level classification and the LSTM recurrent neural network (RNN) for event detection. Then, the experiments on the benchmark data set as well as earthquake signals are conducted in the section "Experimental results" so as to evaluate the performances. We presented the general architecture of the Hadoop-based event detection system as well. Finally, the section "Conclusion" describes the conclusion and future works.

Related works

There have been some attempts to use social network data to detect events such as trends, opinions, or disasters. For example, Sakaki et al.[27] mined Twitter data for the purpose of real-time earthquake detection and to send warnings faster than the Japan Meteorological Agency. This system used a SVM classifier to remove irrelevant tweets, and a probabilistic model was constructed for temporal analysis. The Poisson process is the core of the temporal model, which is used to estimate the time moment as the earthquake happened in real time. Chatfield and Uuf[29] used Twitter data involving the context of the three earthquakes that occurred at the Sumatra coast, Indonesia from 2010 to 2012. Avvenuti et al.[22] developed a system for earthquake data in Italy. It initially considered both tweets and replies from Twitter. In order to filter out irrelevant information, a classification-based sophisticated filter was employed on uniform resource locator (URL), mentions, words, characters, punctuation, and slang/offensive words. For temporal analysis, they created a burst detection method, which observed the number of messages in time windows. The limitations of Sakaki et al.[27] and Avvenuti et al.[22] are that some set of features for input must be predefined. Therefore, the performances of these detection systems depend on how strong features are collected.

CNN has become a popular technology for solving classification tasks. The intuition behind CNNs is that the convolutional layer can automatically learn a better representation of text data, and the fully connected layers finally classify this representation based on a set of labels. It can also be more generic and adaptive to domain and context with the support of embeddings that represent each word as a dense vector. Recently, word embeddings have been exploited for a sentence classification task using CNN. Kim[8] reports a CNN architecture on top of the pre-trained work vector that is static (fixed input) or non-static (tuned during training) for specified tasks: sentiment analysis and question classification. It presents multichannel representation and a variable-size filter. However, it uses two copies of a single version of pre-trained embeddings with initial parameters so as to prevent overfitting. Yin and Schutze[12] show the feeding of multiple word embeddings as separated channels, similar to RGB channels in image processing. This architecture works well on multi-sentence classification. However, it requires additional mutual learning and the embeddings to have the same dimensions; the latter is a limitation on using more than one pre-trained embedding, since pre-trained embeddings can vary widely in dimension.

Chen et al.[30] and Nguyen and Grishman[31] address the problems related to the event detection in sentence levels using a CNN based on word embeddings. While Nguyen and Grishman[31] use a CNN based on multiple channels (word embedding, position, and entity) to identify the event trigger, Chen et al.[30] argue that a sentence may contain two or more events and propose an architecture using a dynamic multi-pooling layer according to event triggers and arguments in order to extract more crucial information. In contrast to the above works, in this research, we investigated a CNN approach over multi pre-trained word embeddings to support the pre-processing stage in event detection. Our architecture is also inspired by the use of various region windows of convolutional filters to learn features well when bigram-, trigram-, or five-gram-based features are considered simultaneously.

Many studies employ burst detection algorithms[24,27,32] to observe event-related messages so as to verify the occurrence. Unexpected growth in the frequency of messages associated with a category of an event can refer to happening. As in Avvenuti et al.,[22] the sensitivity of the event system is interfered by the noise generated from non-informative messages. In order to tackle this problem, classifiers are adopted as a necessary pre-processing stage. For the temporal model in event detection, an event is considered as an anomaly from normal time series data.

Proposed approach

In this section, we introduce the deep learning–based event detection system, which consists of an improved CNN classifier to identify informative messages and an LSTM-based event detection method, which are shown in Figure 1.

CNN

In CNN, local convolution operation is performed over inputs (e.g. image, embedding features matrix, etc.). As shown in Figure 2, we begin with a tokenization task to convert input sentences into a sentence matrix. The standard CNN learns from observed data which are in the form of a word vector representation of each token of the input sentence, expressed as $A^{T \times D}$, where T is the length of input sentences and D is the dimensionality of the embeddings. The rows of the sentence matrix are word representations of token, as determined from

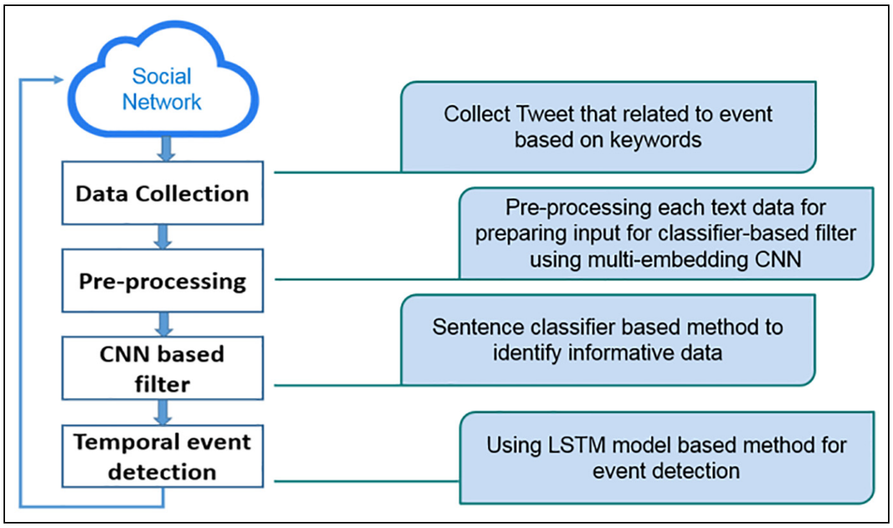

Social Network
Data Collection
Pre-processing
CNN based filter
Temporal event detection
Collect Tweet that related to event based on keywords
Pre-processing each text data for preparing input for classifier-based filter using multi-embedding CNN
Sentence classifier based method to identify informative data
Using LSTM model based method for event detection

Figure 1. Overview architecture of our approach.

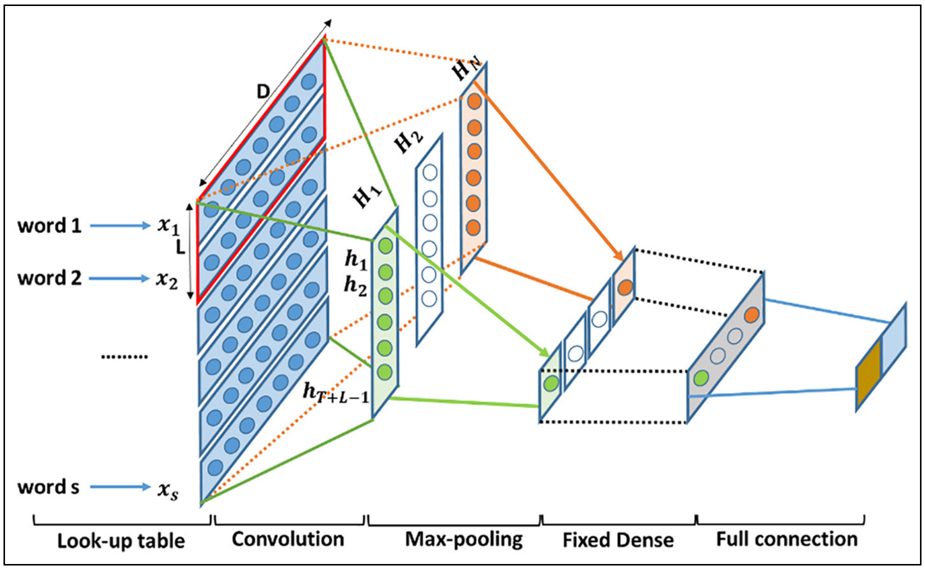


Figure 2. Standard CNN on text classification.

the given Word2vec[33] or GloVe[34] models. A convolution operation denoted as $w_1, w_2, w_k, \ldots$ is applied.[35]

Because text sentences are sequences of words, every full sentence is represented via concatenated rows. It uses filters with widths equal to the size of the word embedding vector. However, these filters correspond to the different sizes of the vertical local region $L = 2, 3, 4, \ldots$ The various sizes of filters help the CNN-based classifier learn many features of natural language. They are the height of the filter, which is the number of adjacent rows concatenated. Suppose that each filter is parameterized by $w \in R^{L \times D}$ to perform the convolution operation on every L-gram phrase of L jointly row $x_{t:t+L-1}$ at the same time. At each sub-sentence, the element-wise multiplication is computed and then summed up. These convoluted results are still active by the non-linear function to generate feature maps with dimensions varying on filters: $H_i = [h_1, h_2, \ldots, h_{T+L-1}]$ with each element determined by equation (1)

$$h_t = f(w.x_{t:t+L-1} + b_t) \tag{1}$$

where $x_{t:t+L-1}$ is the concatenation of L input vectors, $b_t \in R$ is considered a bias term, and f is a non-linear active function (e.g. sigmoid, tanh).

The CNN-based model uses multiple filters for the same region to learn complementary features from the

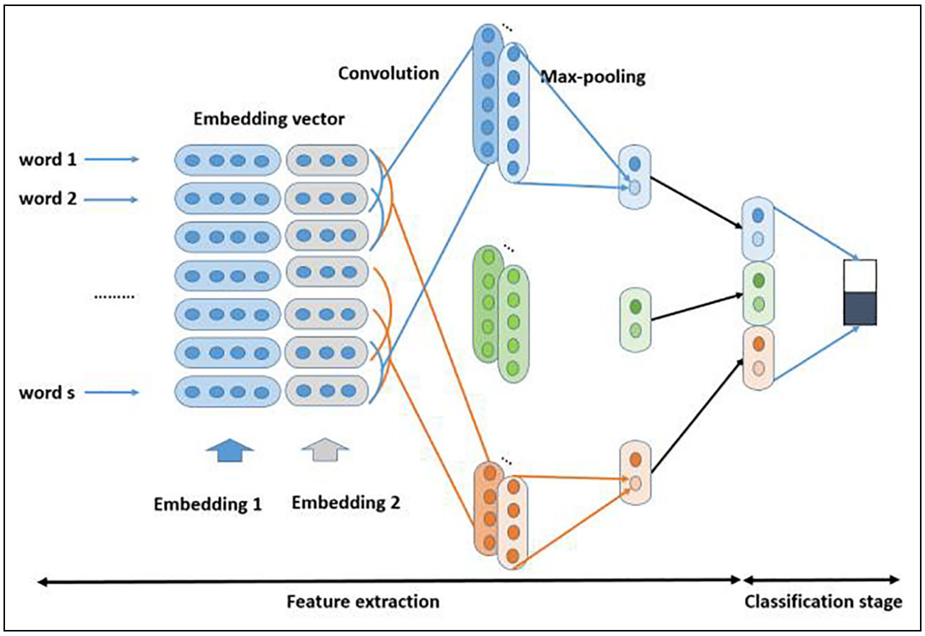


Figure 3. Proposed CNN on multiple word embeddings: concatenated at embedding layer.

same context of sentences. Multiple kinds of filters associated with different sizes L can be used, and for each filter size, there are N different filters.

The size of the feature map of each filter depends on the input sentence length as well as the height of the considered filter, so 1-max pooling operation is selected to apply to each feature map. As a result, the largest number is extracted as equation (2)

$$m = \left[max_p(H_1), max_p(H_2), \ldots, max_p(H_N)\right] \qquad (2)$$

where $max_p(H_1)$ is a max-pooling function with a window of p feature from each feature map H_i. A benefit of pooling is that it provides a fixed size output matrix and retains the most important information from each feature map; these are typically required for classification. This property allows a model to feed variable size sentences and variable size filters while keeping the same size results on output. Finally, combining them together forms a dense feature vector $o \in R^k$ that is fed via a soft-max function for the classification task. The features formed at the penultimate layer are called the fixed dense layer. The fully connected soft-max layer is at the end to perform classification using probability distribution over output labels.[8]

Improved CNN with concatenation of multiple word embeddings

The multichannel idea and variable-size filters are proposed by Kim.[8] However, it just uses a single version of pre-trained embeddings that are initialized with two copies (each considered a channel). One is kept stable, while the other is fine-tuned by the backpropagation algorithm during training. Inspired by the multichannel, we developed this idea with the support of diverse embedding version. With an idea similar to that of the diverse version, Yin and Schutze[12] also propose a method, namely multi-view convolutional neural network (MVCNN), to classify text sentences. Although different word-embedding versions are used as separated channels, it still requires the same dimensionality on input embeddings.

In order to satisfy this requirement, the word embedding should be trained on the same model or same corpus. This is not convenient, as we want to use pre-trained word embeddings, which are often available for downloading. Furthermore, such a model seems very complex to interpret, since it requires a mutual learning phase as a trick to enhance its performance. Therefore, we proposed a method to solve the limitations with the architecture shown in Figure 3. It is easy to implement, and it retains the idea of the variable-size filter of convolution operation.

The performance of the sentence classification can be affected by the input representation, called word embeddings. Kim[8] shows that the multichannel CNN architecture would prevent overfitting in the case of small data. We propose an improved CNN-based classifier with the support of concatenating multiple word embeddings in the input layer. Figure 3 is a visualization of the use of multi-word embedding CNN based on concatenation at the embedding layer. Assuming that we have m various size word embeddings

$D_1, D_2, \ldots, D_m$, they can be trained from separated models or corpora. These embeddings can be extended to another form, as position- or lexical-based embedding (e.g. part of speech (POS) embeddings, etc.). Therefore, we form two single embedding versions to generate diverse embedding versions by concatenating. In other words, we concatenate input sentence matrices called $A_1, A_2, \ldots, A_m (A_l \in R^{T \times D_l})$ of different sizes together corresponding to different dimensionality embeddings in order to apply all convolution filters $\{w_1\}, \{w_2\}, \ldots, \{w_m\}$ as we do the single embedding version. Each $\{w_l\}$ describes a set of the filters having multiple sizes, and for each, there are multiple filters with different learnable weights.

Pre-trained vectors may not always be available for specific words (either in embeding1, embedding2, or both); in such cases, it is optimal that a missed vector for rare words can be supplemented by the sub-vector of others. Unfortunately, in the worst case, we have to randomly initialize them. In particular, for a common word, frequent tokens can have many representations in the embedding layer, instead of only representations. This situation leads to more available information that can be extracted by filters for improving performance at the classification layer.

Applying multi-word input embeddings to the sentence classification task has some advantages.[12] For example, a frequent token can have more representations in input matrix or learned features on the penultimate layer, instead of only one. This means we can obtain more available information. Even though a rare token is missed in some embedding versions, it will be supplemented by others. Our architecture contributes to the development of a diverse version of pre-trained word embeddings and extracting features of sentences with many variable-size convolutional filters.

LSTM-RNN-based event detection

The LSTM architecture is an improved version of the RNN which solves the vanishing gradient problems in RNN. Recently, the use of a recurrent network to form a predictable model or classification model on time series data sets has become popular. LSTM is the core part of our event detection system, which feeds input data underlying time series form. It is capable of predicting several time points ahead (the future data) using input data (the past and the current data).

In this article, the trained model is used to compute the distribution of prediction error as a Gaussian distribution (normal). The prediction error model verifies the likelihood of anomaly (event) behavior. Our approach offers several advantages in that these networks do not require knowledge of abnormalities, handcrafted features, or pre-processing such as fast Fourier transform[25] or wavelet transform.[26]

Consider a time series data $X = \{x^{(1)}, x^{(2)}, \ldots, x^{(n)}\}$ with length L, where each sample $x^{(i)} \in R^m$ is an m-dimensional vector representation of m variables at time instance i. A prediction model f learns parameters as $\emptyset$ annotation to predict the next l values from N input samples. The equation of prediction problem is described as equation (3; k is any start offset)

$$\left\{\hat{x}^{k+N}, \ldots, \hat{x}^{k+N+l-1}\right\} = f\left(\left\{x^k, \ldots, x^{k+N-1}\right\}, \emptyset\right) \tag{3}$$

The function f stands for any prediction method. For a given sufficient training data set, the prediction method adapts its parameters $\emptyset$, which become the characteristic of the normal training.[36] With a predictor trained from the training data set, the prediction error vector can be obtained via some measures[36] such as relative error (RE) and average relative error (ARE). For the sake of simplification, we use the absolution of difference $e^i = \left|x^i - \hat{x}^i\right|$ to estimate the distribution of errors fitting a parametric multivariate Gaussian distribution $N = (\mu, \sum)$. For any point $x^{(i)}$, the anomaly scores are expressed as likelihood $p^{(i)}$ of error vector e^i based on $N = N(\mu, \sum)$. If $p^{(i)} < \tau$, the observation $x^{(i)}$ is classified as "anomaly candidate," else "normal candidate" is assigned to the observation. The threshold τ can be learned by maximizing $F_{\beta-score}$ or pre-defined as a value close to zero (our experience).

Similar to Malhotra et al.,[37] the normal time series are divided into four sets of sequences, namely, S_N (normal training), v_{N1}(normal validation-1), v_{N2} (normal validation-2), and t_N (normal test), while the anomaly time series are divided into the two sets of v_A(anomaly validation) and t_A (anomaly test). The stacked LSTM-based prediction model is implemented with a multiple recurrent LSTM layers, as shown in Figure 4.

The architecture is composed of three LSTM layers with the number of LSTM cells as $\{64, 256, 100\}$, followed by one fully connected layer in a linear activation for regression task. We also regularize between each layer with dropout operation 0.2 (20%). The normal training S_N is used to learn the LSTM prediction models. The normal validation-1 S_N is used for early stopping during the training phase. The error vector calculated on the normal validation-1 time series is used to estimate μ and $\sum$ of the normal distribution using the maximum likelihood estimation (MLE) algorithm. The threshold τ is chosen with maximum of equation (4)

$$F_{\beta-score} = (1+\beta)\frac{P \times R}{\beta^2 P + R} \tag{4}$$

where P is precision and R is recall on the validate sequences in v_{N2} and v_A. We consider anomaly

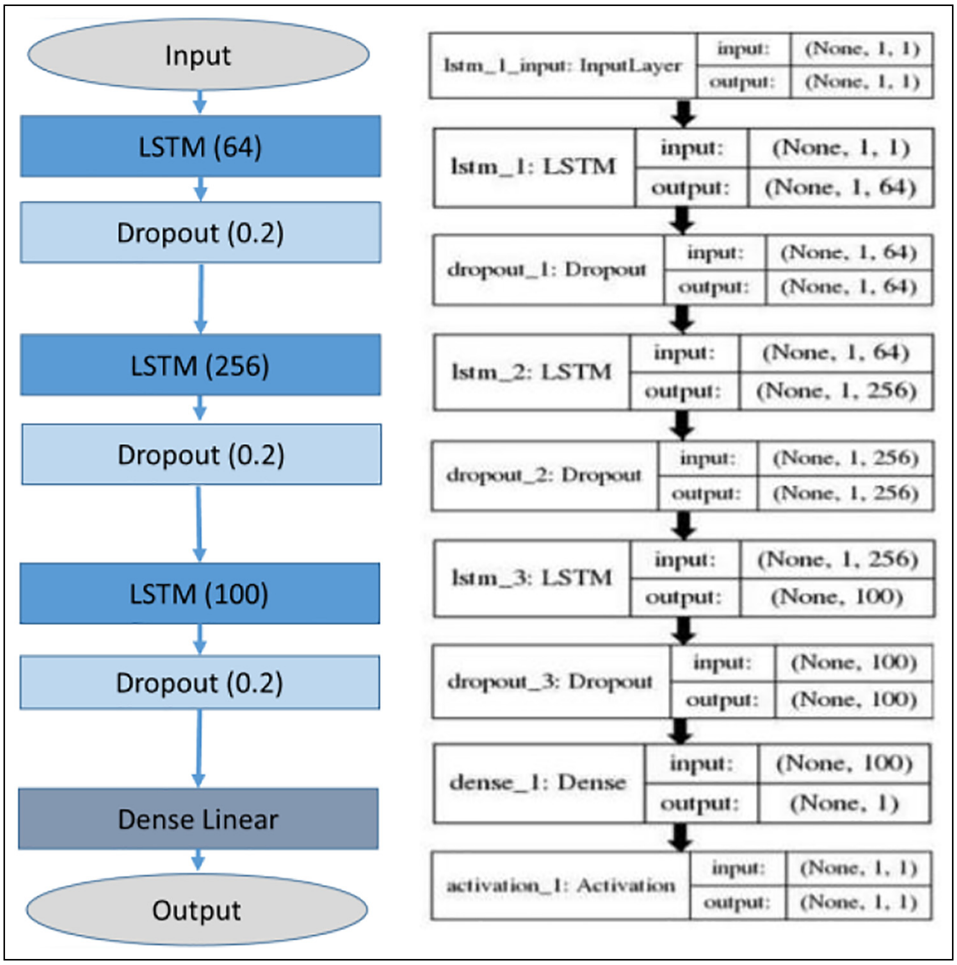

Input
LSTM (64)
Dropout (0.2)
LSTM (256)
Dropout (0.2)
LSTM (100)
Dropout (0.2)
Dense Linear
Output
lstm_1_input: InputLayer | input: (None, 1, 1) | output: (None, 1, 1)
lstm_1: LSTM | input: (None, 1, 1) | output: (None, 1, 64)
dropout_1: Dropout | input: (None, 1, 64) | output: (None, 1, 64)
lstm_2: LSTM | input: (None, 1, 64) | output: (None, 1, 256)
dropout_2: Dropout | input: (None, 1, 256) | output: (None, 1, 256)
lstm_3: LSTM | input: (None, 1, 256) | output: (None, 100)
dropout_3: Dropout | input: (None, 100) | output: (None, 100)
dense_1: Dense | input: (None, 100) | output: (None, 1)
activation_1: Activation | input: (None, 1, 1) | output: (None, 1, 1)

Figure 4. Proposed LSTM-RNN-based predictor model for event detection algorithm.

candidates belonging to the positive class and normal candidates belonging to the negative class.

In order to detect events in real time, we proposed a real-time event detection algorithm (Figure 5) to deal with the streaming data using the classifier model (improved CNN), the predictor model (LSTM), and the error distribution estimation model (using MLE) from previous parts. We have to transform a textual data set of social sensors into a data set of discrete signals for application of the event detection algorithm, that is, time series data underlying discrete signals, which are accumulated from the output of the classifier. Values are the frequency of informative SNS messages in the given interval. Our problem is similar to the anomaly detection of time series signal, in which disaster event detections are experimented.

Algorithm 1 is the real-time event detection algorithm, in which we use both the prediction model and error distribution model to detect temporal information of events. Before running the real-time event detection,

Algorithm 1: Real-time Event Detection

```
Input:
Size of slide window W is fixed
Size of time interval ($\Delta T$)
Threshold for earthquake candidate $\tau$
Threshold for earthquake detection Th
while streaming do
    given $\hat{x}^i$ is prediction using LTSM model
    $x^i$ is accumulated frequency in interval ($\Delta T$)
    calculate Absolute Error $e^i = |x^i - \hat{x}^i|$
    Check anomaly likelihood of $p^{(i)}$:
        if anomaly likelihood $p^{(i)} < \tau$
            Assign earthquake candidate $C^i = 1$
        else
            Assign as non-earthquake candidate $C^i = 0$
        end
    Check window score for event detection:
        if sum ($C^i$)>$Th \times$ size of slide window W
            Event detected
        end
end
```

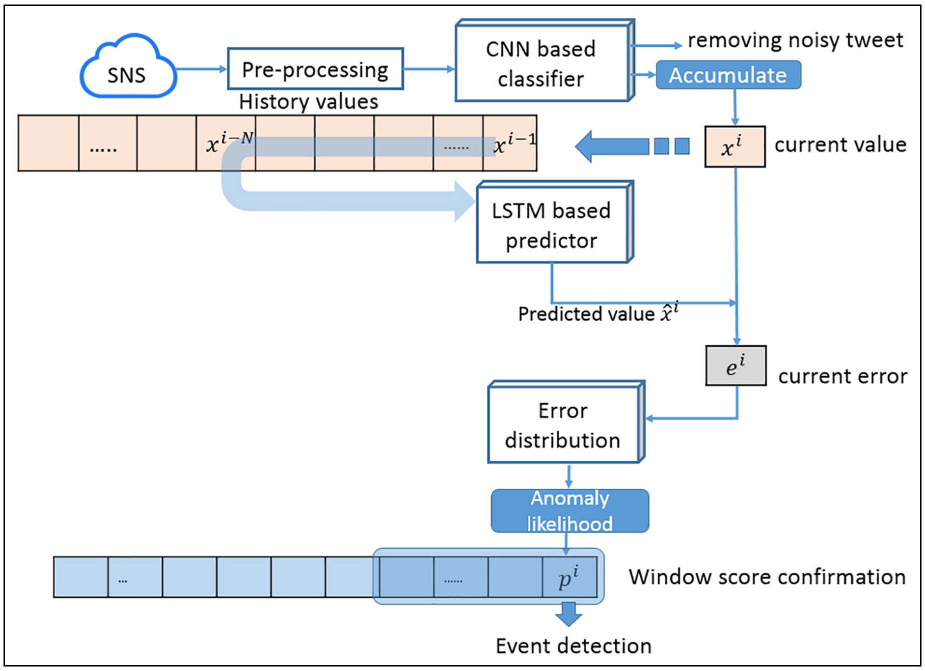


Figure 5. Workflow for real-time event detection.

the LSTM-RNN-based predictor model and error distribution model are trained as described. Informative data are verified by multi-word embedding CNN, and these signals are then approached with the window-based method to be transformed into time series data (sliding window = time interval (ΔT)). Next, the predictor that has already learned normal data behavior can predict future signals using historical signals. In order to check how incoming signals fit to normal signals, the absolute differences between actual signals and predicted signals are computed to estimate anomaly score through the error distribution model.

Experimental results

Data set

In order to evaluate the performance of our approach using a CNN on multi-embedding for identifying informative disaster messages from a social network, we used public data sets from the CrisisLex Project[38] and CrisisNLP project,[39] in which data are manually annotated by crowdsourcing, as shown in Table 1. Since the input data are Twitter messages posted on a social platform, we first preprocess the raw data such as tokenization, language detection, and remove stop-words. These operations also include the normalization of all characters to their lower-case forms, truncating, and punctuation marks. We can annotate some common parts such as user and hyperlink as a special token.

We also conduct the classification on some more benchmark data sets, as detailed in Table 2. CR is the customer review of various products such as cameras and mp3s. They contain positive and negative reviews that are used in Hu and Liu.[40] Subj is the Subjectivity data set,[41] where the task is to classify a sentence as either subjective or objective. MPQA is the opinion polarity data set.[42]

In this work, we consider Word2vec[33] and global vector for word representation[34] (GloVe) as pre-trained word embeddings, as shown in Table 3. Google Word2vec is trained on part of a Google News data set with around 100 billion words with the use of a local context window.[33] GloVe is an embedding version based on global world co-occurrence statistics,[34] which is trained on a corpus of 840 billion tokens from web data, Common Crawl.

Performance of proposed method

Classification performance. The classifier model on the sentence classification was trained with multiple versions of word-embedding vectors from GloVe and Word2vec. All word vectors are not kept static, but we trained parameters of the model, and the pre-vector is fine-tuned for each problem. Table 4 shows the performance of our model against those of the other state-of-the-art methods. We recorded the average accuracy using 10-fold cross-validation (CV).

CCAEs are combinatorial category auto-encoders with combinatorial category grammar operators.[43] Sent-parser is the sentiment analysis-specific parser.[44] NBSVM and MNB are naive Bayes SVM and

Table 1. Disaster data set.[38,39]

Disaster name (year)	# positive	# negative	Total
Philippines floods (2012)	760	145	905
Colorado floods (2013)	768	157	925
Queensland floods (2013)	728	191	919
Manila floods (2013)	628	293	921
Chile earthquakes (2014)	1447	287	1734
Australia fires (2013)	704	245	949

Table 2. Sentence classification data set.

Data	Classes	Average sentence length	Data size
CR	2	19	10,000
Subj	2	23	3774
MPQA	2	3	106,606

CR: customer review; Subj: subjectivity data set; MPQA: opinion polarity data set.

Table 3. Description of each single version of word embeddings.

Name	Vocabulary size	Dimensionality	Source
GloVe	1,193,514	200	Download
Word2vec	3,000,000	300	Download

Table 4. Performance of the proposed method (accuracy measure (average)) on balanced data set using 10-fold cross-validation.

	Model	CR	Subj	MPQA
Compared work	CNN-non-static[8]	84.7	93.0	89.6
	CCAE[43]	–	–	87.2
	Sent-parser[44]	–	–	86.3
	NBSVM[32]	81.8	93.2	86.3
	MNB[32]	80.0	93.6	86.3
	G-dropout[45]	82.1	93.4	86.1
	F-dropout[45]	81.9	93.6	86.3
	Tree-CRF[46]	81.4	–	86.1
	CRF-PR[44]	82.7	–	–
Our approach	Multi-word embedding CNN	**84.08**	**92.96**	**89.6**

CR: customer review; Subj: subjectivity data set; MPQA: opinion polarity data set; CNN: convolutional neural network; CCAE: combinatorial category auto-encoders; NBSVM: naive Bayes support vector machine; MNB: multinomial naive Bayes; CRF-PR: conditional random fields with posterior regularization; Tree-CRF: dependency tree with conditional random fields; G-dropout: Gaussian dropout; F-dropout: fast dropout. Bold values highlights the performance.

multinomial naive Bayes with uni-bigrams, respectively, from Wang and Manning.[32] G-dropout and F-dropout are Gaussian dropout and fast dropout, respectively, from Wang and Manning.[45] Tree-CRF refers to dependency tree with conditional random fields.[46] CRF-PR refers to conditional random fields with posterior regularization.[44]

The results proved that our approach is a useful tool for sentence classification. Our approach based on the improved CNN with pre-trained embedding features enables the automatic learning of text features without hand-generation feature engineering. We compared our proposed method with that of standard CNN[8] that feeds single word embeddings (GloVe or Word2vec), as shown in Table 5. We concatenate more than one single word embedding (GloVe and Word2vec) in order to form a diverse word-embeddings version (GloVe–Word2vec). Since the data sets are imbalanced between

Table 5. Performance of the proposed method (average AUC measure) on imbalanced disaster data set using 10-fold cross-validation.

Word embeddings	Philippines floods	Colorado floods	Queensland floods	Manila floods	Chile earthquake	Australia fire
GloVe(200)[8]	92.13	86.24	89.05	91.12	89.92	85.85
Word2vec(300)[8]	92.27	87.69	88.34	91.01	90.16	86.20
GloVe–Word2vec (proposed method)	**92.47**	85.94	**89.00**	**92.37**	**90.55**	**86.88**

Performance of multiple word-embedding based method that is better than individual word-embedding based method is highlighted by bold values.

the numbers of each class, we mainly considered area under curve (AUC) rather than accuracy score under 10-fold Cross-validation (CV). The imbalanced training data set always appears in disaster situations; therefore, it is better if we use a multiple word-embedding CNN-based classifier in a disaster detection system.

As shown in Table 5, the combination of two embeddings with different dimensionalities is the better point of the proposed approach as compared to the multi-channel model,[4] as that requires the same size of dimensionality. Furthermore, it automatically learns sentence features better and thus obtains a better result in AUC, as highlighted in the last row. We also reported the AUC of the proposed approach in identifying informative messages in the case of imbalanced available data on disaster situations such as flood, earthquake, and fire.

We also compared the result of our approach on embedding features to supervised classification such as SVMs, artificial neural networks (ANNs), and CNNs on bag of word (BoW), as well as in Table 6. The SVM, ANN, and CNN classifiers are trained on both unigrams and n-grams ($n > 1$ defined as a sequence of n contiguous words), and these are reported in Caragea et al.[47]

Table 6. Performance of the proposed method (accuracy) as compared with several methods on flood data set.

Flood data	PCQM[a]
Naïve approach	68.18
SVM(1)[b]	77.74
SVM(2)	78.39
SVM(3)	78.50
ANN(2)	80.46
CNN(2)	82.52
Proposed method (multi-embedding + CNN)	**86.27**

SVM: support vector machine; CNN: convolutional neural network; ANN: artificial neural network.
[a]PCQM is Philippines floods + Colorado floods + Queensland floods + Manila floods data set.
[b]Classifiers are trained on bag of word (BoW; *tf* or *tf-idf*) with (1): unigram; (2): unigram + bigrams and (3): unigrams + bigrams + trigrams.
Bold value highlights the performance of the proposal.

We use three convolutional operations on word embeddings with various size windows in the set $\{3, 4, 5\}$ in order to generate feature maps. There are 100 filters for each size of convolution filter. We trained CNN classifier models by optimizing the cross-entropy using the gradient-based method along with dropout regularization. The rule Adadelta[48] aids learning rate estimation during training. We realized that our proposed approach using multiple embeddings outperforms the standard CNN-based approach which only uses single word embeddings. The fact that millions of tweets are published during a disaster, a few percentage points of improvement, could represent a huge number of informative messages for exploring information. For example, informative messages will be further processed and used in the disaster event detection system as in Nguyen et al.,[49] and it does not need complicated handcraft feature engineering as in Sakaki et al.[27] and Avvenuti et al.,[22] which used a SVM as a classifier.

Temporal event detection performance. Figure 6 illustrates the process of an event detection algorithm using an LSTM-based predictable model and error estimation model on earthquake-related data. From the top down, each sub-figure presents actual time series x^i (blue color) and predicted signal $\hat{x}^i$ (green color) that come from the LSTM-RNN module, absolute error signal $e^i = |x^i - \hat{x}^i|$ denoted by red color, and the two last indicators are anomaly likelihood $p^{(i)}$ and earthquake (yellow color) candidate maker C^i. Because REs are not suitable for our data, where the frequencies of event-mentioning tweets could be zero in a no event situation, we use absolute errors in algorithm 1: Real-time Event Detection. The differences are between the real measurement and predicted value of itself. An observation is assigned as an "earthquake" candidate if the likelihood of error satisfies one threshold which is predefined or learned. Then, window scores are calculated as the sum of the number of "earthquake" candidates in sliding windows and refers to an empirical condition.

From an earthquake data set without labels of events, only the time of actual earthquake occurrence is available. In this case, the threshold is pre-defined with a value close to zero. In practice, if an event-related

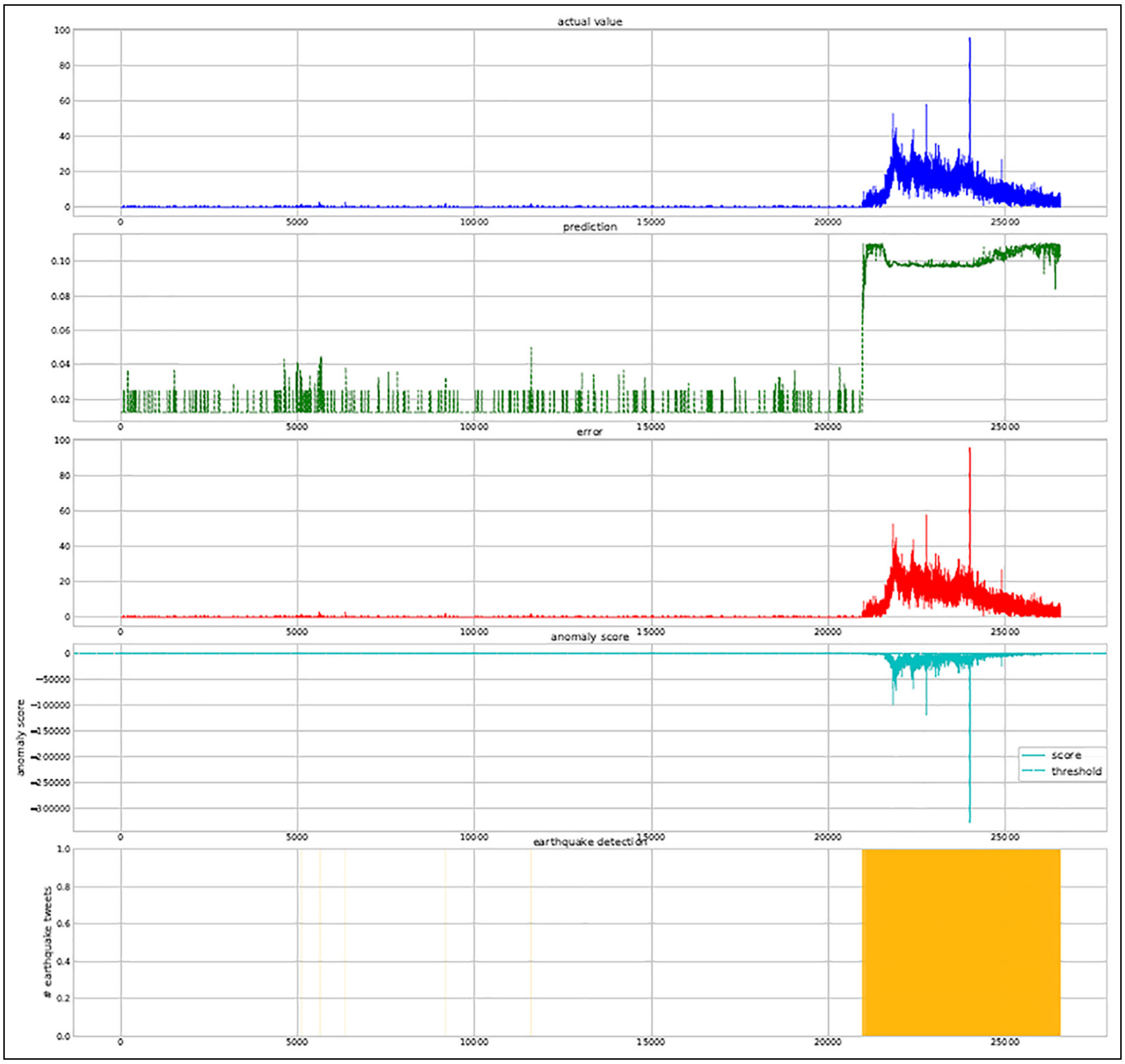


Figure 6. Event detection on earthquake data.

data set is available, we can learn this threshold by maximum score metric (F-score). We experimented the event detection algorithm on earthquake-related tweets in order to obtain the time delay information, as shown in Table 7.

Figure 7 illustrates the performance of event detection in a disaster situation. For obtaining the time delay of our system, we calculate the difference between the response of event detection and the actual earthquake (https://earthquake.usgs.gov) on time aspect. The figure below denotes (blue color) the results from considered references.

Implementation of event detection system. In order to implement the application in a real environment, we should handle some problems related to big data and real-time response. For the acquisition, distribution, and integration of large-scale multi-source information, along with the demand for real-time responses of the whole system, we implemented a Haddoop-based[50] system, as shown in Figure 8. In this system, the social data are collected by a Flume tool according to a specific topic. An Apache Flume-owning ingestion mechanism is used for collecting, aggregating, and transporting large amounts of streaming data from various sources. Hive serves as a data warehouse infrastructure to access data, Structured Query Language (SQL) in Hive facilitates the reading, writing, and managing of large data residing in distributed storage (Hadoop distributed file system (HDFS)). The convolution neural network–based model is used to determine informative data or sorting before moving to LSTM-based anomaly detection. Hadoop streaming is

Table 7. Experiments on earthquake-related tweets.

Actual earthquake (https://earthquake.usgs.gov)		CNN + LSTM-based approach		Improved CNN + LSTM-based approach	
Location	Time	Response	Delay	Response	Delay
M 4.6—10 km SSW of Kyonju, South Korea 35.761°N 129.163°E	19 September 2016, 11:33:58 (UTC)	21:35:25	87 s	11:35:25	87 s
M 4.9—14 km SW of Gyeongju, South Korea 35.743°N 129.106°E	12 September 2016, 10:44:33 (UTC)	10:46:30	117 s	10:45:25	52 s
M 5.6—14 km W of Nereju, Romania 45.714°N 26.528°E	27 December 2016, 23:20:55 (UTC)	23:22:52	117 s	23:21:20	25 s
M 5.6—27 km SW of Hawthorne, Nevada 38.376°N 118.899°W	28 December 2016, 08:18:00 (UTC)	08:19:16	76 s	08:19:12	72 s
M 5.9—13 km NE of Daigo, Japan 36.860°N 140.442°E	28 December 2016, 12:38:49 (UTC)	12:39:57	68 s	12:39:19	30 s
M 5.3—18 km SE of Volcano, Hawaii 19.330°N 155.121°W	08 June 2017, 17:01:19 (UTC)	17:02:49	90 s	17:02:19	60 s
Average time response		96 s = 1 min 36 s		53 s	

CNN: convolutional neural network; LSTM: long short-term memory; UTC: Coordinated Universal Time; M: magnitude.

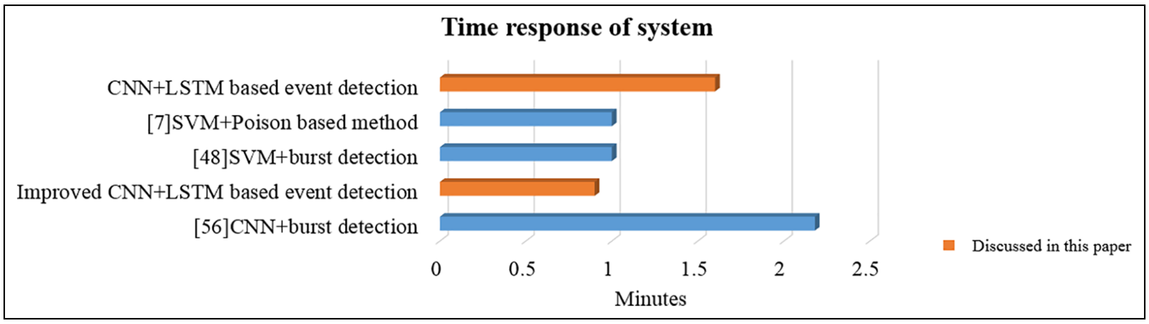
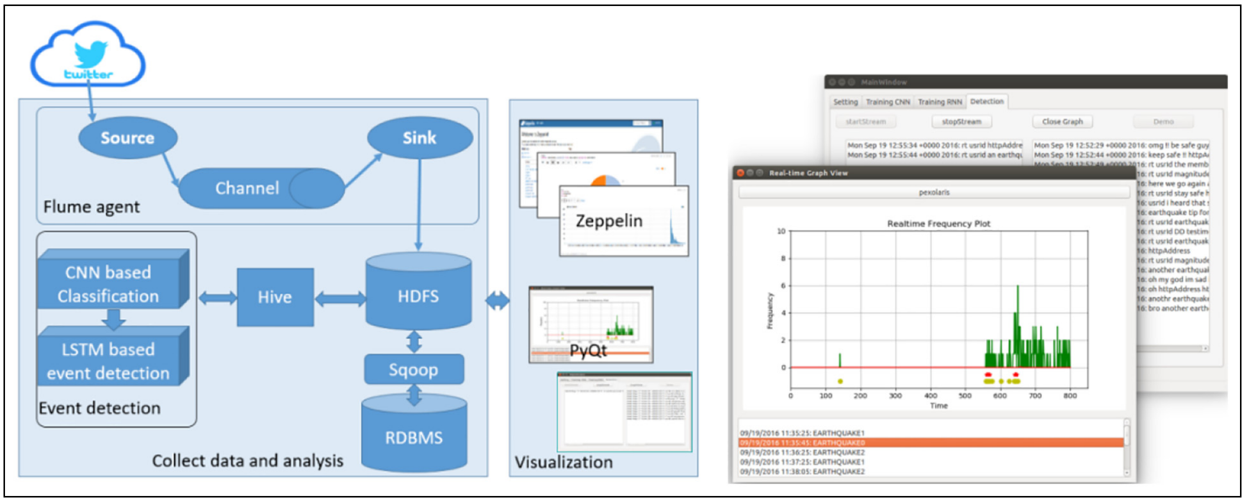


Figure 7. Time delay of real-time event detection systems.

a utility that comes with the Hadoop distribution, so we will use this aid to run executable or script as the mapper or reducer for performing classification and event detection. Sqoop is a tool designed to transfer data between HDFS and the relational database. The analyzed social data are then visualized using many solutions (Zeppelin-based dashboard, desktop-application). Visualizations are very useful and informative for management in both local and remote locations.

Conclusion

Recently, since social media has emerged as a dominant channel for gathering and spreading information during the occurrence of some event, social networks have become a useful means of communicating information. For streaming event detection, we introduced an event detection approach composed of CNN and LSTM models. The CNN augmented with multiple word embedding performs well on short text sentences for identifying informative messages as pre-processing procedure. This overcomes the limitation of multichannel CNN while retaining the simple implementation and increased flexibility in the domain of disaster detection application. We can combine many pre-trained word embeddings with different sizes so as to obtain better results as compared to standard CNN on individual word embedding. The LSTM-based predictor is proposed on time series data to learn temporal signal features that are used to detect anomaly patterns. This anomaly detection model shows potential for



```
Mon Sep 19 12:55:34 +0000 2016: rt usrid httpAddre
Mon Sep 19 12:55:44 +0000 2016: rt usrid an earthqu
```

```
Mon Sep 19 12:52:29 +0000 2016: omg !! be safe guy
Mon Sep 19 12:52:44 +0000 2016: keep safe !! httpA
Mon Sep 19 12:52:49 +0000 2016: rt usrid the memb
16: rt usrid magnitude
16: here we go again
16: rt usrid stay safe h
16: usrid i heard that
16: earthquake tip for
16: rt usrid earthquak
16: rt usrid DO testim
16: rt usrid earthquak
16: httpAddress
16: rt usrid magnitude
16: another earthquak
16: oh my god im sad
16: oh httpAddress ht
16: anothr earthquake
16: bro another earth
```

```
09/19/2016 11:35:25: EARTHQUAKE1
09/19/2016 11:35:45: EARTHQUAKE0
09/19/2016 11:36:25: EARTHQUAKE2
09/19/2016 11:37:25: EARTHQUAKE1
09/19/2016 11:38:05: EARTHQUAKE2
```

Figure 8. Hadoop-based event detection architecture.

application in different time series data such as electrocardiogram (ECG) and sensor signal. The proposed method showed that the time response of detection is acceptable, and the system can adapt many domains.

This work proposed a general application framework for social sensor–based event detection systems such as disaster events, sports events, social events, and even political events. The current limitation of our event detection system is to just focus on a single topic–based tweet collection using relevant keywords. In order to improve the convenience of the system for different events, we will investigate the automatic cluster or classifier-based method using the semantic map technique. The content of Twitter will be represented in form of semantic maps, from which the main topics as well as hot events of the social network may easily be read. In the future, we can implement our approach on a big data framework, as well as the integration of Global Positioning System (GPS) information for visualization over a dashboard.

Declaration of conflicting interests

The author(s) declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

Funding

The author(s) disclosed receipt of the following financial support for the research, authorship, and/or publication of this article: The research was supported by the MSIP (Ministry of Science, ICT and Future Planning), Korea, under the ITRC (Information Technology Research Center) support program (IITP-2017-2016-0-00314) supervised by the IITP (Institute for Information & communications Technology Promotion). The work was also supported by the National Research Foundation of Korea (NRF) funded by the Ministry of Education (NRF-2017R1A2B4011409).

ORCID iDs

Van Quan Nguyen https://orcid.org/0000-0001-9703-0692
Hyung-Jeong Yang https://orcid.org/0000-0003-3024-5060

References

1. McCord M and Chuah M. Spam detection on Twitter using traditional classifiers. In: *Autonomic and trusted computing*, vol. 6906 [Lecture Notes in Computer Science]. Berlin; Heidelberg: Springer, 2011, pp.175–186.
2. Schein AI, Alexandrin P, Lyle HU, et al. Methods and metrics for cold-start recommendations. In: *Proceedings of the 25th annual international ACM SIGIR conference on research and development in information retrieval* (SIGIR '02), Tampere, 11–15 August 2002, pp.253–260. New York: ACM.
3. Shekar C, Wakade S, Liszka KJ, et al. Mining pharmaceutical spam from Twitter. In: *2010 10th international conference intelligent systems design and applications (ISDA)*, Cairo, Egypt, 29 November–1 December 2010, pp.813–817. New York: IEEE.
4. Kathy L, Diana P, Ramanathan N, et al. Twitter trending topic classification. In: *Proceedings of the 2011 IEEE 11th international conference on data mining workshops (ICDMW'11)*, Vancouver, BC, Canada, 11 December 2011, pp.251–258. New York: IEEE.
5. Sharifi B, Mark-Anthony H and Jugal K. Automatic summarization of Twitter topics. In: *National workshop on design and analysis of algorithms*, Tezpur, India, 22–23 January 2010, pp.121–128, http://citeseerx.ist.psu.edu/viewdoc/summary?doi = 10.1.1.473.2602
6. O'Connor B, Balasubramanyan R, Routledge BR, et al. From tweets to polls: linking text sentiment to public opinion time series. In: *The international AAAI conference on weblogs and social media (ICWSM)*, Washington, DC, 23–26 May 2010, pp.122–129, https://www.aaai.org/ocs/index.php/ICWSM/ICWSM10/paper/viewFile/1536/1842
7. Zubiaga A, Spina D, Fresno V, et al. Classifying trending topics: a typology of conversation triggers on

Twitter. In: *Proceedings of the 20th ACM international conference on information and knowledge management (CIKM '11)*, Glasgow, 24–28 October 2011, pp.2461–2464. New York: ACM.

8. Kim Y. Convolutional neural networks for sentence classification. In: *11th conference on empirical methods in natural language processing*, Doha, Qatar, 25–29 October 2014, pp.1746–1751. Stroudsburg, PA: Association for Computational Linguistics.
9. Kalchbrenner N, Grefenstette E and Blunsom P. A convolutional neural network for modelling sentences. *arXiv* 1404.2188, 2014.
10. Collobert R, Weston J, Bottou L, et al. Natural language processing (almost) from scratch. *J Mach Learn Res* 2011; 12: 2493–2537.
11. Zhang Y and Wallace BC. A sensitivity analysis of (and practitioners' guide to) convolutional neural networks for sentence classification, 2015, https://arxiv.org/abs/1510.03820
12. Yin W and Schutze H. Multichannel variable-size convolution for sentence classification. In: *Proceedings of the conference on computational natural language learning*, Beijing, China, 30–31 July 2015, pp.204–214. Stroudsburg, PA: Association for Computational Linguistics.
13. Li Q, Nourbakhsh A, Shah S, et al. Real-time novel event detection from social media. In: *Proceedings of the IEEE 33rd international conference data engineering (ICDE)*, San Diego, CA, 19–22 April 2017, pp.1129–1139. New York: IEEE.
14. Phuvipadawat S and Murata T. Breaking news detection and tracking in Twitter. In: *Proceedings of the 2010 IEEE/WIC/ACM international conference on web intelligence and international conference on intelligent agent technology—workshops*, Toronto, ON, Canada, 31 August–3 September 2010, pp.120–123. New York: IEEE.
15. O'Connor B, Krieger M and Ahn D. Tweetmotif: exploratory search and topic summarization for Twitter. In: *Proceedings of the fourth international conference on weblogs and social media (ICWSM, 2010)*, Washington, DC, 23–26 May 2010. PA: AAAI Press.
16. Sankaranarayanan J, Samet H, Teitler BE, et al. Twitterstand: news in tweets. In: *Proceedings of the 17th ACM SIGSPATIAL international symposium on advances in geographic information systems (ACM-GIS 2009)*, Seattle, WA, 4–6 November 2009, pp.42–51. New York: ACM.
17. Fung GPC, Yu JX, Yu PS and Lu H. Parameter free bursty events detection in text streams. *Proceedings of the 31st international conference on very large data bases*, Trondheim, 30 August–2 September 2005, pp.181–192. New York: ACM.
18. Becker H, Naaman M and Gravano L. Beyond trending topics: real-world event identification on Twitter. In: *Proceedings of the fifth international conference on weblogs and social media*, Barcelona, 17–21 July, 2011. Palo Alto, CA: AAAI.
19. Shamma DA, Kennedy L and Churchill EF. Peaks and persistence: modeling the shape of microblog conversations. In: *Proceedings of the 2011 ACM conference on computer supported cooperative work (CSCW 2011)*, Hangzhou, China, 19–23 March 2011, pp.355–358. New York: ACM.
20. Yang J and Leskovec J. Patterns of temporal variation in online media. In: *Proceedings of the forth international conference on web search and web data mining (WSDM 2011)*, Hong Kong, China, 9–12 February, 2011, pp.177–186. New York: ACM.
21. Li R, Lei KH, Khadiwala R, et al. TEDAS: A Twitter-based event detection and analysis system. In: *IEEE 28th international conference on data engineering (ICDE 2012)*, Washington, DC, 1–5 April 2012, pp.1273–1276. New York: IEEE.
22. Avvenuti M, Cresci S, La Polla MN, et al. Earthquake emergency management by social sensing. In: *The second IEEE international workshop on social and community intelligence*, Budapest, 24–28 March 2014. New York: IEEE.
23. Nguyen DT and Jai EJ. Real-time event detection for online behavioral analysis of big social data. *Future Gener Comp Systems* 2017; 66: 137–145.
24. Nguyen VQ, Yang H-J, Kim K and Oh A-R. Real-time earthquake detection using convolutional neural network and social data. In: *2017 IEEE third international conference multimedia big data*, Laguna Hills, CA, 19–21 April 2017, pp.154–157. New York: IEEE.
25. He Q, Chang K and Lim E. Analyzing feature trajectories for event detection. In: *Proceedings of the 30th annual international ACM SIGIR conference on research and development in information retrieval (SIGIR 2007)*, Amsterdam, 2007, pp.207–214. New York: ACM.
26. Weng J and Lee B. Event detection in Twitter. In: *Proceedings of the fifth international conference on weblogs and social media*, Barcelona, 17–21 July 2011. Palo Alto, CA: AAAI.
27. Sakaki T, Okazaki M and Matsuo Y. Tweet analysis for real-time event detection and earthquake reporting system development. *IEEE T Knowl Data En* 2013; 25(4): 919–931.
28. Hochreiter S and Schmidhuber J. Long short-term memory. *Neural Comput* 1997; 9(8): 1735–1780.
29. Chatfield AT and Uuf B. Twitter early tsunami warning system: a case study in Indonesia's natural disaster management. In: *46th Hawaii IEEE international conference on system sciences (HICSS)*, Maui, HI, 7–10 January 2013. New York: IEEE.
30. Chen Y, Xu L, Liu K, et al. Event extraction via dynamic multi-pooling convolutional neural networks. In: *Proceedings of the 53rd annual meeting of the association for computational linguistics*, Beijing, China, 26–31 July 2015, pp.167–176. Stroudsburg, PA: Association for Computational Linguistics.
31. Nguyen TH and Grishman R. Event detection and domain adaptation with convolutional neural networks. In: *Proceedings of the 53rd annual meeting of the association for computational linguistics*, Beijing, China, 26–31 July 2015, pp.365–371. Stroudsburg, PA: Association for Computational Linguistics.
32. Wang S and Manning CD. Baselines and bigrams: simple, good sentiment and topic classification. In: *Proceedings of the 50th annual meeting of the association for computational linguistics: short papers (ACL 2012)*, vol. 2, Jeju,

Republic of Korea, 8–14 July 2012, pp.90–94. Stroudsburg, PA: Association for Computational Linguistics.

33. Mikolov T, Sutskever I, Chen K, et al. Distributed representations of words and phrases and their compositionality. In: *Advances in neural information processing systems*, Lake Tahoe, NV, 5–10 December 2013, pp.3111–3119. Red Hook, NY: Curran Associates.
34. Pennington J, Socher R and and Manning C. Glove: global vectors for word representation. In: *Proceeding of the empirical methods in natural language processing (EMNLP)*, Doha, Qatar, 25–29 October 2014, pp.1532–1543. Stroudsburg, PA: Association for Computational Linguistics.
35. Collobert R and Weston J. A unified architecture for natural language processing. In: *Proceedings of the 25th International Conference on Machine learning (ICML '08)*, Helsinki, 5–9 July 2008, pp.160–167. New York: ACM Press.
36. Bontemps L, Cao VL, McDermott J, et al. Collective anomaly detection based on long short-term memory recurrent neural networks. In: *Proceedings third international conference* (FDSE 2016), Can Tho City, 23–25 November 2016, pp.141–152. Cham: Springer.
37. Malhotra P, Vig L, Shroff G, et al. Long short term memory networks for anomaly detection in time series. In: *23rd European symposium on artificial neural networks, computational intelligence and machine learning (ESANN)*, Bruges, 22–24 April 2015, https://www.elen.ucl.ac.be/Proceedings/esann/esannpdf/es2015-56.pdf
38. Crisis collections, http://crisislex.org/data-collections.html#CrisisLexT26
39. Twitter datasets from crises, http://crisisnlp.qcri.org/lrec2016/lrec2016.html
40. Hu M and Liu B. Mining and summarizing customer reviews. In: *Proceedings of the ACM SIGKDD international conference on knowledge discovery & data mining*, Seattle, WA, 22–25 August 2004. New York: ACM.
41. Pang B and Lee L. A sentimental education: sentiment analysis using subjectivity summarization based on minimum cuts. In: *Proceedings of meeting of the association for computational linguistics (ACL-2004)*, Barcelona, 21–26 July 2004. Stroudsburg, PA: Association for Computational Linguistics.
42. Wiebe J, Wilson T and Cardie C. Annotating expressions of opinions and emotions in language. *Lang Resour Eval* 2005; 39(2–3): 165–210.
43. Hermann K and Blunsom P. The role of syntax in vector space models of compositional semantics. In: *Proceedings of the 51st annual meeting of the association for computational linguistics*, Sofia, 4–9 August 2013. Stroudsburg, PA: Association for Computational Linguistics.
44. Yang B and Cardie C. Context-aware learning for sentence-level sentiment analysis with posterior regularization. In: *Proceedings of the 52st annual meeting of the association for computational linguistics*, Baltimore, MD, 23–25 June 2014. Stroudsburg, PA: Association for Computational Linguistics.
45. Wang S and Manning C. Fast dropout training. In: *Proceedings of the 30th international conference on machine learning*, Atlanta, GA, 16–21 June 2013. JMLR.org ©2013
46. Nakagawa T, Inui K and Karohashi S. Dependency tree based sentiment extraction using CRFs with hidden variables. In: *Proceedings of the 2010 annual conference of the North American chapter of the association for computational linguistics*, Los Angeles, CA, 2–4 June 2010, pp.786–794. Stroudsburg, PA: Association for Computational Linguistics.
47. Caragea C, Silvescu A and Tapia A. Identifying informative messages in disasters using Convolutional Neural Networks. In: *Proceedings of the 13th international ISCRAM conference*, Rio de Janeiro, Brazil, 22–25 May 2016. Information Systems for Crisis Response and Management (ISCRAM).
48. Zeiler MD. ADADELTA: an adaptive learning rate method. *Cornell University*, December 2012, https://arxiv.org/abs/1212.5701
49. Nguyen VQ, Yang HJ, Kim Y, et al. Design and implementation of event detection system on Hadoop. In: *Proceedings 1st International Conference Software & Smart Convergence (ICSSC 2017)*, Vol. 6, no. 2, Vladivostok, 27–30 June 2017, pp.201–205. Korean Institute of Smart Media.
50. http://hadoop.apache.org