Quest Journals
Journal of Software Engineering and Simulation
Volume 7 ~ Issue 9 (2021) pp: 38-45
ISSN(Online) :2321-3795 ISSN (Print):2321-3809
www.questjournals.org

Research Paper

A Method for Classifying Tweets about Emergency Events using Deep Neural Networks

D.J.S. Sako[1], F.E. Onuodu[2], B.O. Eke[2]
[1]*Department of Computer Science, Rivers State University, Port Harcourt, Nigeria*
[2] *Department of Computer Science, University of Port Harcourt, Nigeria*

ABSTRACT
Social media platforms have increasingly become a source of high volume, real-time information describing events and developing information in a timely manner especially during disasters or emergencies. However, the challenges of incorporating social media data into emergency situations response, especially for detecting small scale emergency events where there are only small bits of information, remain. To deal with these difficulties, in this paper, we conduct studies on every day small scale incidents and present a method of classifying tweets using convolutional neural networks trained on top of pre-trained word vectors that perform classification in layers in order to (i) determine whether or not the tweet is incident-related; (ii) if a tweet is incident-related, then which category it belongs to. Experiments on a home-grown dataset show that a system using the proposed method and architectures can classify tweetswith an F_1-score of 86.57%.
KEYWORDS: *Social media, classification, emergency situations, deep neural networks, Twitter,*

Received 20 September, 2021; Revised: 03 October, 2021; Accepted 05 October, 2021 © The author(s) 2021. Published with open access at www.questjournals.org

I. INTRODUCTION

Each society and its components (individuals, groups, organizations, communities, etc) are occasionally and usually in an unexpected way exposed to disasters, crisis/emergency situations. Whether an emergency situation is triggered by a natural phenomenon (e.g. storms, earthquakes, bushfire, flooding, etc.), or at the instigation of humans (terrorism, riots, communal conflicts, crimes, shootings, a technological failure or an industrial accident, widespread power outages, explosions, structure collapses, fire in residential buildings, etc), the normal functioning of a society and its parts will be disturbed in a more or less serious manner [2].

The ubiquity of social media platforms presents an opportunity to harness developing information to contributes to situation awareness [13] for management and response teams [5] during crisis or emergency situations.With increasingly widespread access to mobile phones and the Internet, in times of crisis, it is increasingly common for the public or people affected by the crisis/event to use social media to broadcast their needs, propagate news, post text messages and photographs, requesting assistance, describing the situation on the ground and staying abreast of evolving situations [10]. This huge resource may potentially gather a valuable body of information regarding incidents that differ significantly by type, location, and time. Such availability of content-rich data is extremely valuable for emergency management (EM) personnel as they can take more accurate decisions in emergency situations [17].

In this paper, we use Convolutional Neural Network (CNN), a deep neural network, to address two types of information needs of response agencies and organizations: (i) identifying informative tweets by determining whether or not the tweet is incident-related as most messages contain irrelevant information not useful for emergency response and management; (2) if a tweet is incident-related, then which category of target incident/emergency events it belongs to.

CNNs use distributed representation of words and learn the representation as well as higher level features automatically for the classification task. We propose a two-layer convolutional neural network (CNN) classification of tweets for incident types.

In the next section, we present existing work on emergency tweet classification. Deep learning convolutional neural network and word embedding are reviewed and our proposed methodology is presented. We describe our dataset and report on the experimental results before concluding with final remarks.

II. RELATED WORK

Research in recent years has uncovered the increasingly importance of social media during emergency situations and shown that information broadcast via social media can enhance situational awareness during a crisis situation [1, 15, 16]. [14] were one of the first to study and analyze microblogs usage and information lifecycles during crisis situations. They performed a qualitative analysis on tweets published during a flooding incident to study the behaviour of microbloggers.

[11,12] detected earthquakes by monitoring Twitter in Japan. They developed a system that acts as a quick alarm for people to prepare themselves for the coming disaster by developing a classifier that works on tweets retrieved with two specific query terms related to earthquakes: earthquake and shaking. They used three sets of features: tweet length and position of query term in the tweet, tweet words, and the context of the query term in the tweet. They reported an F-measure of around 73% for two queries on a data set of 597 positive examples.

[8] presented an online learning model namely Convolutional Neural Network for the purpose of classifying tweets in a disaster response scenario. They proposed a new online learning algorithm for training CNNs in online fashion and showed that online training of the model perfectly suits the disaster response situation.

III. SYSTEM OVERVIEW AND METHODS

In this section, we present the approach used to retrieve, process, and classify incident-related tweets. The block diagram of the proposed method is shown in Figure 1. To perform classification for each of the layers of the classification tasks, tweets are captured using Twitter Streaming API, filtered by incident-related keywords. The text content of the tweet is extracted, preprocessed and given as an input to the Embedding layer, which gives the word vectors from the tweets. The word vectors are sent to the classifier for training the model.The trained model is used for predicting the label of the testing tweets. The output of the classifier can be predicted by the class label (either incident-related tweet or not) in the first instance and if it is incident-related, then which category of target incident or emergency events it belongs to.

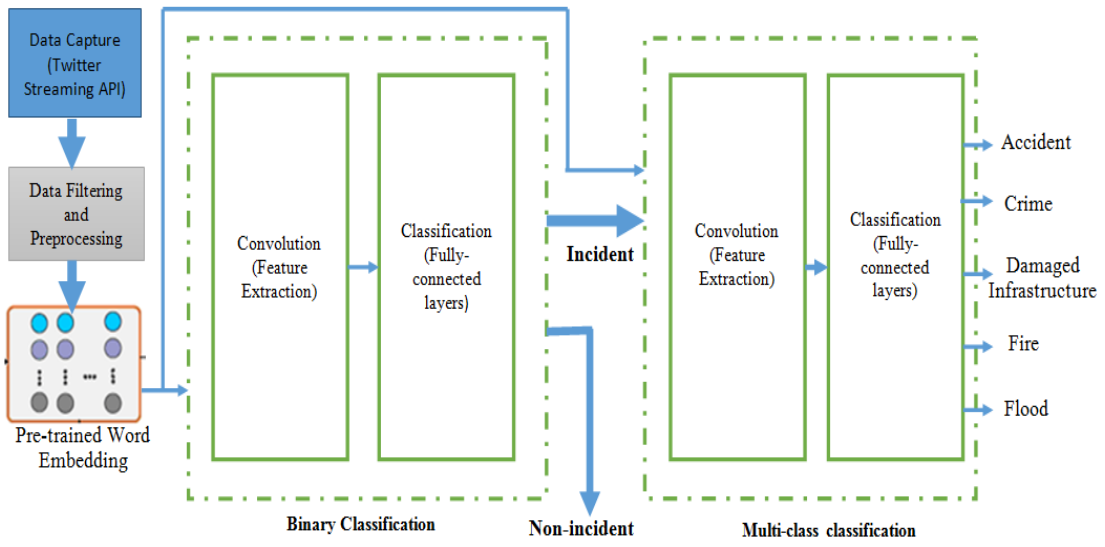
Data Capture (Twitter Streaming API)
Data Filtering and Preprocessing
Pre-trained Word Embedding
Convolution (Feature Extraction)
Classification (Fully-connected layers)
Incident
Non-incident
Convolution (Feature Extraction)
Classification (Fully-connected layers)
Accident
Crime
Damaged Infrastructure
Fire
Flood
Binary Classification
Multi-class classification

Figure 1: Structure of the Proposed CNN Model for classifying incident types of reported target emergency events

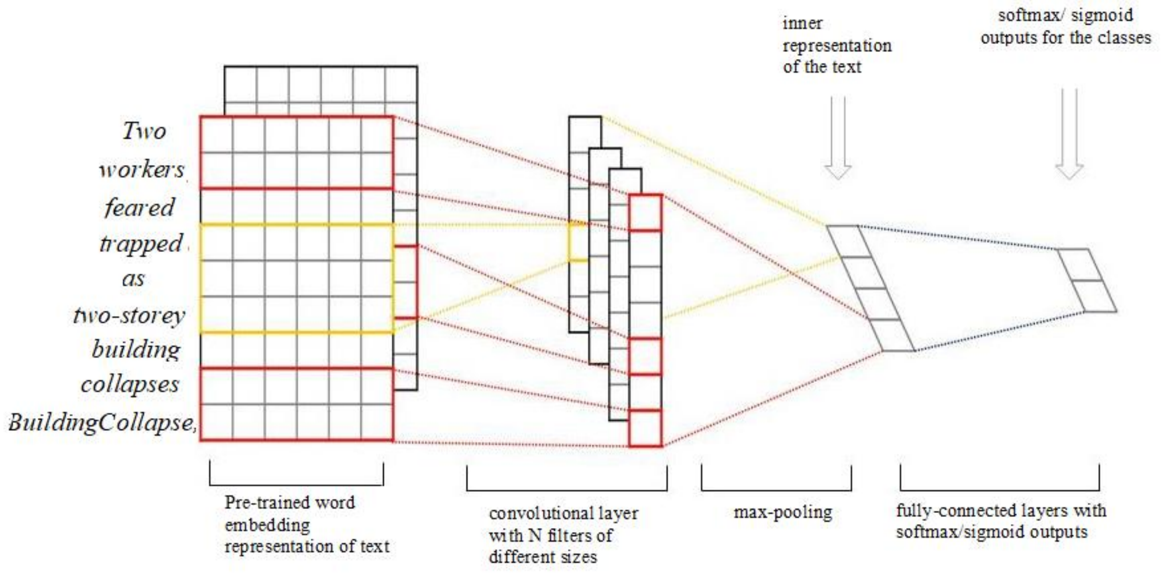
Two
workers
feared
trapped
as
two-storey
building
collapses
BuildingCollapse
inner representation of the text
softmax/ sigmoid outputs for the classes
Pre-trained word embedding representation of text
convolutional layer with N filters of different sizes
max-pooling
fully-connected layers with softmax/sigmoid outputs

Figure 2: Convolutional neural network on a sample tweet: "*Two workers feared trapped as two-storey building collapses #BuildingCollapse"*

3.1 Convolutional Neural Network (CNN)

In order to effectively classify tweets, which are short and informal, a classification model should learnthe key features at different levels of abstraction. In this paper, we propose a convolutional neural network (CNN) for the classification tasks.CNN, as a deep learning neural network (DNN), provides efficient computational models using combinations of nonlinear processing elements organized in layers and captures the most salient n-gram information by means of its convolution and max-pooling operations [8] and has been shown to be effective for sentence-level classification tasks [3]. Figure 2 presents the CNN model architecture, adapted from Kim (2014), used to learn the internal representation of the tweet "*Two workers feared trapped as two-storey building collapses #BuildingCollapse"*.

The organization of simple elements in CNN allows the total network to generalize (i.e., predict correctly on new data) [4, 6]. CNNs can be quite effective in classifying tweets during a disaster situation because of their distributed representation of words and automatic feature learning capabilities. Each word in the vocabulary V is represented by a D dimensional vector in a shared look-up table $L \in \mathbb{R}^{|V|xD}$ where L is considered a model parameter to be learned. We initialize L randomly or using GloVe [9] pretrained word embedding vectors.

The input to our CNN is a matrix where each row is a real-valued vector representation of each word in the caption or tweet. Our word-vectors are obtained from a pretrained GloVe word embeddings. Given an input tweet $s = (w_1, \dots, w_T)$, we first transform it into a feature sequence by mapping each word token $w_t \in s$ to an index in L.The look-up layer then creates an input vector$x_t \in \mathbb{R}^D$ for each token w_t which are passed through a sequence of convolution and pooling operations to learn high-level feature representations. This vector is then fed further down in the network to capture the most relevant features of the tweet and finally to a fully connected layers to perform prediction. Algorithm1 demonstrates how our CNN model can be trained to learn key features of the tweet at different levels of abstraction for binary and multi-class classification tasks.

3.2 Word Embedding

It is common to perform transfer learning with natural language processing problems that use text as input or output. For these types of problems, a word embedding is used that is a mapping of words to a high-dimensional continuous vector space where different words that have a similar meaning (based on their usage) also have a similar vector representation. In our work, we initialize the word embeddings L randomly, and learn them as part of model parameters by backpropagating the errors to the look-up layer. We also use GloVepre-trained word embeddings to better initialize our models, and we fine-tune them for our task, which turns out to be beneficial.

Algorithm 1: Training CNN Model

Input: Input vector $x_t \in \mathbb{R}^{ND}$ of *D*-dimensional word embeddings (word vector) of the *i*th word in the tweet text for each token w_t

Output: Class probability, performance metric values

```
1.  Represent each word as a vector.
        $\mathbf{X}=\{x_1,\ldots,x_t\} \in \mathbb{R}^{ND}$
2.  for each row in the input vector, X,
    Perform convolution operation by applying a filter $\boldsymbol{w} \in \mathbb{R}^{L.D}$ to a window of $L$ words to
    produce a new feature, $h_t$, is generated from a window of words $x_{t:t+L-1}$ by
            $h_t = f(w.x_{t:t+L-1} + b_t)$                                        (1)

    where $x_{t:t+L-1}$ denotes the concatenation of $L$ input vectors, $b_t$ is a bias term, and $f$ is a
    nonlinear activation function such as ReLU.
        [A filter is also known as kernel or a feature detector]
    for N different filters [get N different feature maps]
        Apply this filter to each possible L-word window in the tweet $\{x_{1:L}, x_{2:L+1}, \ldots, x_{T-L+1:T}\}$
    togenerate a feature map

            $h_i = [h_1, h_2, \ldots, h_{T+L-1})$                                 (2)

        with $x_t \in \mathbb{R}^{T+L-1}$
    end for
    end for[Repeat until the last row is reached]
3.  for each feature map in N feature maps [After the convolution]
    Apply a max-pooling operation over the feature map and take the maximum value
    $\boldsymbol{m} = \max\{h_i\}$ as the feature corresponding to this particular filter
        $\boldsymbol{m} = [w_p(h_1), \ldots, w_p(h_N)]$                           (3)
    where $w_p(h_i)$refers to the max operation applied to each window of $p$ features in the
    feature map, $h_i$.
    endfor
4.  Model interactions between the features picked up by the filters and the pooling byfeeding the
    vector, $\boldsymbol{m}$, into a fully connected layer (dense layer of hidden nodes on top of the pooling layer) to
    learn mid-level text representation.

            $z = f(V_{\boldsymbol{m}} + b_h)$                                     (4)

    where $V$ is the weight matrix, $b_h$ is a bias vector, and $f$ is a nonlinear activation. The dense layer
    naturally deals with variable sentence lengths by producing fixed size output vectors $z$, which are
    fed to the output layer for classification.
5.  Produce the sigmoid/softmax outputs (output layer defines a probability distribution) for each class.
    5.1 For binary classification with sigmoid function (sig), we define the Bernoulli
        distribution (a discrete probability distribution) as:

            $p(y|s,\theta) = Ber(y|sig(W^T Z + b))$                              (5)
        where$w$ are the weights from the dense layer to the output layer and $b$ is a
        bias term.
    5.2 For the multi-class classification with a softmax function, the probability of k-thlabel in the
        output for classification into $K$ classes:
            $P(y=k|s,\theta) = \frac{\exp(\boldsymbol{w}_k^T \boldsymbol{z} + b_k)}{\sum_{j=1}^{K} \exp(\boldsymbol{w}_j^T \boldsymbol{z} + \boldsymbol{b_j})}$   (6)
        where, $w_k$ are the weights associated with class $k$ in the output layer.
6.  Calculate cross entropy Loss using the objective function, $f(\theta)$,

            $f(\theta) = \sum_{n=1}^{N} \sum_{k=1}^{K} y_{nk} \log P(y_n = k|s_n,\theta)$   (7)
    where N is the number of training examples and $y_{nk} = I(y_n = k)$ is an indicator variable to encode
    the gold labels, ie$y_{tk} = 1$ if the gold label $y_t = k$, otherwise 0 [8].
7.  Fit the models by minimizing the cross-entropy loss between the predicted distributions
    $\hat{y}_{n\theta} = p(y_n|s_n,\theta)$ and the target distribution $y_n$ (ie the gold labels).
8.  The classification decision for the classifier made based on the highest decision score in $P$:
            $Class(j) = arg \max_{1 \le j \le h} P$
    (8)
```

9.	Calculate the precision, recall and F_1

Table 1: Description of the datasets

	Dataset		
Class	**DS-1**	**DS-2**	**Description**
Damaged Infrastructure	2830	38266	Collapsed/Damaged buildings or roads, destroyed bridges, utilities/services interrupted (etc falling electricity pole).
Fire	3036	38804	Building fire
Flood	2935	37815	floods caused by heavy rainfall
Traffic Accidents	3004	37900	Motor vehicle crash such as motorcycle accident, car accident, plane crash, etc.
Crime and Civil Disorder	3038	38201	shootings, cultism and terrorism, kidnapping, riot, protest, etc.
Non-incident related	14841	190576	Not useful for emergency response

IV. EXPERIMENTS AND ANALYSIS

In this section, we describe the experimentally obtained results to demonstrate theeffectiveness of our proposed CNN model for tweet classification.

4.1 Dataset and Experimental Setup

A total of 383562 tweet text samples were collected using Twitter Streaming API and samples from human and infrastructural dataset [7]to form our dataset.We divided the dataset into two parts as shown in Table 1. Dataset 1 (DS-1) containing 29684 samples is a subset of this full dataset. Dataset 2 (DS-2), containing383562 text samples, is the full data set. Tables 2 and 3 show some sample tweets in the dataset.

Table 2: Sample tweets for the incident/non-incident classification

Tweet	***Label***
Two workers feared trapped as two-storey building collapses #BuildingCollapse	*incident*
Success is no accident. it is hard work, perservance, learning, studying sacrifice, and most of all love of what you are doing	*non-incident (false positive)*
if you are making the trip to #cheltenhamfestival .... #accident on m6 southbound between junctions 8 and	*incident*
#reno Truck trailer catches fire in Reno http://t.co/k5FIJaNkJb	*incident*
Dreaming of this after a looooong day of Black Friday shopping #HoteletteNashville (photo cred: @catherinetrumanphoto)	*non-incident*
Homecoming Queen Killed on Way Home from the Prom by Flood Waters! #socialnewshttp://t.co/VmKexjTyG4	*incident*
My first staining attempt was a disaster https://t.co/buDmKE3nNf	*non-incident*

Table 3: Sample tweets for the incident type classification

Tweet	***Label***
Two workers feared trapped as two-storey building collapses #Berekusu, #BuildingCollapse, #EasternRegionGhana	*Infrastrucuture damage*
if you are making the trip to #cheltenhamfestival .... #accident on m6 southbound between junctions 8 and 7	*accident*
At least 50 Dead and Several Others Injured as Suicide Bomber attacks Mosque in Adamawa State. Read more @ www.celestreet.com #SuicideBomber #Mosque #AdamawaState #Mubi #MubiNorthLGA #MadinaMosque #BokoHaram #SuicideBombing #Terrorism	*Crime and civil disorder*
Homecoming Queen Killed on Way Home from the Prom by Flood Waters! #socialnewshttp://t.co/VmKexjTyG4	*Flood*
#reno Truck trailer catches fire in Reno http://t.co/k5FIJaNkJb	*Fire*

The target value has two levels; the parent level for binary classification has two classes, $k_0 \in \{0, 1\}$ and which are $k_0 \in \{incident, non-incident\}$ and the child-level of the labels (for multi-class classification) has five classes, $k_\varphi \in \{0, 4\}$, which contains specific target emergency events belonging to the *incident* element of k_0. They are $k_\varphi \in \{accident, crime\ and\ civil\ disorder, damaged\ infrastructure, fire, flood\}$

In preprocessing the text data for training,we discarded non-English tweets, deleted all non-ASCII characters, hashtagsigns (#) fromthebeginningofallhashtagwords, and allthepunctuationmarks andnumbers. The stop words were removed, as they do not convey any meaningful information. Any unwanted multiple dots were removed, and multiple spaces were merged into one. We tokenized and converted text to lowercase.

Table 4: Binary Classification Performance Evaluation (%)

	SVM			$CNN_I^{(1)}$			$CNN_{II}^{(1)}$			$CNN_{III}^{(1)}$		
class	**P**	**R**	**F_1**	**P**	**R**	**F_1**	**P**	**R**	**F_1**	**P**	**R**	**F_1**
							DS-1					
Incident-related	78.02	74.11	76.01	78.89	75.02	76.91	82.34	84.02	83.17	87.43	83.9	85.63
Non-incident	82.08	86.34	84.16	83.89	86.45	85.15	85.09	80.87	82.93	82.78	84.21	83.49
Average	80.05	80.23	80.09	81.39	80.74	81.03	83.72	82.45	83.05	85.11	84.06	84.56
							DS-2					
Incident-related	67.22	71.02	69.07	76.65	83.34	79.86	81.09	80.02	80.55	83.12	89.43	86.16
Non-incident	76.83	75.65	76.24	75.43	71.45	73.39	79.91	80.44	80.17	84.01	84.12	84.06
Average	72.03	73.34	72.65	76.04	77.40	76.62	80.50	80.23	80.36	83.57	86.78	85.11

Table 5: Multi-class Classification Performance Evaluation (%)

	SVM			$CNN_I^{(2)}$			$CNN_{II}^{(2)}$			$CNN_{III}^{(2)}$		
class	**P**	**R**	**F_1**	**P**	**R**	**F_1**	**P**	**R**	**F_1**	**P**	**R**	**F_1**
							DS-1					
Accidents	71.45	60.33	65.42	76.9	68.01	72.18	74.34	77.98	76.12	76.32	76.54	76.43
Crimes and Civil Disorder	65.53	70.45	67.9	65.76	78.9	71.73	65.22	80.65	72.12	86.21	81.23	83.65
Damaged Infrastructure	67.34	72.56	69.85	68.54	69.87	69.2	75.33	72.65	73.97	78.34	87.23	82.55
Fire	70.34	72.43	71.37	73.12	74.32	73.72	79.45	81.03	80.23	72.34	83.89	77.69
Flood	73.11	67.34	70.11	67.76	65.43	66.57	74.07	78.1	76.03	84.32	85.9	85.10
Average	71.73	69.89	70.74	70.44	69.88	70.14	76.76	79.57	78.13	78.33	84.9	81.4
							DS-2					
Accidents	67.87	70.76	69.28	82.91	81.65	82.28	83.12	83.23	83.17	85.39	87.92	86.64
Crimes and Civil Disorder	64.84	71.32	67.93	81.32	83.45	82.37	85.74	84.83	85.28	87.34	86.28	86.81
Damaged Infrastructure	69.37	74.45	71.82	82.21	84.24	83.21	79.13	85.22	82.06	85.33	87.18	86.25
Fire	72.83	69.34	71.04	83.93	85.63	84.77	86.42	86.25	86.33	87.56	84.98	86.25
Flood	70.27	70.26	70.26	82.11	86.78	84.38	83.92	87.25	85.55	85.84	87.98	86.90
Average	71.55	69.8	70.65	83.02	86.21	84.58	85.17	86.75	85.94	86.7	86.48	86.57

We experimented with baseline and different CNN architectures as follows:

i) Baseline model: Support vector machine (SVM) trained using bag-of-words feature extractionand tested to validate the accuracy advantage of our models.

ii) CNN_I: A CNN model where all words are randomly initialized and then modified during training.

iii) CNN_{II}: A CNN model with pre-trained vectors from Glove for word embedding. All word-including the unknown ones that are randomly initialized – are kept static and only the other parameters of the model are learned.

iv) CNN_{III}: Same as in CNN_{II} but with parameters of the model being learned.

We used 80% of the dataset as the training set and 20% as the test set. The validation set is 20% of the training set.We trained a binary classifierand a multi-class classifier. For the binary classifier training, we merged all the incident types to create one general incident-related class. We trained CNN models for binary and multi-class classifications by optimizing the cross entropy in Equations 5 and 6 respectively using the Adam optimizer. We trained all the models for a maximum of 25 epochs. We experimented with $\{0.25, 0.5\}$ dropout rates, $\{16, 32\}$ mini-batch sizes and early stopping based on the accuracy on the validation set. We used rectified linear units (ReLU) for the activation function. In all experiments, the models (and hyper parameters) are tuned on the validation (development) set and the final performance evaluated against the test set. The word vectors in L were initialized with pre-trained 300-dimensional GloVe word embeddings. Words not present in the set of pre-trained words are initialized randomly.All the models are built and implemented using Python Programming language based on Keras deep learning library for Python running on tensorflow framework.

4.2 Evaluation Metrics:

In order to evaluate the performance of the classifiers, we calculate precision, recall, and F1-score.

$$Precision\ (P) = \frac{TP}{TP+FP} \qquad (9)$$

where *TP*is the number of true class samples that have been classified as true, and *FP*is thenumber of false class samples that have been classified as true.

$$Recall\ (R) = \frac{TP}{TP+FN} \qquad (10)$$

where *FN*is the number of true class samples that have been classified as false.

$$F1 - score = 2\,x\,\frac{Precision\ \ x\ Recall}{Precision\ +Recall} \tag{11}$$

4.3 Results and Discussions

In this section, we present the experimentally obtained results for the binary and multi-class classification tasks.

A. Binary Classification

We summarize the performance of the baseline and CNN models trained on DS-1 and DS-2 datasets in Table 4. The parameters are computed (at the class level) for each observation and then averaged. For each prediction task, we compared approaches that yielded the best absolute performance. The best CNN model performs better than the baseline model by a margin of 4.47% and 12.46% for DS-1 and DS-2 datasets respectively. However, CNN_I model with all randomized initialized words does not perform well on its own. For the last two models, CNN_{II} and CNN_{III} there were some gains, obviously due to the use of the pretrained vectors. There was however, no significant difference between CNN_{II} where words were kept static and CNN_{III} where pretrained vectors were fine tuned for each task in the DS-1. DS-2 produced about 3% improvement from CNN_{II} to CNN_{III}.

B. Multi-class Classification

Table 5 summarizes the performance of the baseline and CNN models for multi-class classification. Like in the case of binary classification, the parameters are computed (at the class level) for each observation and then averaged. Our CNN models performed better than the baseline model. The CNN models with all randomly initialized words (CNN_I) does not perform on DS-1 but with significant improvement in DS-2 dataset. The last two models, CNN_{II} to $CNN_{III,}$ with pre-trained vectors performed significantly well. The best model, CNN_{III}, performs better than the baseline model by 10.66% and 15.92% on DS-1 and DS-2 datasets respectively.

The final F1 score of the whole classification framework is 86.57% which is 5.17% over DS-1; an indication that the classifiers improve with increase in the size of data. The CNN_{III} model outperforms the other models in both data sets. CNN_{II} model performs second best for the two data sets. The overall experimental results demonstrate that pre-trained vectors are good feature extractors and the Deep learning models outperform conventional machine learning model like the Support Vector Machine (SVM).

V. CONCLUSION

In this paper, we presented a method of classifying tweets using convolutional neural networks trained on top of pre-trained word vectors that perform classification in layers, for binary and multi-class classification tasks, in order to (i) determine whether or not the tweet is incident-related; (ii) if a tweet is incident-related, then which category it belongs to.

This work proposed solutions to the challenges of incorporating social media data into emergency situations response; especially for detecting small scale emergency events where there are only small bits of information, thus complicating the detection of relevant information; how to filter-out noisy and irrelevant messages from big crisis data and categorization of the incident-related messages into different target classes of emergency event.

We tested our models using target incident-related real world home-grown dataset and the results demonstrated that pre-trained vectors are good feature extractors and the Deep learning models outperform conventional machine learning modelThe results showed that a system using the proposed method and architectures can classify tweets with an F1 score of 86.57%.

REFERENCES

[1]. Alsaedi, N., Burnap, P. and Rana, O. (2017). Can We Predict a Riot? Disruptive Event Detection Using Twitter. *ACM Transactions on Internet Technology*, 17(2), 18.

[2]. De Smet, H., LeysenJ andLagadee, P. (2011). The Response Phase of the Disaster Life Cycle Revisited. *Proceedings of the 2011 Industrial Engineering Research Conference.*

[3]. Kim, Y. (2014). Convolutional Neural Networks for Sentence Classification. *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, 1746–1751. arXiv preprint arXiv:1408.5882.

[4]. Kowsari, K., Brown, D.E., Heidarysafa, M., Meimandi, K.J., Gerber, M.S., Barnes, L.E. (2017). HDLTex: Hierarchical Deep Learning for Text Classification. *In Proceedings of the 2017 16th IEEE International Conference on Machine Learning and Applications (ICMLA)*, Cancun, Mexico. 0-134, doi:10.1109/icmla.

[5]. Lagerstrom, R., Arzhaeva, Y., Szul, P., Obst, O., Power, R., Robinson, B., and Bednarz, T. (2016).Image classification to Support Emergency Situation Awareness. *Frontiers in Robotics and AI*. Retrieved August 20, 2018, from https://doi.org/10.3389/frobt.2016.00054

[6]. LeCun, Y., Bengio, Y and Hinton, G. (2015). Deep learning, *Nature*, 521(7553),436–444

[7]. Mouzannar, H., Rizk, Y. and Awad, M. (2018). Damage Identification in Social Media Posts using Multimodal Deep Learning. *Proceedings of the 15th International Conference on Information Systems for Crisis Response and Management (ISCRAM)*, Rochester, USA, 529-543.

[8]. Nguyen, D.T., Joty, S., Imran, M., Sajjad, H. and Mitra, P. (2016) Applications of Online Deep Learning for Crisis Response Using Social Media Information. *4th international workshop on Social Web for Disaster Management (SWDM)*. Retrieved September 11, 2018, from https://arxiv.org/pdf/1610.01030.pdf

[9]. Pennington, J., Socher, R. and Manning, C.D. (2014). Glove: global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*

[10]. Ramchurn, S.D., Trung Dong Huynh, T.D.,Feng Wu, F., Yuki Ikuno, Y., Flann, J., Moreau, L., Fischer, J.E., Jiang, W., Rodden, T., Simpson, E., Reece, S., Roberts, S., Jennings, N. R. (2016). A Disaster Response System based on Human-Agent Collectives. *Journal of Artificial Intelligence Research*, 57, 661-708.

[11]. Sakaki, T., Okazaki, M. and Matsuo, Y. (2010). Earthquake shakes Twitter users: real-time event detection by social sensors. In *Proc. of WWW*, ACM. 851–860.

[12]. Sakaki, T., Okazaki, M., and Matsuo, Y. (2013). Tweet analysis for real-time event detection and earthquake reporting system development. *IEEE Trans. Knowl. Data Eng.* 25, 919–931. doi:10.1109/TKDE.2012.29

[13]. Schulz, A., Mencia, E. L., and Schmidt, B. (2016). A Rapid-Prototyping Framework for Extracting Small-Scale Incident-Related Information in Microblogs: Application of Multi-Label Classification on Tweets. Information Systems, 57, 88–110. DOI: 10.1016/j.is.2015.10.010.

[14]. Starbird, K., Palen, L. Hughes, A.L. and Vieweg, S. (2010). Chatter on the red: what hazards threat reveals about the social life of microblogged information. *Proceedings of CSCW.* ACM, 241–250.

[15]. Vieweg, S., Castillo, C. and Imran, M. (2014). Integrating social media communications into the rapid assessment of sudden onset disasters. *In Social Informatics*, Springer, 444–461.

[16]. Vieweg, S., Hughes, A. L, Starbird, K. and Palen, L. (2010). Microblogging during two natural hazards events: what twitter may contribute to situational awareness. *Proceedings of the SIGCHI conference on human factors in computing systems*. 1079-1088. https://doi.org/10.1145/1753326.1753486

[17]. Yang, Y., Ha, H.-Y., Fleites, F., Chen, S.-C., and Luis, S. (2011). Hierarchical disaster image classification for situation report enhancement. *Proceedings of theIEEE 12th International Conference on Information Reuse and Integration* (Las Vegas, Nevada).