A Deep Learning Model for Event Extraction and Classification in Hindi for Disaster Domain

Zishan Ahmad
Dept. of Computer Sc.and Engg.
Indian Institue of Technology Patna
Patna, Bihar-801106, India
1821cs18@iitp.ac.in

Sovan Kumar Sahoo
Dept. of Computer Sc.and Engg.
Indian Institue of Technology Patna
Patna, Bihar-801106, India
sovan.pcs17@iitp.ac.in

Asif Ekbal
Dept. of Computer Sc.and Engg.
Indian Institue of Technology Patna
Patna, Bihar-801106, India
asif@iitp.ac.in

Pushpak Bhattacharyya
Dept. of Computer Sc.and Engg.
Indian Institue of Technology Patna
Patna, Bihar-801106, India
pb@iitp.ac.in

Abstract

Relevant information extraction in disaster situation plays an important role for crisis management. In this paper we propose a deep learning based method for event extraction in Hindi from the man-made and natural disaster related texts. The overall task is to identify the event triggers from text and then classify them into predefined categories of interest. Our proposed model follows an ensemble architecture where we use Convolutional Neural Network (CNN) and Bidirectional Long Short-Term Memory (Bi-LSTM) network as the base learning models. We crawl the data from various newswire sources, define the event types and its annotation guidelines, annotate the datasets and then create a benchmark setup for event extraction. Experiments on 5-fold cross-validation with approximately 80K token datasets show the macro average and micro average F1-scores of 0.40 and 0.59, respectively for event trigger detection and classification.

1 Introduction

Event extraction is a very important task in Natural Language Processing. Event is a basic unit of knowledge representation, and has been drawing growing attention of the researchers and practitioners as an effective mean for information organization. Event is an occurrence that happens at a particular place and at a particular time or time interval. Different agencies and individuals introduce a large amount of data in the web related to any particular event, by publishing news reports in platforms like personal blogs, web sites and news portals. The amount of data is tremendous and hence retrieving relevant information through manual process is infeasible. Hence, there is a necessity to build robust systems that would be able to mine the desired information in an automated way. Extracting event triggers, classifying them into a predefined set of categories and then associating arguments to these events plays an important role in building an information extraction system. In some of the domains such as disaster, extracting relevant information in appropriate time is very important, in order to alert both, the public and the government. Armed with this effective information, post disaster management activities can be carried out because it not only seeks public attention but also the attention of government as well as non-government agencies which are the key players in post disaster management. Some of the existing works focusing on English language are reported in (Nugent et al., 2017; Dittrich and Lucas, 2014; Yun, 2011; Klein et al., 2013; Burel et al., 2017; Tanev et al., 2008). In contrast, there has not been any significant attempt to build event extraction system in Indian languages. In recent times, some of the works as reported in (Kuila and Sarkar, 2017; Singh et al., 2017) focus event extraction in Indian languages such as Hindi, Tamil and Malayalam. These were mostly focused on extracting events from disaster re-

Proc. of ICON-2018, Patiala, India. December 2018 ©*2018 NLPAI*, pages 122–131

lated tweets. Where as the first paper used deep learning techniques to solve the problem, later one used classical machine learning techniques such as Support Vector Machine (SVM), Gradient Boosting and Random Forest.

In our current work we propose a deep neural network based model for event extraction that operates in two steps, *viz.* identification of event triggers from text and then classifying them into a set of predefined categories of interest. Our focus is on Hindi language.

1.1 Problem Definition and Contributions

Given a Hindi sentence of the form w_1,w_2,w_3,....w_n, the task is to (i). identifying event triggers from text and (ii). classifying event triggers into a set of predefined categories.

(i) Predict the sequence of labels of the form l_1,l_2,l_3,....l_n, where each label l_i corresponds to w_i and $l_i \in$ I,O,B. The tags I,O and B indicate the inside, outside and beginning of entity of an event [1]. The example mentioned below depicts the input sentence and output label sequence. Event triggers are boldfaced in input sentence example.

- **Input Hindi Sentence:** गृह मंत्रालय मुंबई के **बम विस्फोटों** के मद्देनजर इस बात की विशेष तौर पर जांच कर रहा है कि अक्षरधाम मंदिर और १९९३ के मुंबई **बम विस्फोटों** के फैसलों की प्रतिक्रिया के रूप में तो यह **हमले** नहीं हुए

- **Transliteration:** grih mantraalay mumbai ke **bam visphoton** ke maddenajar is baat kee vishesh taur par jaanch kar raha hai ki aksharadhaam mandir aur 1993 ke mumbai **bam visphoton** ke phaisalon kee pratikriya ke roop mein to yah **hamale** nahin hue

- **Gloss:** home/ ministry/ mumbai/ of/ **bomb**/ **blasts**/ of/ `in_wake_of`/ this/ talk/ of/ special/ modus/ on /investigation /do /stay /is / that /akshardhaam /temple /and /1993 /of /mumbai /**bomb** /**blasts** /of /decisions /of /reaction/ of /form / in/ so/ this/ **attack**/ not/ happened

- **Translation:** In view of the Mumbai **bomb blasts**, the Home Ministry is specially investigating the fact that these **attacks** did not take place as response to the Akshardham Temple and the 1993 Bombay **bomb blasts**.

- **Output:** O O O O I I O O O O O O O O O O O O O O O O O O O O O O I I O O O O O O O O O I O O

(ii) Classify the detected event triggers into predefined event types. For example, in the above Hindi input sentence the boldfaced event triggers belong to `Terrorist_Attack` type.

The key contributions of our proposed work lies in the following:

- Building a deep learning based event extraction system in Hindi for disaster domain. Our proposed model is an ensemble of both Convolution Neural Network (CNN) (Kim, 2014) and Bidirectional Long Short-Term Memory (Bi-LSTM) (Schuster and Paliwal, 1997).

- Creating a benchmark setup for event extraction in Hindi. This may be used as a baseline model for further research towards this direction.

2 Related Works

Event extraction is a well-known problem in Natural Language Processing (NLP). Both the feature based as well as neural network based approaches have been used to solve this problem. Some of the feature based approaches framed the entire event extraction task into two subtasks, and solve each subtask separately (Ji and Grishman, 2008; Liao and Grishman, 2010; Hong et al., 2011). The main disadvantage of this approach is the error propagation. To overcome this problem (Li et al., 2013) proposed a joint event extraction algorithm which predicts both event triggers and its arguments simultaneously. In (Yang and Mitchell, 2016) both the approach are used i.e

[1]The encoding scheme is according to IOB2, where I indicates the tokens that appear within trigger, B denotes the beginning of a trigger and O denotes the outside of an event trigger. The B is used only when two events of the same type appear in consecutive sequence

extracting event and entities jointly and that also in document level. Chen et al. (Chen et al., 2015) introduced a convolutional neural network (CNN) based word representation model to capture meaningful semantic regularities for words. CNN captures only the most important information in a sentence but may miss other valuable information. So for multiple-event sentences, they proposed a dynamic multi-pooling CNN (DMCNN). DMCNN uses a dynamic multi-pooling layer according to event triggers and arguments, to reserve more crucial information. They reported that their system significantly outperforms other state-of-the-art systems.

Event extraction task has also been addressed in specialized tracks dedicated in the Text Analysis Conference (TAC). Many research groups participated in the workshop and submitted their works covering various methods (classical supervised and deep learning models). We present here a brief survey of the works that focused on deep learning based methods. Frank et. al. submitted a system at TAC KBP 2016 (Mihaylov and Frank, 2016). They built a neural architecture for event trigger detection, event type classification and event realis classification. They used a Bidirectional Long Short-Term Memory (Bi-LSTM) based system for detecting event triggers from text. They carried out experiments with various configurations including using Part-of-Speech (PoS) and dependency label embeddings as additional information into deep neural network. They also performed experiments with short-cuts to the output layer and reported improvement in the performance. The work reported in (Dubbin et al., 2016) implemented two variants of event detection systems. Their first system used manually created rules and a set of a very rich linguistic resources whereas their second system used deep neural networks.

Event extraction in disaster situation is very crucial as it assists in supplying relevant information to the affected people and the various other stakeholders including the government agencies. A real-time news event extraction system was developed by Joint Research Center of the European Commission (Tanev et al., 2008). The developed system could extract violent and disaster events accurately from on-line news though their system was linguistically lightweight. Another study (Yun, 2011) described a rapid event detection system of disaster events and showed how to detect a target event from tweets. They used features like location, time, keywords and frequency of keywords in tweets. A prototype of real-time multilingual natural disaster identification and monitoring system based on Twitter was introduced by (Dittrich and Lucas, 2014). In (Nugent et al., 2017) the authors compared a set of supervised learning methods to event type classification on English news data. Majority of the works have been carried out mostly in English and some other resource-rich languages. In contrast there has been a very little works on event extraction on resource-poor languages such as the Indian ones. A deep learning based system was built by (Kuila and Sarkar, 2017) for event extraction in Indian languages like Hindi, Tamil and Malayalam. The system was built for handling tweets. Event classification and location prediction was done in (Singh et al., 2017) using traditional machine learning techniques. Event extraction in Tamil language was reported in (SharmilaDevi et al., 2017). They extracted features and classified each word or chunk into event and non-event class using SVM. In another work (Sristy et al., 2017) formulated the event extraction task as sequence labeling problem and used Conditional Random Fields(CRF) to extract events.

It is seen that most of the previous works used tweets for event extraction and classification but our focus is on more structured newswire data, which are collected from different on-line news portals in Hindi language. We follow a pipelined approach where event detection and event classification are tackled in two consecutive steps. Each of the modules is based on an ensemble of Convolutional Neural Network (CNN) and Bidirectional Long Short-Term Memory (Bi-LSTM).

3 Methodology

Our overall task consists of detecting the trigger words and phrases from a given sentence, and classifying these into different types. The schematic diagram of the complete system is

shown in the Figure 1.

We use these models in a pipeline to get the final output. Each sentence is passed through the 'Trigger Detection Model', the task of which is to detect trigger expressions denoting the events. Trigger refers to the textual content that corresponds to the tokens denoting the events related information. The detected triggers are then passed through 'Trigger Classification Model' which classifies the triggers into different event types.

3.1 Word Embedding Representation

The proposed system uses word-embedding vectors of size 300 to represent the words as input to the models. The word embeddings are trained using word2vec algorithm (Mikolov et al., 2013) on Hindi Wikipedia dump. The size of corpus is 323M and vocabulary size is 30,393 tokens. However the word-embeddings are pre-trained and downloaded from github page[2]. To represent Part-of-Speech (PoS) tags, we use one-hot encoding representation.

3.2 Trigger Detection Model

We formulate the task of event trigger detection as a sequence labeling problem, i.e. for each token in the sentence we need to decide whether it denotes an event expression or not. Our proposed model is ensemble in nature where the base models are Bi-LSTM and CNN (c.f. Figure 2).

The word embedding of each word is passed through a Bi-LSTM model, and we obtain an output representation for each word. The word embedding of each word is passed through CNN and convoluted features are obtained. The one-hot vector representation of PoS tag of each word is also passed through a separate Bi-LSTM. We obtain the PoS information from the Hindi Shallow Parser [3]. We obtain an output representation of each PoS tag. Finally, all these output representations are concatenated. This concatenated feature is then passed through a Multi Layer Perceptron (MLP) followed by a Softmax layer which computes the probability distribution over the possible tags of I_Event_Trigger or O_Event_Trigger [4].

[2]http://github.com/Kyubyong/wordvectors

[3]http://ltrc.iiit.ac.in/analyzer/hindi/

[4]Here I and O denote the intermediate tokens of an event expression and outside token, respectively.

3.3 Trigger Classification Model

The input to this event trigger classification model is the trigger expression or phrase and the output is a possible class to be assigned to that particular trigger. Our current work is a multi-class classification problem that classifies each trigger into seventeen possible classes. Similar to trigger detection, we also use here an ensemble network model consisting of Bi-LSTM and CNN. The overall schematic diagram of the classification model is depicted in Figure 3. We pad each trigger phrase by the placeholder '<pad>', to make each phrase of equal length. Word embedding of each word is passed through a CNN to obtain the convoluted feature representation. These convoluted features are flattened and concatenated with Bi-LSTM representation. This concatenated feature is then passed through a Multi Layer Perceptron model followed by a Softmax layer, which is used to obtain the probability distribution over a set of seventeen classes.

4 Datasets and Experiments

In this section, we provide the description of the datasets, experimental setup, results and provide necessary analysis of the results.

4.1 Datasets

The news data are crawled from several Hindi news portals. In the dataset there are 253 news documents consisting of 4,403 sentences. In total 80,136 words are present among which 1,179 words are trigger words. All the news documents are annotated by three annotators who are from linguistic background and having sufficient knowledge of the related area, particularly the TAC-KBP event and entity annotation guidelines. In order to measure the inter-annotator agreement ratio, we asked three annotators to annotate 5% of total documents. We found the multi-rater Kappa agreement ratio of 0.85.

The news data crawled, belong to disaster domain. Seventeen disaster event types including both man-made and natural disaster. The event types are: *Terrorist_Attack, Storm, Cyclone, Normal_Bombing, Earthquake, Transport_Hazard, Floods, Land_Slide,*

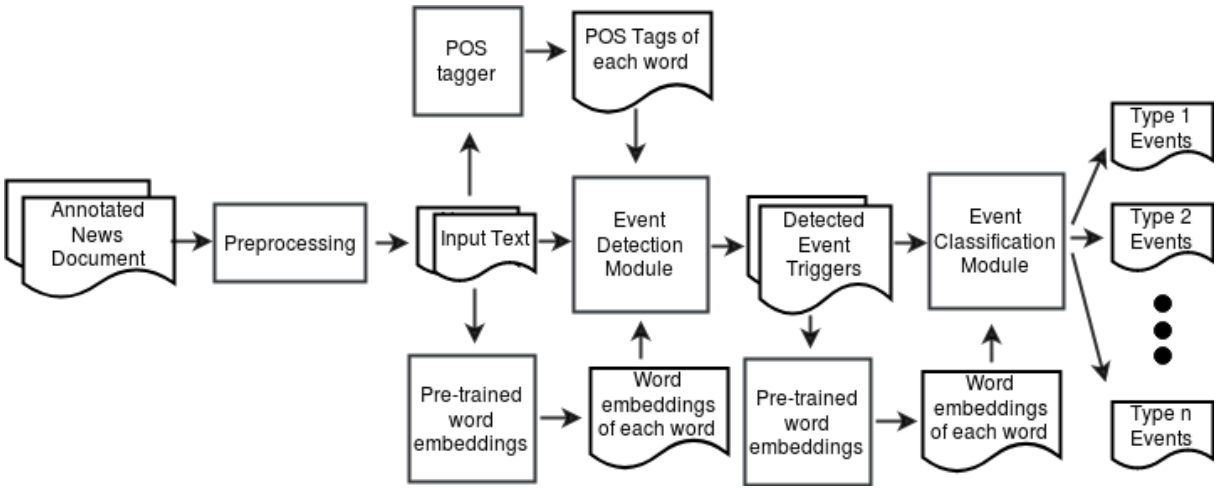

Figure 1: Schematic diagram of the overall system

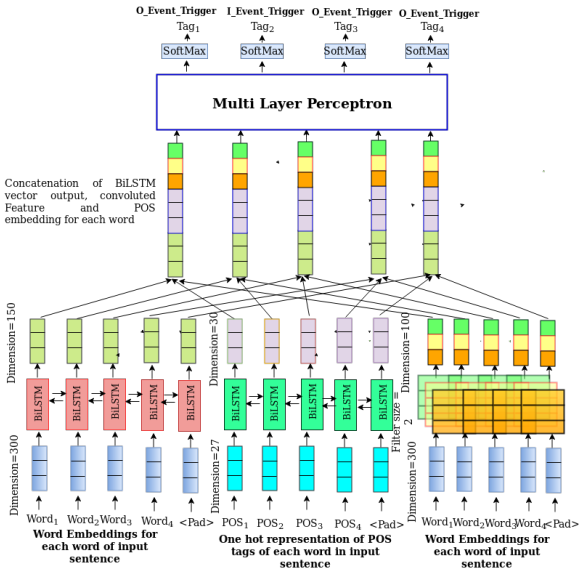

Figure 2: Trigger Detection Model

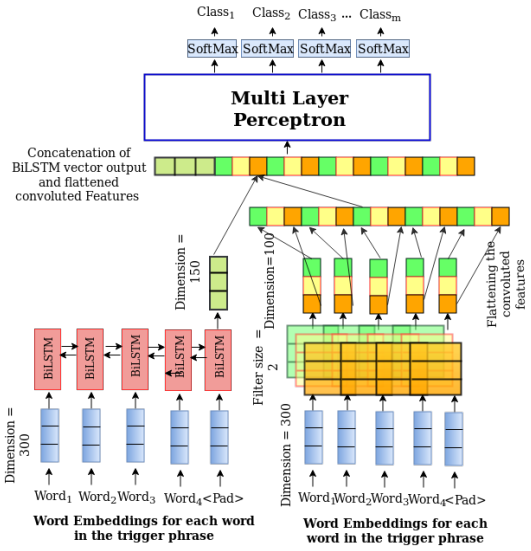

Figure 3: Trigger Classification Model

Train_Collision, Armed_Conflict, Hurricane, Shoot_Out, Riots, Aviation_Hazard, Hail_Storm, Tsunami, Surgical_Strike.

While annotating the event triggers, the above disaster types are used as classes, and each event trigger is associated with one of the seventeen classes. Thus our data contains the event information as well as the fine-grained class information of the event.

4.2 Experimental Setups

For implementing the deep learning models a Python based library Keras [5] is used. We use the IOB format (Ramshaw and Marcus, 1999) [6].

The trigger detection model (c.f. Figure 2) has word-embedding vectors of size 300, and one hot encoding of PoS tags of size 27 as inputs to the Bi-LSTM and CNN layers. The Bi-LSTM layer with word embeddings as input has 150 neurons and the Bi-LSTM layer with encoded PoS tags as input has 20 neurons. The CNN layer has 100 filters sliding over 2 words at a time. *'Relu'* is used as an activation function for CNN and *'dropout'* of 30% is used between layers for regularization. The multi-layer perceptron comprises of 150 and 75 neurons in the first and second layer respec-

[5] http://keras.io

[6] B, I and O denote the beginning, intermediate and outside of an event trigger

tively. *'Softmax'* is used in the final layer for classification of trigger and non-trigger words.

The input to the trigger classification model (c.f. Figure 3) is also word-embedding vectors of size 300. Similar to trigger detection model, Bi-LSTM layer has 150 neurons and CNN layer has 100 filters sliding over 2 words at a time. The *'dropout'* and activation functions used are the same as in the trigger detection model. The multi-layer perceptron comprises of 600 and 100 neurons in the first and second layer respectively and *'Softmax'* is used in the final layer for classification of 17 event classes.

Training is done using a learning rate of 0.001 and *'Adam'* optimizer is used for fast convergence. The data is fed to the neural network in batches of 32. *'Checkpoints'* are used to save the best weights of the model based on training accuracy.

For evaluation we use *'Precision'*, *'Recall'* and *'F1-Score'* as the metrics for event trigger detection. We further use micro and macro averaging of the *precision*,*recall* and *F1-score* of each class for evaluating event classification model. Macro-average computes the metrics independently for each class and then takes the average (hence treating all the classes equally), whereas micro-average aggregates the contributions of all classes to compute the average metric. Thus in case of class imbalance micro-average is a more preferred criteria.

5 Results and Analysis

In this section we present the details of experiments and the results that we obtain along with necessary analysis.

5.1 Trigger Detection Model

For trigger detection we develop three different variations of event trigger detection. First model is based on Bi-LSTM. The word embedding of each word in the sentence is concatenated with one hot vector of its PoS tag. This is then passed through a Bi-LSTM. The output representation of the Bi-LSTM is classified into event and non-event by Softmax at the output layer.

The second model is based on CNN. The input to the CNN is same as that of Bi-LSTM. 100 filters were used to obtain features with a kernel size of 2. The convoluted features, thus obtained, are classified using Softmax at the output layer.

The third model is described in Section 3.2 (Figure 2). We also re-implement the system of event extraction proposed in (Kuila and Sarkar, 2017) and evaluate it on our dataset. Their system uses two back to back CNNs and then a Bi-LSTM in sequence.We demonstrate the results of 5-Fold cross-validation in Table 1.

	Precision	Recall	F1-Score
Bi-LSTM	0.74	0.71	0.72
CNN	0.70	0.69	0.69
Ensemble Model	0.74	0.75	0.74
Kuila et. al, 2017	0.68	0.75	0.71

Table 1: Results of different event trigger detection models: 5-fold cross-validation

From the results in Table 1 it can clearly be seen that the ensemble model performs better than the individual Bi-LSTM and CNN models. The Bi-LSTM model treats the problem as a sequence to sequence labeling task and it tries to classify the current word by looking at the context from both the directions as well as the current input word. The CNN model has a kernel size of 2, i.e. it tries to extract bi-gram features that are relevant in classifying a word into trigger or non-trigger. These two approaches are combined through an ensemble model that exploits both the contextual information from Bi-LSTM and the bi-gram feature extracted from CNN to make prediction. It is able to make use of the strength of both the models and optimize it to solve the problem.

5.2 Trigger Classification Model

For trigger classification we again build three different models. Similar to the trigger detection model we use Bi-LSTM, CNN and the ensemble model (c.f. Figure 3) for trigger classification. We obtain the results in terms of *macro-averaging* and *micro-averaging* of *precision*, *recall* and *F1-score* of each class.

Evaluation results of 5-Fold cross-validation are shown in Table 2. Results show that the ensemble model performs better than the individual models.

We present the detailed class-wise evaluation results in Table 3 for the ensemble model.

	$Macro_p$	$Macro_r$	$Macro_{f1}$
Bi-LSTM	0.40	0.45	0.42
CNN	0.42	0.44	0.43
Ensemble Model	0.45	0.46	0.45
	$Micro_p$	$Micro_r$	$Micro_{f1}$
Bi-LSTM	0.69	0.69	0.69
CNN	0.70	0.70	0.70
Ensemble Model	0.72	0.72	0.72

Table 2: Evaluation results of 5-fold cross-validation for trigger classification. Here, Macro-averaged Precision: $Macro_p$, Macro-averaged Recall: $Macro_r$, Macro-averaged F1-Score: $Macro_{f1}$, Micro-averaged Precision:$Micro_p$, Micro-averaged Recall: $Micro_r$, Micro-averaged F1-Score: $Micro_{f1}$

For the classes for which the performance are very less are due to the less number of instances for the respective class.

Class	Precision	Recall	F1-Score
Terrorist_Attack	0.82	0.95	0.88
Storm	0.72	0.97	0.80
Cyclone	0.67	0.32	0.43
Normal_Bombing	0.20	0.22	0.21
Earthquake	0.44	1.00	0.62
Transport_Hazard	1.00	0.33	0.50
Floods	0.70	1.00	0.82
Land_Slide	1.00	1.00	1.00
Train_Collision	0.02	0.02	0.02
Armed_Conflict	0.01	0.01	0.01
Hurricane	0.00	0.00	0.00
Shoot_Out	0.70	0.95	0.72
Riots	0.40	0.15	0.21
Aviation_Hazard	0.33	0.18	0.23
Hail_Storm	0.02	0.20	0.04
Tsunami	0.20	0.20	0.20
Surgical_Strike	0.20	0.20	0.20

Table 3: Evaluation results of class-wise trigger classification: 5-fold cross-validation. Here we report macro scores.

5.3 Pipelined Model for Trigger Detection and Classification

The final system (c.f Figure 1) uses both event trigger detection model (c.f. Figure 2) and event classification model (c.f. Figure 3). Each document is divided into sentences, and each sentence is represented by the sequence of word-embeddings of its words. This sequence is passed through the trigger detection model and triggers are obtained. These trigger phrases are again represented as a sequence of word-embeddings of its words. The trigger expressions so detected are passed through the trigger classification model, and the appropriate classes of triggers are obtained.

The macro and micro-averaged results of 5-fold cross-validation are shown in Table 4.

	Precision	Recall	F1-Score
Macro-Averaged	0.35	0.48	0.40
Micro-Averaged	0.59	0.59	0.59

Table 4: Evaluation results of 5-fold cross-validation

From Table 4 it can be clearly seen that the performance in this pipelined model is less compared to what are reported in the earlier section. This is because the errors encountered in the trigger detection model are propagated to the next stage, i.e. event classification model. We also show the class-wise evaluation results obtained through 5-fold cross-validation in Table 5.

Since a lot of triggers are not detected by the trigger detection model (i.e. false negative cases), these triggers are not classified at all. A number of false positives in the trigger detection phase also contributes to the overall classification error. These two factors causes a degradation of the overall performance.

Class	Precision	Recall	F1-Score
Terrorist_Attack	0.82	0.97	0.88
Storm	0.40	0.80	0.53
Cyclone	0.51	0.32	0.39
Normal_Bombing	0.77	0.35	0.49
Earthquake	0.63	0.20	0.4
Transport_Hazard	0.89	0.22	0.35
Floods	0.50	0.20	0.28
Land_Slide	0.67	0.32	0.60
Train_Collision	0.20	0.26	0.22
Armed_Conflict	0.01	0.01	0.01
Hurricane	0.00	0.00	0.00
Shoot_Out	0.20	0.61	0.30
Riots	0.36	0.20	0.25
Aviation_Hazard	0.30	0.18	0.22
Hail_Storm	0.02	0.20	0.04
Tsunami	0.1	0.1	0.1
Surgical_Strike	0.2	0.2	0.2

Table 5: Evaluation results of 5-fold cross-validation for pipelined model. Here we report macro scores.

5.4 Error Analysis

We analyze the errors for both trigger detection and trigger classification models. For error analysis we split the data into training set (80% of data) and testing set (20% of data). The training set contains 3,522 sentences and the testing set contains 881 sentences.

5.4.1 Trigger detection model

Out of total triggers of 323 in the test data, 255 triggers are captured by our trigger detection system. Few examples of the errors caused by our system are as follows:

1. The system sometimes confuses between an actual event and a hypothetical event.

   (a) **Sentence-1**: छत्तीसगढ़ में **नक्सली हमले** में 3 जवान शहीद
   Transliteration: Chhatteesagadh mein **naksalee hamale** mein 3 javaan shaheed
   Translation: 3 soldiers killed in **Naxal attack** in Chhattisgarh

   (b) **Sentence-2**: इन आतंकवादियों की दिल्ली में **हमले** की योजना थी
   Transliteration: In aatankavaadiyon kee dillee mein **hamale** kee yojana thee
   Translation: These terrorists had plans to **attack** Delhi

   In the above example, **Sentence-1** talks about a real event but **Sentence-2** talks about a hypothetical event. However, the system is unable to detect the real event in **Sentence-1** but detects the hypothetical event in **Sentence-2**.

2. Long phrases used to describe an event are being confused by the system.

   (a) **Sentence-3** अलबामा में ट्रांसफॅर्मरों **में बारिश का पानी भर गया**
   Transliteration: Alabaama mein traansaphairmaron **mein baarish ka paanee bhar gaya**
   Translation: **Rain water was flooded** in transformers in Alabama

   (b) **Sentence-4** निचले इलाके में मौजूद एक व्यक्ति क्रिस रॅबिंसन ने फोन से बताया कि **पानी तेजी से भर रहा है**
   Transliteration: nichale ilaake mein maujood ek vyakti kris raibinsan ne phon se bataaya ki **paanee tejee se bhar raha hai**
   Translation: Chris Robinson, a person in the lower area, told on the phone that **water is filling up fast**.

   In **Sentence-3** the phrase में बारिश का पानी भर गया describes 'flooding', but this is not captured by the system. However, in **Sentence-4**, the phrase पानी तेजी से भर रहा है is not used for an event, but the system wrongly captures it as an event. This is because both these phrases are very close in meaning, and the context for proper disambiguation is lacking.

3. Another major cause of error is the class-imbalance problem. Out of a total 1,179 trigger words, 543 are of class *Terrorist_Attack*. On the other hand there are only 5 trigger words of type *Tsunami*, 7 triggers of type *Hail_Storm*, 9 triggers of type *Aviation_Hazard* and *Riots*, 14 triggers of type *Shoot_Out* and 15 triggers of type *Hurricane*. Uneven class distribution influences to the overall error. Balancing these classes may reduce such errors.

5.4.2 Trigger classification model

The results of event classification model are discussed in Section 5.2. The confusion matrix after classification (by ensemble model) on the test data can be seen in Figure 4.

	None	Terrorist_Attack	Storm	Cyclone	Normal_Bombing	Earthquake	Transport_Hazard	Floods	Land_Slide	Train_Collision	Armed_Conflict	Hurricane	Shoot_Out	Riots	Aviation_Hazard	Hail_Storm	Tsunami	Surgical_Strike
None	0	46	29	3	9	5	0	2	0	0	1	0	1	0	0	1	0	0
Terrorist_Attack	0	141	0	1	5	0	0	0	0	0	1	0	0	0	0	0	0	0
Storm	0	0	39	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Cyclone	0	0	21	10	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Normal_Bombing	0	7	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0
Earthquake	0	0	0	0	0	4	0	0	0	0	0	0	0	0	0	0	0	0
Transport_Hazard	0	0	0	0	2	0	1	0	0	0	0	0	0	0	0	0	0	0
Floods	0	0	0	0	0	0	0	3	0	0	0	0	0	0	0	0	0	0
Land_Slide	0	0	0	0	0	0	0	0	7	0	0	0	0	0	0	0	0	0
Train_Collision	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
Armed_Conflict	0	3	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0
Hurricane	0	0	4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Shoot_Out	0	0	0	0	0	0	0	0	0	0	0	0	3	0	0	0	0	0
Riots	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0
Aviation_Hazard	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
Hail_Storm	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Tsunami	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
Surgical_Strike	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 4: Confusion matrix of classification by event classification model (ensemble model)

From the confusion matrix it can be seen that the system is confused between the types that are very close to each other. *Terrorist_Attack* is being confused with *Normal_Bombing* and *Normal_Bombing* is being confused with *Terrorist_Attack*. We can also see that *Storm* is being confused with *Cyclone*, *Cyclone* is being confused with *Storm* while *Hurricane* is being classified to either *Storm* and *Cyclone*. This is because the events *Storm*, *Cyclone* and *Hurricane* are very close in nature. Also the Hindi word used for all the three is 'तूफान', which further confuses the system.

6 Conclusion and Future Works

In this paper we have presented a hybrid deep learning approach based on Bi-LSTM and CNN for event trigger detection and classification. As there was no readily available data, we have collected Hindi documents from the various web sources, annotated data with event triggers and a predefined set of categories. Preliminary evaluation shows promising results. Experiments on these datasets show that our proposed ensemble model performs better compared to the individual model. This can be used as a benchmark setup for further research and development in the related areas.pr We have also performed detailed analysis to understand the shortcoming of our proposed system. To fix these errors we would like to explore ways in which context could be added to the input. This will help the system in making better informed decision for trigger detection. To mitigate the class imbalance problem, we would like to crawl enough documents relevant to the specific classes that suffer from the data sparsity problem. Another future direction would be to develop a stacked based classifier with Bi-LSTM as a base followed by Conditional Random Field (CRF).

7 Acknowledgement

The research reported here is an outcome of the project titled "A Platform for Cross-lingual and Multi-lingual Event Monitoring in Indian Languages", supported by IMPRINT-1, MHRD, Govt. of India, and MeiTY, Govt. of India.

References

Grégoire Burel, Hassan Saif, Miriam Fernandez, and Harith Alani. 2017. On semantics and deep learning for event detection in crisis situations. *Workshop on Semantic Deep Learning (SemDeep), at ESWC 2017, 29 May 2017, Portoroz, Slovenia*.

Yubo Chen, Liheng Xu, Kang Liu, Daojian Zeng, and Jun Zhao. 2015. Event extraction via dynamic multi-pooling convolutional neural networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, volume 1, pages 167–176.

André Dittrich and Christian Lucas. 2014. Is this twitter event a disaster? *AGILE Digital Editions*.

Greg Dubbin, Archna Bhatia, Bonnie J Dorr, Adam Dalton, Kristy Hollingshead, Suriya Kandaswamy, Ian Perera, and Jena D Hwang. 2016. Improving discern with deep learning. In *TAC*.

Yu Hong, Jianfeng Zhang, Bin Ma, Jianmin Yao, Guodong Zhou, and Qiaoming Zhu. 2011. Using cross-entity inference to improve event extraction. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 1127–1136. Association for Computational Linguistics.

Heng Ji and Ralph Grishman. 2008. Refining event extraction through cross-document inference. *Proceedings of ACL-08: HLT*, pages 254–262.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*.

Bernhard Klein, Federico Castanedo, Inigo Elejalde, Diego López-de Ipina, and Alejandro Prada Nespral. 2013. Emergency event detection in twitter streams based on natural language processing. In *International Conference on Ubiquitous Computing and Ambient Intelligence*, pages 239–246. Springer.

Alapan Kuila and Sudeshna Sarkar. 2017. An event extraction system via neural networks. *FIRE (Working Notes)*, pages 136–139.

Qi Li, Heng Ji, and Liang Huang. 2013. Joint event extraction via structured prediction with global features. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 73–82.

Shasha Liao and Ralph Grishman. 2010. Using document level cross-event inference to improve event extraction. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 789–797. Association for Computational Linguistics.

Todor Mihaylov and Anette Frank. 2016. Aipheshd system at tac kbp 2016: Neural event trigger detection and event type and realis disambiguation with word embeddings. *Proceedings of the TAC Knowledge Base Population (KBP)*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Tim Nugent, Fabio Petroni, Natraj Raman, Lucas Carstens, and Jochen L Leidner. 2017. A comparison of classification models for natural disaster and critical event detection from news. In *Big Data (Big Data), 2017 IEEE International Conference on*, pages 3750–3759. IEEE.

Lance A Ramshaw and Mitchell P Marcus. 1999. Text chunking using transformation-based learning. In *Natural language processing using very large corpora*, pages 157–176. Springer.

Mike Schuster and Kuldip K Paliwal. 1997. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681.

V SharmilaDevi, S Kannimuthu, and G Safeeq. 2017. Kce_dalab@ eventxtract-il-fire2017: Event extraction using support vector machines. *FIRE (Working Notes)*, pages 144–146.

Jyoti Prakash Singh, Yogesh K Dwivedi, Nripendra P Rana, Abhinav Kumar, and Kawaljeet Kaur Kapoor. 2017. Event classification and location prediction from tweets during disasters. *Annals of Operations Research*, pages 1–21.

Nagesh Bhattu Sristy, N Satya Krishna, and Durvasula VLN Somayajulu. 2017. Event extraction from social media text using conditional random fields. In *FIRE (Working Notes)*, pages 140–143.

Hristo Tanev, Jakub Piskorski, and Martin Atkinson. 2008. Real-time news event extraction for global crisis monitoring. In *International Conference on Application of Natural Language to Information Systems*, pages 207–218. Springer.

Bishan Yang and Tom Mitchell. 2016. Joint extraction of events and entities within a document context. *arXiv preprint arXiv:1609.03632*.

Hong-Won Yun. 2011. Disaster events detection using twitter data. *Journal of information and communication convergence engineering*, 9(1):69–73.