

# An Event Extraction System via Neural Networks

Alapan Kuila

Indian Institute of Technology Kharagpur  
alapan.cse@iitkgp.ac.in

Sudeshna Sarkar

Indian Institute of Technology Kharagpur  
sudeshna@cse.iitkgp.ernet.in

## ABSTRACT

In this paper we describe the IIT KGP team’s participation in the Event Extraction task at FIRE 2017. We have developed an event extraction system which can extract event-phrases from tweets written in Indian language scripts along with Roman script. We designed our system on Hindi language and then used the same system for Malayalam and Tamil languages. We have submitted two systems one uses pipelined architecture another uses non-pipelined architecture. In case of pipelined architecture we first identify the tweets which contain event inside it and then extract the event-phrase from those tweets. In case of non-pipelined system all the tweets are directly pass to the event extraction system. Though conceptually simple, non-pipelined approach gives better result than pipelined approach and achieves F1-score of 50.01, 48.29 and 51.80 on Hindi, Malayalam and Tamil dataset respectively.

## 1 INTRODUCTION

Event Extraction from unstructured text is one of the most important and problematic task in Information extraction and natural language processing. Event extraction deals with automatic extraction of events depicting accidents, crime, natural disasters, political events etc. from various newswires, discussion forums, social media texts. Most of the existing event extraction systems [2, 8, 14] deals with English texts where main objective is to detect event trigger words and to classify those trigger words among predefined event classes [11, 14]. Though there exists several successful works for English language such as ACE, TAC<sup>1</sup> evaluation tracks but there is no such standard event extraction tool for Indian Languages. The Event extraction task at FIRE 2017 aims to identify and extract events from newswires and social media text specifically tweets. The tweets are written in three Indian language scripts: Hindi, Malayalam and Tamil along with romanized script. Unlike typical event extraction systems[8, 14] where the objective is to detect the trigger words from sentences and classify the words to a predefined event types, the FIRE 2017 shared task on event extraction deals with extraction of event-phrase (which depicts any event) from the given tweets. In this paper, we present the system we developed for this event extraction task at FIRE 2017 which deals with event extraction from newswires and social media text in Indian languages.

## 2 RELATED WORK

Many approaches have been taken to extract events from text. *Judea and Strube,2015* formulated the event extraction problem as frame-semantic parsing [4]. *McClosky et al.,2011* [12] uses dependency parsing to extract events. Previously researchers use feature based approach to extract events [3, 9, 18]. But features are domain dependent and needs huge linguistic knowledge [15].

<sup>1</sup><https://tac.nist.gov/2017/KBP/>

To overcome the difficulties of complicated feature engineering and domain dependency, researchers use neural network approach for event classification [2, 11, 14]. But all these works deal with English language and principle objective of these tasks is to detect the trigger word in the text which indicate an event. Some of these papers also identify the arguments related to these event trigger and their corresponding roles in the events [2, 14, 18].

## 3 TASK DEFINITION

Event extraction task at Fire 2017 requires participants to detect event-phrase from given tweets. In the training set tweets are written in three Indian languages: **Hindi, Malayalam and Tamil** along with romanized script. The objective is to detect the phrase within the tweet which depicts events such as natural disasters(floods, earthquakes etc), man made disasters (accidents, crime etc), political events (inaugurations by political leaders, political rallies etc), cultural/social events (Seminars, Conferences, light music events etc).

## 4 DATASETS

Dataset contain tweets written in both Indian languages and Roman script. Three Indian languages are: Hindi, Malayalam and Tamil. Training dataset contains two file for each language. One file contains all the tweets obtained using the Twitter API. Another annotation file contains event phrases extracted from tweets present in previous file. Each line in the annotation file contains: tweet-id, user-id, Event phrase of the tweet, index where this phrase starts in the tweet string, string length of the event phrase. Test file contains only the tweets with corresponding tweet-id and user-id. The details of the training and test dataset is depicted in the Table 1.

Table 1: Dataset description: number of tweets

Language	Training data	No of events in annotation file	Test data
Hindi	1024	402	4451
Malayalam	2218	674	5173
Tamil	3843	1109	5304

## 5 SYSTEM DESCRIPTION

In this section we describe our event extraction system. We have experimented with two types of event extraction systems: 1. **Non-pipelined approach** 2. **Pipelined approach**. We have used neural networks as the main technique in both the cases.

### 5.1 Preprocessing

The training file contains tweets which are written in mainly Indian language script with some Romanized script. Some of the tweets are ending with urls. To avoid data sparseness problem we have replaced all the urls with a unique token. Event annotation file contains some event phrases which are taken from same tweets and indicate same event and the words contained in those event-phrases are more or less same. We have omitted those redundant event-phrases.

### 5.2 Run1: Non-pipelined approach

In case of non-pipelined approach we have formulated the event extraction problem as sequence labelling problem. For every token in the input tweet we have tagged the word with '0' or '1' i.e. 'outside event-phrase' or 'inside event-phrase' respectively. And for this task we have used a combination of convolution neural network [7] along with bidirectional LSTM [16]. In order to prepare the input to the convolution layer we have made a fixed sequence length which is same as maximum tweet length and also used padding for shorter sentences with a special token when necessary. We have used an embedding layer in the neural network to transform each token into a real valued vector [13, 17]. And then the sequence of real valued vectors is fed to the neural network model. The main neural network architecture employed here is a combination of convolution neural network(CNN) [7] followed by a bidirectional LSTM [16]. Input to the convolution layer is a matrix of size  $n * m$  where  $n$  is the sequence length and  $m$  is the dimensionality of the word vector. CNN pass the input matrix representation through a convolution layer with a fixed filter length and filter size. And then without using any pooling layer we have again passed the output of the first convolution layer to the second convolution layer with another fixed filter length and size keeping the sequence length same as input sequence length. Now this internal representation is of size  $n * m_c$  where  $m_c$  is the dimension of internal vector representation. This internal vector representation is fed to a bidirectional LSTM with one hidden layer. The output of the bidirectional LSTM layer is followed by a softmax layer to compute the probability distribution over the possible tags of '0' or '1' for each token in the sequence.

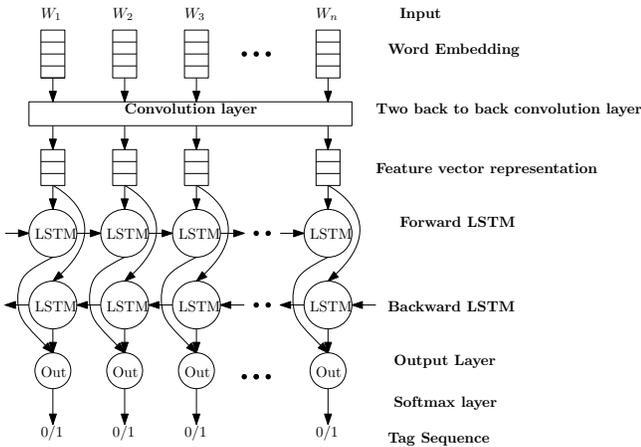


Figure 1: Event-extraction architecture

### 5.3 Run2: Pipelined approach

It is noticed in the training corpus that approximately 40% of the tweets contains event phrases. So it is vacuous to check all the tweets for extracting event phrase. From this intuition we have employed an event classification module before the event extraction module, depicted in non-pipelined approach. In case of pipelined approach first events are classified as **event-tweet** and **without event-tweets**. Tweets which are classified as event-tweets by our classification module are fed to the event extraction module. Other tweets which are classified as without event-tweets are discarded and are not fed to the event-extraction module. The classification module is similar to [5] [6] where authors have done sentence modelling and sentence classification using convolution neural network.

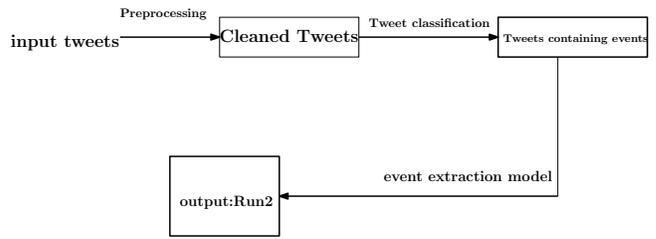


Figure 2: Block diagram of pipelined approach

5.3.1 Tweet Classification. Here we have used a convolution neural network(CNN) based architecture for tweet classification. As the tweets are of different length so padding is applied to make them of fixed size. Now these padded sequences are fed to an embedding layer to convert the tokens into a fixed size real-valued vectors. Then the sequences of fixed size vectors are fed to a convolution layer followed by maxpooling layer. The internal representation again fed to a combination of convolution layer followed by a pooling layer. The model uses multiple filter size to get multiple features. Now the output is fed to a fully connected softmax layer which gives the probability distribution over two classes: event-tweet or without event-tweet. The performance of Tweet-classification module is reported in Table 2.

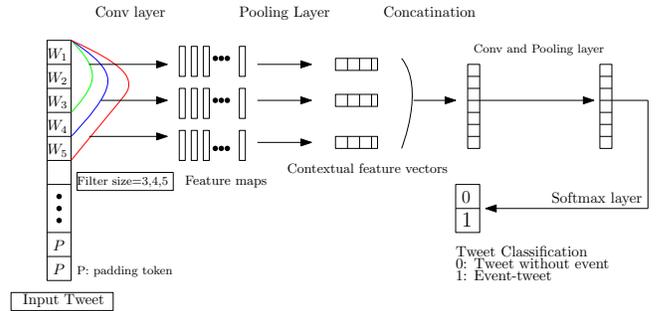


Figure 3: Tweet Classification Module

Eventually the tweets classified as event-tweets are fed to the event extraction module described in non-pipelined section. The

architecture of event extraction module in pipelined approach is same as non-pipelined approach. The only difference is that in case of pipelined approach at the training time we use only those tweets which contains events. Tweets which contains no event are discarded from training data.

Event extraction module will give the event span(i.e. the event phrase) within the tweets.

**Table 2: Tweet classification accuracy**

Language	Precision(%)	Recall(%)
Hindi	82.92	64.15
Malayalam	86.08	62.26
Tamil	83.33	63.69

## 5.4 Postprocessing

The event phrase which depicts events inside a tweet consists of cosecutive word sequences. So after sequence tagging if there exist '0's inside sequence of '1's then first '1' is taken as the strating point of event-phrase and the last '1' in the sequence indicates the ending of event-phrase. All the tokens inside the boundary are cosidered as event-phrase. We use this heuristic to maintain the constraint that all the event-phrases consists of consecutive tokens.

## 5.5 Parameters and training

Event extraction model used in pipelined and non-pipelined approach uses same architecture and hyperparameters. Regarding embeddings we have used 100 dimensions for word embedding in the word embedding layer. The first convolution layer uses filter size of 3 and number of filters used  $m_f = 30$ . In second convolution layer the filter size is 4 and number of filters  $m_h = 20$ . The bidirectional LSTM layer uses one hidden layer with hidden layer size 60. For event classification we have used CNN based classification approach which uses word embedding of size 100. These vectors are randomly initialized and fed to the embedding layer. We have employed filter size of {3, 4, 5} with 20 filters for each filter size for the convolution operation. Finally, we have trained the neural network models using adam optimizer with suffled minibatches, dropout rate=0.5, backpropagation for gradient calculation and parameter modification.

## 6 RESULT AND ERROR ANALYSIS

Table 3 shows the performance of event extraction in all three languages using both pipelined and non-pipelined approach. While examining the result in each language we have found that non-pipelined system has given better F-score than pipelined approach. In Hindi dataset Pipeline system acquire F-score of **40.35** but in non-pipelined approach the F-score is **50.01**. For Malayalam the F-score in Pipelined and non-pipelined approach are **47.17** and **48.29** respectively which are comparable. But in Tamil non-pipelined system whose F-score is **51.80** beats pipelined system (F-score: **44.01**). Error propagation in pipelined approach may be responsible for this low performance of pipelined system. The performance of tweet-classification module directly influenced the event extraction system in pipelined approach. It is also obvious from the Table.

3 that the precision is very much low in both pipelined and non-pipelined system. We will investigate on our model to improve the precision score.

**Table 3: Result on the final test set[P: Precision, R: Recall]**

Language	Run1			Run2		
	P (%)	R (%)	F-score (%)	P (%)	R (%)	F-score (%)
Hindi	<b>36.58</b>	<b>79.02</b>	<b>50.01</b>	<b>31.42</b>	<b>56.37</b>	<b>40.35</b>
Malayalam	<b>32.98</b>	<b>90.20</b>	<b>48.29</b>	<b>39.98</b>	<b>57.50</b>	<b>47.17</b>
Tamil	<b>43.16</b>	<b>64.77</b>	<b>51.80</b>	<b>39.73</b>	<b>49.33</b>	<b>44.01</b>

## 7 CONCLUSION AND FUTURE WORK

We have taken two strategies for event extraction. In case of non-pipelined approach we have classified each word with tag '0' or '1' indicating inside event phrase or outside event-phrase. But there are many tweets which do not indicate any event. So in pipelined approach first we have detected those tweets which contain any event and then identify the span of the event inside the tweet. The accuracy of the pipelined approach depends on accuracy of the tweet classification module. So we will try to improve the performance of tweet-classification module. In our experiment the number of training tweets are very low. If more training data could be used the event extraction accuracy may increase. In future we will try to increase the performance of the event extraction system by using more training data and other advanced strategies [1, 10].

## REFERENCES

- [1] Yubo Chen, Shulin Liu, Xiang Zhang, Kang Liu, and Jun Zhao. 2017. Automatically Labeled Data Generation for Large Scale Event Extraction. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*. 409–419. <https://doi.org/10.18653/v1/P17-1038>
- [2] Yubo Chen, Liheng Xu, Kang Liu, Daojian Zeng, Jun Zhao, et al. 2015. Event Extraction via Dynamic Multi-Pooling Convolutional Neural Networks.
- [3] 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*. Association for Computational Linguistics, 1127–1136.
- [4] Alex Judea and Michael Strube. 2015. Event Extraction as Frame-Semantic Parsing. In *\*SEM@NAACL-HLT*. 159–164.
- [5] Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A convolutional neural network for modelling sentences. *arXiv preprint arXiv:1404.2188* (2014).
- [6] Yoon Kim. 2014. Convolutional Neural Networks for Sentence Classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*. 1746–1751. <http://aclweb.org/anthology/D/D14/D14-1181.pdf>
- [7] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. ImageNet Classification with Deep Convolutional Neural Networks. In *Advances in Neural Information Processing Systems 25*, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger (Eds.), Curran Associates, Inc., 1097–1105. <http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf>
- [8] Qi Li, Heng Ji, and Liang Huang. 2013. Joint Event Extraction via Structured Prediction with Global Features. In *ACL (1)*. 73–82.
- [9] 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*. Association for Computational Linguistics, 789–797.
- [10] Shulin Liu, Yubo Chen, Shizhu He, Kang Liu, and Jun Zhao. 2016. Leveraging FrameNet to Improve Automatic Event Detection. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August*

7-12, 2016, Berlin, Germany, Volume 1: Long Papers. <http://aclweb.org/anthology/P/P16/P16-1201.pdf>

- [11] Shulin Liu, Yubo Chen, Kang Liu, and Jun Zhao. 2017. Exploiting Argument Information to Improve Event Detection via Supervised Attention Mechanisms. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, 1789–1798. <https://doi.org/10.18653/v1/P17-1164>
- [12] David McClosky, Mihai Surdeanu, and Christopher D. Manning. 2011. Event Extraction As Dependency Parsing. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1 (HLT '11)*. Association for Computational Linguistics, Stroudsburg, PA, USA, 1626–1635. <http://dl.acm.org/citation.cfm?id=2002472.2002667>
- [13] Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. 2013. Linguistic regularities in continuous space word representations.
- [14] Thien Huu Nguyen, Kyunghyun Cho, and Ralph Grishman. 2016. Joint Event Extraction via Recurrent Neural Networks. In *HLT-NAACL*. 300–309.
- [15] Thien Huu Nguyen and Ralph Grishman. 2015. Event Detection and Domain Adaptation with Convolutional Neural Networks.
- [16] M. Schuster and K.K. Paliwal. 1997. Bidirectional Recurrent Neural Networks. *Trans. Sig. Proc.* 45, 11 (nov 1997), 2673–2681. <https://doi.org/10.1109/78.650093>
- [17] Joseph Turian, Lev Ratinov, and Yoshua Bengio. 2010. Word representations: a simple and general method for semi-supervised learning. In *Proceedings of the 48th annual meeting of the association for computational linguistics*. Association for Computational Linguistics, 384–394.
- [18] Bishan Yang and Tom M. Mitchell. 2016. Joint Extraction of Events and Entities within a Document Context. *CoRR* abs/1609.03632 (2016). [arXiv:1609.03632](http://arxiv.org/abs/1609.03632)