Multi-linguality Helps: Event-Argument Extraction for Disaster Domain in Cross-lingual and Multi-lingual Setting

Zishan Ahmad, Deeksha Varshney, Asif Ekbal, Pushpak Bhattacharyya
Department of Computer Science and Engineering
Indian Institute of Technology Patna
{1821cs18, 1821cs13, asif, pb}@iitp.ac.in

Abstract

Automatic extraction of disaster-related events and their arguments from natural language text is vital for building a decision support system for crisis management. Event extraction from various news sources is a well-explored area for this objective. However, extracting events alone, without any context provides only partial help for this purpose. Extracting related arguments like *Time*, *Place*, *Casualties*, etc., provides a complete picture of the disaster event. In this paper, we create a disaster domain dataset in *Hindi* by annotating disaster-related event and arguments. We also obtain equivalent datasets for *Bengali* and *English* from a collaboration. We build a multi-lingual deep learning model for argument extraction in all the three languages. We also compare our multi-lingual system with a similar baseline monolingual system trained for each language separately. It is observed that a single multi-lingual system is able to compensate for lack of training data, by using joint training of dataset from different languages in shared space, thus giving a better overall result.

1 Introduction

The ability to extract real time news of disaster events automatically, can potentially help in better decision-making for planning and coordination of disaster relief efforts. Event extraction from text entails the extraction of particular types of events along with their arguments. Information obtained from extracted event mentions provides a more structured and clear picture when augmented with related arguments like *Time*, *Place*, *Participant*, *Casualty* etc. In a language rich world where each event is documented in multiple languages, argument extraction in multi-lingual setting stands as a crucial task.

Extraction of events from news is a well explored area in Natural Language Processing. Competitions such as ACE2005 (Doddington et al., 2004) and TAC-KBP2015 (Mitamura et al., 2015) have investigated the area and provided a large body of literature on event extraction from news articles. Event extraction was done on ACE2005 dataset by Ji and Grishman (2008) by combining global evidence from related documents with local decisions. Hou et al. (2012) introduced a method of event argument extraction based on CRFs model for ACE 2005 Chinese event corpus. Event and its arguments were extracted by Petroni et al. (2018), for the purpose of extracting breaking news. Although extraction of events is quite well examined, there is a scarcity of work in extraction of detailed arguments for disaster domain like *casualties*, *reason*, *after-effects* etc.

In this paper we create and publish a dataset annotated for events in disaster domain, for three different languages, i). *Hindi*, ii). *Bengali* and iii). *English*. This dataset is annotated for the task of argument extraction by expert annotators. We build a 'mono-lingual' deep learning system, based on CNN (Convolutional Neural Network) and Bi-LSTM (Bi-Directional Long Short Term Memory) for the task of argument extraction. In order to leverage the information from all the languages while training, and improve the performance of the system, we build a 'multi-lingual' argument extraction system. This is done by adding separate language layers for each language to our 'mono-lingual' system. To bring the datasets of all the languages to the same vector space, we make use of 'multi-lingual' word embeddings. We show that by training our model in this way we are able to utilize the dataset of all the three languages and improve the performance of our system for most arguments in the three languages. We also investigate how the syntactic difference of the languages is handled by our system. Through analysis, we show that 'multi-lingual' training is espe-

D M Sharma, P Bhattacharyya and R Sangal. Proc. of the 16th Intl. Conference on Natural Language Processing, pages 135–142
Hyderabad, India, December 2019.

cially helpful in improving the performance when some argument is under-represented in the 'monolingual' training data.

1.1 Problem Definition

Argument extraction entails classifying each word in the sentence into some argument or not argument. Therefore, it has been formulated as a sequence labelling task. Given a sentence of form $w_1, w_2, ..., w_n$, the task is to predict the sequence of event-arguments, of the form $l_1, l_2, ..., l_n$. Six different types of arguments were annotated in the dataset: i). *Place*, ii). *Time*, iii). *Reason*, iv). *Casualties*, v). *Participant* and vi). *After-effects*. To label multi-word event-arguments, IOB-style encoding is used where B, I and O denote the beginning, intermediate and outside token of an event.

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

2 Related Works

A major task in information extraction is detection of event triggers, event classification and event argument extraction. Recent works on event trigger detection and classification discuss efficient feature representation techniques which can help in event extraction. Nguyen and Grishman (2015) proposed a convolutional neural network for event extraction which automatically learns features from text. Chen et al. (2015) introduced dynamic convolutional neural network (DMCNN), which adopt a dynamic multi-pooling layer in accordance with the event triggers and its arguments. In 2016, Nguyen and Grishman (2016) improved their CNN model by introducing the non-consecutive convolution by skipping irrelevant words in a sequence. Feng et al. (2018) designed a combined model of LSTM's and CNN's which helped in capturing both sequence level and chunk level information from specific contexts. Nguyen and Grishman (2018) explored graph convolutional network over dependency trees and entity mention-guided pooling. For low resource languages, Liu et al. (2018) came up with Gated Multi-Lingual Attention (GMLATT) and Lin et al. (2018) developed a multi-lingual multi-task architecture alleviating data sparsity problem in related tasks and languages.

Previously, in event argument extraction researchers have experimented with pattern based methods (Patwardhan and Riloff, 2007; Chambers and Jurafsky, 2011) and machine learning based methods (Patwardhan and Riloff, 2009; Lu and Roth, 2012) most of which utilise the various kinds of features obtained from the context of a sentence. Higher level representations such as cross-sentence or cross-event information were also explored by Hong et al. (2011) and Huang and Riloff (2011). Maximum Entropy based classifiers were applied for event and argument labeling by Ahn (2006); Chen and Ji (2009); Zhao et al. (2008). The disadvantage with ME classifier is that it gets stuck in local optima and fails to fully capture the context features. To overcome this Hou et al. (2012) proposes a event argument extraction system based on Conditional Random Fields (CRF) model that can select any features and normalizing these features in overall situation helps in obtaining optimal results. While, these models can get affected by the error propagated from upstream tasks, a joint model can help us utilise the close interaction between one or more similar tasks. Li et al. (2013) presented a joint model for Chinese Corpus which identifies arguments and determines their roles for event extraction using various kinds of discourse-level information. On ACE2005 dataset Sha et al. (2018) proposed a dependency bridge recurrent neural network (dbRNN) built upon LSTM units for event extraction. They use dependency bridges over Bi-LSTM to join syntactically similar words. A tensor layer is applied to get the various argument-argument interactions. Event triggers and arguments are then jointly extracted utilising a max-margin criterion. Nguyen et al. (2016) presented a GRU model to jointly predict events and its arguments.

We introduce two systems for the task of event argument extraction. First is our monolingual system built using CNN (Convolutional Neural Network) and Bi-LSTM (Bi-Directional Long Short

Term Memory). To exploit the information from related languages, we develop a second system that can use information from all the languages for training. This multi-lingual system is built by using shared vector space of embeddings while training, and by using separate language layers for each language to accommodate for diversity in syntax of the languages.

3 Methodology

In this paper, we propose that joint training of IE system on different language datasets, using 'multi-lingual' word embeddings and language layers helps in better extraction of arguments. This is particularly true when the dataset is limited in size. To corroborate our claim, we device two different systems, i). monolingual baseline system, and ii). multi-lingual system. The 'monolingual baseline' system only takes input data (sentence wise) from one language and extracts the arguments. For word representation, it uses monolingual word embeddings. The 'multi-lingual' argument extraction system uses separate language layers and multi-lingual word embeddings for joint training on all the three languages.

3.0.1 Monolingual Word Embedding

The monolingual word-embeddings that are used in our experiments are also known as fastText[1]. It was proposed by Bojanowski et al. (2017), and is based on the skipgram model. However instead of using one-hot vector encoding for each word while training, a vector representation of a word that considers character n-grams occurring in the word is formed. To get this representation, the n-grams from all the words for 'n' greater than 2 and smaller than 7 are extracted. After this, a dictionary of all the extracted n-grams is created. A given word w, can now be denoted by $\Gamma_w \subset \{1, ..., G\}$ i.e the set of n-grams appearing in the word; where G is the size of the n-gram dictionary. With each n-gram in G, a vector representation z_g is associated. A word represention is obtained by summing up all the n-grams, as described in Equation 1:

$$V_w = \sum_{g \in G_w} z_g \tag{1}$$

The continuous skip-gram model used these word vectors V_w, to obtain word-embedding representations of words. The main advantage of this technique is that, even in the absence of some word in the training corpus, some representations of the word is still obtained as the n-gram representation of words is considered. This skip-gram model is trained using *Wikipedia* data dump of each language. The dimension of the word vector to is set to 300.

3.0.2 Multi-lingual Word Embedding

Multi-lingual embeddings are obtained by learning a mapping matrix W, between source embeddings $X = \{x_1, x_2, x_3..., x_n\}$ and target embeddings $Y = \{y_1, y_2, y_3, ..., y_n\}$ without cross-lingual supervision.Adversarial training was used in this method proposed by Conneau et al. (2017). A discriminator is trained to discriminate between a randomly sampled element from $WX = \{Wx_1, ..., Wx_n\}$ and Y. At the same time W is trained to prevent the discriminator from making correct prediction. Thus making it a two-player game, where the discriminator tries to maximize its capability of identifying the origins of an embedding, and W aims to prevent the discriminator from doing so by making WX and Y as indistinguishable as possible. The W matrix is trained with near orthogonality constraint, to ensure that while transforming the source vector to the target vector space, the angles and distances between words in the embeddings are not distorted during transformation. To achieve this near orthogonality constraint, weight updation for W is done using Equation 2.

$$W \leftarrow (1 + \beta)W - \beta(WW^T)W \tag{2}$$

Here, β was set to 0.01 for the transformation. For our experiments we trained mapping matrices W_{hindi} and $W_{bengali}$ that map the *Hindi* and *Bengali* word embeddings to the vector space of *English* embeddings.

3.1 Monolingual Baseline Model

The 'monolingual baseline' model (c.f Figure 1) is based on Bi-Directional Long Short Term Memory (Bi-LSTM) (Hochreiter and Schmidhuber, 1997; Schuster and Paliwal, 1997) and Convolutional Neural Networks (CNN) (Kim, 2014). The input to the model is a sentence, represented by a sequence of monolingual word embeddings. Since Bi-LSTM and CNN take sequences of equal lengths, the shorter sequences are padded by zero

[1] https://github.com/facebookresearch/fastText

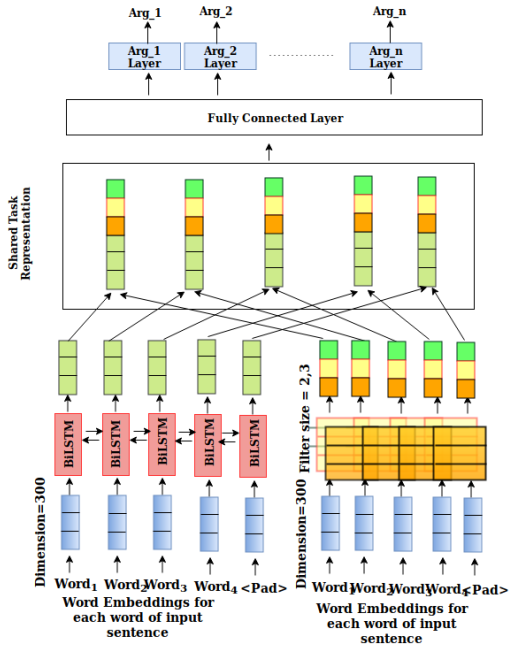

Figure 1: monolingual baseline model for argument extraction

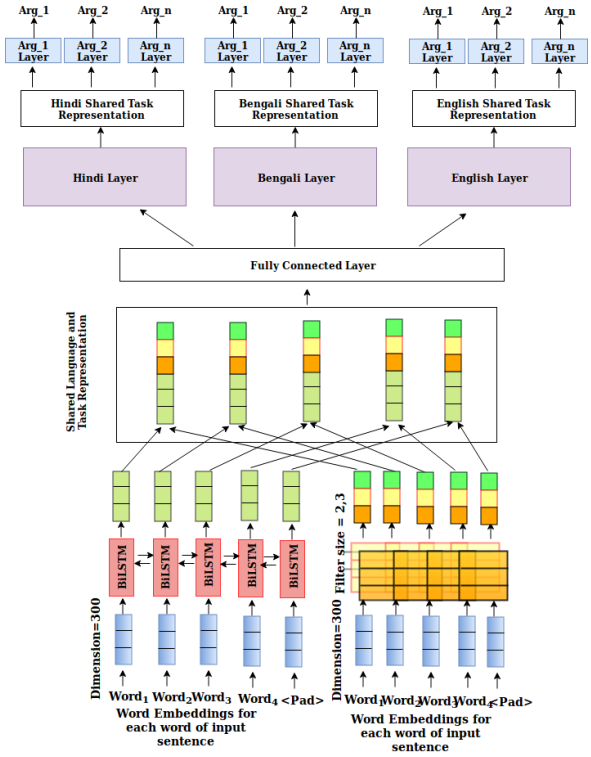

Figure 2: Multi-lingual baseline model for argument extraction

vectors. This sequence is passed through Bi-LSTM and CNN having filter size 2 and 3. The Bi-LSTM gives contextual representation of each word, while the CNN extracts the 'bi-gram' and 'tri-gram' features for the sequence. These features are concatenated and passed through a fully connected layer. This layer gives shared representation for the task of argument extraction. Since the arguments in the dataset are not mutually exclusive (E.g: *Place* or *Participant* argument can also be a part of *Reason* or *After-effect* argument), we have different layers to predict different arguments independently. We have 6 different fully-connected layers in parallel, each of them specialized for detection of one of the 6 arguments. 'Softmax' is used after each of the final layers to classify the representation into I, O or B of an argument.

3.2 Multi-lingual Model

For multi-lingual system, we build a model based on the baseline model, by adding separate language layers (L_1, L_2 and L_3) for each language (c.f Figure 2). A layer L_i and its subsequent layers are only trained when input data is also of language L_i. We represent the input sentence as a sequence of multi-lingual word embeddings, and padding with zero vectors is used to make the sequence equal in length. Similar to the 'monolingual baseline' model, Bi-LSTM, CNN and a fully connected layer is used. This fully connected layer produces shared language and task representation as output. Three separate language layers for the languages *Hindi*, *Bengali* and *English* are used in parallel. These language layers decode the language specific representation from shared representation. After each language layer we have 6 fully connected layers for each of the 6 arguments. 'Softmax' classifier is used to classify the representation into I, O or B of an argument.

4 Dataset and Experiments

In this section, we describe the dataset used and the experiments conducted.

4.1 Dataset

To create the dataset, we crawled news articles in disaster domain from popular news websites in *Hindi*. These news articles were annotated by three annotators, with good language abilities and having satisfactory knowledge in the relevant area. The guidelines for annotation used were similar to the guidelines given by TAC KBP 2017 Event Sequence Annotation Guidelines[2]. We recorded that the annotators had Kappa agreement score of 0.85

[2] https://cairo.lti.cs.cmu.edu/kbp/2017/event/TAC_KBP_2017_Event_Coreference_and_Sequence_Annotation_Guidelines_v1.1.pdf

Argument	Hindi	Bengali	English
Time	3,953	11,042	822
Place	12,410	10,576	3,018
Reason	1,573	1,744	544
Casualties	12,171	15,870	4,823
Participant	2,264	4,311	639
After-effects	13,355	9,731	274

Table 1: Distribution of number of arguments in *Hindi*, *Bengali* and *English* datasets

on average. We also obtained equivalent dataset in *Bengali* and *English* language from a collaboration. The total dataset is comprised of 2,191 documents (*Hindi*: 922, *Bengali*: 999 and *English*: 270). It contains 44,615 sentences (*Hindi*: 17,116, *Bengali*: 25,717 and *English*: 1,782). The six arguments in the dataset and their distribution in the three languages are detailed in the Table 1.

4.2 Experiments

We conduct two separate experiments to show that dataset from different languages (L_1 and L_2) can be leveraged to improve the performance of argument extraction system of a different language (L_3). First we conduct experiment to obtain baseline results on 'mono-lingual' setup. Next, we perform experiment using the combined dataset of all the three languages using 'multi-lingual' argument extraction model.

4.2.1 Monolingual Experiment

This experiment is conducted separately on each dataset using the 'monolingual baseline model' (c.f. Figure 1) and monolingual *fastText* embeddings. The results of this experiment is used as a baseline, against which the results of the other experiment is compared. The following set-up is used for the experiment: i). learning rate: 1×10^{-2}, ii). batch size: 32, iii). optimizer: Adam (Kingma and Ba, 2014), iv). loss function: Binary cross-entropy. The best model based on validation-set accuracy was saved after 100 epochs.

4.2.2 Multi-lingual Experiment

This experiment is conducted on the combined dataset of three languages, using the 'multi-lingual model' (c.f Figure 2). Multi-lingual word embeddings (described in Section 3.2) were used for word representation in all the three languages, in this experiment. The same experimental set-up used for the 'monolingual baseline' experiment, is also used for this experiment. The training of multilingual system was done batch wise, i.e. each language branch was trained for one batch alternatively. The number of steps per epochs was decided by the number of batches needed to complete one epoch of the largest training set, among the different language datasets.

5 Results and Analysis

In this section, we discuss the results obtained for the two experiments described in Section 4.2. We also provide analysis of the results. F1-Score is used as an evaluation metric, and all the results reported are 5-Fold cross-validated. The results for both, 'monolingual' and 'cross-lingual' experiments are reported in Table 2. From the results, it can be observed that F1-score for *Hindi* and *English* datasets improve for most arguments (5 out of 6 arguments), while the results for *Bengali* dataset improves for three out of the six arguments.

We also test the statistical significance of each increment in F1-Score for argument extraction. The 'p-values' obtained after 't-test' are shown in Table 3. It can be seen that most improvements in F1-score are statistically significant.

It is observed that multi-word *Time* arguments are better captured by 'multi-lingual' model than by the 'monolingual baseline' model. An example of this can be seen in the following sentence:

- **Hindi Text:** एसएसपी संतोष कुमार सिंह ने बताया कि रविवार रात को जलालपुर पर तैनात पुलिसकर्मियों ने बाइक पर सवार दो युवकों को रोकने की कोशिश की
- **Transliteration:** esesapee santosh kumaar sinh ne bataaya ki **ravivaar raat** ko jalaalapur par tainaat pulisakarmiyon ne baik par savaar do yuvakon ko rokane kee koshish kee
- **Translation:** SSP Santosh Kumar Singh said that on **Sunday night**, policemen stationed at Jalalpur tried to stop two youths riding on bikes.

In the aforementioned sentence the actual phrase denoting time is 'रविवार रात' (Sunday night). However the 'monolingual' model only detects 'रविवार' (Sunday) as the *Time* argument. However, after multi-lingual training the entire time phrase is correctly detected. This is because the lack of training data for multi-word time arguments in *Hindi*,

	Mono-lingual			Multi-lingual		
Argument	**Hindi**	**Bengali**	**English**	**Hindi**	**Bengali**	**English**
Time	0.60	**0.86**	0.56	**0.61**	0.85	**0.58**
Place	**0.58**	**0.61**	**0.57**	0.56	0.59	0.55
Reason	0.01	0.19	0.14	**0.16**	**0.22**	**0.20**
Casualties	0.58	**0.73**	0.62	**0.59**	0.71	**0.63**
Participant	0.35	0.50	0.30	**0.41**	**0.53**	**0.32**
After-effects	0.25	0.28	0	**0.30**	**0.35**	**0.13**

Table 2: Results (F1-Scores) for 'mono-lingual' and 'multi-lingual' experiments on *Hindi*, *Bengali* and *English* datasets: 5-Fold cross-validated

Argument	**Hindi**	**Bengali**	**English**
Time	0.46	n/a	0.03
Place	n/a	n/a	n/a
Reason	0.03	0.18	0.04
Casualties	0.39	n/a	0.10
Participant	0.01	0.11	0.54
After-effects	0.04	0.09	0.01

Table 3: The 'p-values' obtained for each improvement in results from the baseline 'mono-lingual' to 'multi-lingual' experiment (n/a is used for instances where no improvement was observed)

is supplemented by training data from *Bengali* and *English*.

Another interesting observation is that, for *Casualty* argument of *English* dataset, the 'monolingual' system often confuses people as casualties, even when they are not. An example of such observation is as follows:

- **Actual:** Over 200000 people in 36 villages located 6 miles (10 km) from the volcano were advised to evacuate immediately.
- **Monolingual Prediction:** Over **200000 people** in 36 villages located 6 miles (10 km) from the volcano were advised to evacuate immediately.
- **Multi-lingual Prediction:** Over 200000 people in 36 villages located 6 miles (10 km) from the volcano were advised to evacuate immediately.

In the above example the phrase '200000 people' does not denote casualty, however the 'monolingual' model confuses it as casualty. This is due to the lack of training data in *English* to learn the difference between some count of people and actual casualty. However, after 'multi-lingual' training the model is able to make this distinction correctly.

The F1-score for *Place* arguments for all the datasets, is better for the 'monolingual baseline' model. This is because *Place* argument is present in good numbers for all the datasets, therefore there are enough instances for proper training of deep learning model, even in monolingual setting. Using 'multi-lingual model' for such cases is of little help. Furthermore, the syntactic difference between languages confuses the system, thus degrading the performance of the 'multi-lingual' system. A good example of this phenomenon is show below:

- **Actual:** Three youths lost their lives when the car they were travelling in collided with a truck near **Gaddoli village of Naraingarh in Ambala**.
- **Monolingual Prediction:** Three youths lost their lives when the car they were travelling in collided with a truck near **Gaddoli village of Naraingarh in Ambala**.
- **Multi-lingual Prediction:** Three youths lost their lives when the car they were travelling in collided with a truck near **Gaddoli village of Naraingarh** in **Ambala**.

It can be observed that the 'monolingual baseline' model predicts the entire phrase describing the *Place* argument correctly. However the prediction by 'multi-lingual model' misses the preposition 'in', which is present between 'Naraingarh' and 'Ambala'. The same sentence can be written in *Bengali* as follows:

- **Bengali Transliteration:** Ambālāra nārāẏanagaṛēra gāddali grāmēra kāchē ēkaṭi ṭrākēra sāthē ṭrēnēra mukhōmukhi saṅgharṣē tinajana yubaka prāṇa hārāẏa.

The phrase 'in Ambala' is represented by a single word 'Ambālāra', in *Bengali*. This difference in syntax between languages, makes the 'multi-lingual' system miss the word 'in' thus degrading the performance of the system.

The best improvement in F1-score is observed for the arguments *Reason* and *After-effects* for the *English* language. This is because these two arguments have least support in the dataset, and thus multi-lingual training helps by mitigating the scarcity in training examples. The same phenomenon can also be observed for *Reason* argument which has a low support in *Hindi* dataset. Thus through our analysis we can conclude that, 'multi-lingual' training can help in improving the performance of the system for low support classes. However, it can also cause confusion and deteriorate the performance for high support classes.

6 Conclusion

In this paper we create a dataset for argument extraction for disaster domain, for three languages *Hindi*, *Bengali* and *English*. We then build a deep learning model for extraction of these argument in each language separately. Since the data is limited in size, we build another model that leverages data from all the languages. To make use of different language datasets, we first bring the word embeddings of all the three languages to the same vector space. We also use separate language layers to accommodate divergence in syntax of the languages. Through our experiments we show that training in shared vector space by using 'multi-lingual' system helps in improving the performance of low support arguments. We also show that the for high support arguments, the syntactic difference in language can sometimes overcome the benefit of 'multi-lingual' training and cost in performance of our proposed 'multi-lingual' system.

In future we would like to explore how to handle these syntactic differences so that the performance can be further improved. It would also be interesting to explore the range of languages that can be trained successfully in a multi-lingual setting.

7 Acknowledgement

The research reported in this paper 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

David Ahn. 2006. The stages of event extraction. In *Proceedings of the Workshop on Annotating and Reasoning about Time and Events*, pages 1–8.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Nathanael Chambers and Dan Jurafsky. 2011. Template-based information extraction without the templates. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 976–986. Association for Computational Linguistics.

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)*, pages 167–176.

Zheng Chen and Heng Ji. 2009. Language specific issue and feature exploration in chinese event extraction. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics, Companion Volume: Short Papers*, pages 209–212.

Alexis Conneau, Guillaume Lample, Marc'Aurelio Ranzato, Ludovic Denoyer, and Hervé Jégou. 2017. Word translation without parallel data. *arXiv preprint arXiv:1710.04087*.

George R Doddington, Alexis Mitchell, Mark A Przybocki, Lance A Ramshaw, Stephanie M Strassel, and Ralph M Weischedel. 2004. The automatic content extraction (ace) program-tasks, data, and evaluation. In *LREC*, volume 2, page 1.

Xiaocheng Feng, Bing Qin, and Ting Liu. 2018. A language-independent neural network for event detection. *Science China Information Sciences*, 61(9):092106.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

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.

Libin Hou, Peifeng Li, Qiaoming Zhu, and Yuan Cao. 2012. Event argument extraction based on crf. In *Workshop on Chinese Lexical Semantics*, pages 32–39. Springer.

Ruihong Huang and Ellen Riloff. 2011. Peeling back the layers: detecting event role fillers in secondary contexts. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 1137–1147. 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*.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Peifeng Li, Qiaoming Zhu, and Guodong Zhou. 2013. Joint modeling of argument identification and role determination in chinese event extraction with discourse-level information. In *Twenty-Third International Joint Conference on Artificial Intelligence*.

Ying Lin, Shengqi Yang, Veselin Stoyanov, and Heng Ji. 2018. A multi-lingual multi-task architecture for low-resource sequence labeling. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 799–809.

Jian Liu, Yubo Chen, Kang Liu, and Jun Zhao. 2018. Event detection via gated multilingual attention mechanism. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Wei Lu and Dan Roth. 2012. Automatic event extraction with structured preference modeling. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*, pages 835–844. Association for Computational Linguistics.

Teruko Mitamura, Zhengzhong Liu, and Eduard H Hovy. 2015. Overview of tac kbp 2015 event nugget track. In *TAC*.

Thien Huu Nguyen, Kyunghyun Cho, and Ralph Grishman. 2016. Joint event extraction via recurrent neural networks. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 300–309.

Thien Huu Nguyen and Ralph Grishman. 2015. Event detection and domain adaptation with 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 2: Short Papers)*, pages 365–371.

Thien Huu Nguyen and Ralph Grishman. 2016. Modeling skip-grams for event detection with convolutional neural networks. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 886–891.

Thien Huu Nguyen and Ralph Grishman. 2018. Graph convolutional networks with argument-aware pooling for event detection. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Siddharth Patwardhan and Ellen Riloff. 2007. Effective information extraction with semantic affinity patterns and relevant regions. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 717–727.

Siddharth Patwardhan and Ellen Riloff. 2009. A unified model of phrasal and sentential evidence for information extraction. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 1-Volume 1*, pages 151–160. Association for Computational Linguistics.

Fabio Petroni, Natraj Raman, Tim Nugent, Armineh Nourbakhsh, Žarko Panić, Sameena Shah, and Jochen L Leidner. 2018. An extensible event extraction system with cross-media event resolution. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 626–635. ACM.

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

Lei Sha, Feng Qian, Baobao Chang, and Zhifang Sui. 2018. Jointly extracting event triggers and arguments by dependency-bridge rnn and tensor-based argument interaction. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Yan-yan Zhao, Bing Qin, Wan-xiang Che, and Ting Liu. 2008. Research on chinese event extraction. *Journal of Chinese Information Processing*, 22(1):3–8.