Using Entity Relation to Improve Event Detection via Attention Mechanism

Jingli Zhang, Wenxuan Zhou, Yu Hong(✉), Jianmin Yao, and Min Zhang

Computer Science and Technology, Soochow University, Suzhou, Jiangsu, China
jlzhang05@gmail.com, chrisnotkris7@gmail.com, tianxianer@gmail.com,
{jyao,minzhang}@suda.edu.cn

Abstract. Identifying event instance in texts plays a critical role in the field of Information Extraction (IE). The currently proposed methods that employ neural networks have successfully solve the problem to some extent, by encoding a series of linguistic features, such as lexicon, part-of-speech and entity. However, so far, the entity relation hasn't yet been taken into consideration. In this paper, we propose a novel event extraction method to exploit relation information for event detection (ED), due to the potential relevance between entity relation and event type. Methodologically, we combine relation and those widely used features in an attention-based network with Bidirectional Long Short-term Memory (Bi-LSTM) units. In particular, we systematically investigate the effect of relation representation between entities. In addition, we also use different attention strategies in the model. Experimental results show that our approach outperforms other state-of-the-art methods.

Keywords: Event detection · Attention mechanisms · Entity relation

1 Introduction

Event extraction (EE) is an important task in IE. The purpose is to detect event triggers with specific types and their arguments. This paper only tackles event detection (ED) task, which is a crucial part of EE, focusing on identifying event triggers and their categories. Take the following sentence for example:

S1: *David Kaplan was killed by sniper* ***fire*** *in the Balkans in 1992.*

There is an *Attack* event is mentioned in **S1**. The "***fire***" is annotated as trigger in ACE-2005 corpus. Thus, an ED system should be able to identify the trigger as "***fire***" and assign it an event type ***Attack***.

However, it might be easily misclassified as ***End-Position*** event in reality because "***fire***" is a polysemy. In this case, Liu et al. [15] utilized entity to reinforce the classification. Such as in **S1**, they proposed that considering the word "***sniper***", which serves as an entity (*Attacker)* of the target event, to get more confidence in predicting the trigger as an ***Attack*** event.

M. Zhang et al. (Eds.): NLPCC 2018, LNAI 11108, pp. 171–183, 2018.
https://doi.org/10.1007/978-3-319-99495-6_15

Unfortunately, although the entity information is considered, there are still difficulties to identify event type correctly for some sentences. For example, consider the following two instances:

S2: *Swenson herself would also* ***leave*** *the NCA*.

S3: *It's the Iraqi ambassador's time to* ***leave*** *the United States*.

Table 1 lists the relevant information about S2 and S3. In S2, "*leave*" is a trigger of type *End-Position*. However, in S3, "*leave*" is a trigger of type *Transport*, which is more common than *End-Position*. In addition, the entities in S2 are *Non-Governmental* and *Individual*, which have no discrimination for identifying event type with entities (*Nation* and *Individual*) in S3. Because they all indicate institution and individual. However, if we consider the relation between the entities, which is *Membership* in S2 and *Located* in S3, we would have more confidence in predicting the *End-Position* event and the *Transport* event respectively. Such as in S2, *Membership* indicates the member relationship between two entities, then the probability that the sentence will contain an *End-Position* event will be higher than a *Transport* event.

Table 1. Event, Entity and Relation labels for the above two instances.

Instance	S2	S3
Event	*leave:* **End-Position**	*leave:* **Transport**
Entity1	*NCA:* **Non-Governmental**	*United States:* **Nation**
Entity2	*swenson:* **Individual**	*ambassador:* **Individual**
Relation	**Membership**	**Located**

In addition, we note that words in sentence have different contribution degrees for the correct recognition of trigger. Some words are important, yet others are meaningless. For example, in **S1**, "*killed*" and "*sniper*" provide more important clues than other words that an *Attack* event might happen. Thus, these words should get more attention than others. Guided by this, we employ attention mechanism to model the sentence. Attention values indicate the importance of different words in the sentence of predicting the event type.

To sum up, based on the entity relation information and different significant clues that different words can provide, we propose an attention-based Bi-LSTM model and utilize relation type embedding in the model to detect event. In summary, the contributions of this paper are as follows:

- We analyze the impact of entity relation information in ED task, and effectively merge it to the ED system.
- We propose an attention-based Bi-LSTM model, which aims to capture more important information within a sentence for ED task. In addition, we also investigate different attention strategies for the proposed model.

- We conduct extensive experiments on the ACE-2005 corpus. The experimental results show that our method outperforms other state-of-the-art methods.

The rest of the paper is organized as follows. Section 2 provides a brief overview of some event research works. Section 3 describes the ED task specifically. Section 4 gives a detailed introduction of our proposed method. Section 5 shows the experimental results and Sect. 6 concludes the paper.

2 Related Work

Event detection is an important subtask of event extraction [5]. Many approaches have been explored for ED. We divide these methods into feature-based methods and structure-based methods.

In feature-based methods, Ahn and David [1] leverage lexical, syntactic and external knowledge features to extract the event. Ji and Grishman [9] use rich evidence from related documents for the event extraction. Furthermore, Liao et al. [11] and Hong et al. [8] proposed the cross-event and cross-entity inference methods to capture more clues from the texts. Benefiting from the common modelling framework, these methods can achieve the fusion of multiple features, in addition, they can be used flexibly through feature selection. But feature engineering requires considerable expertise.

In structure-based methods, the use of neural network for ED has become a promising research. Some researchers like Nguyen et al. [18] and Chen et al. [3] learn continuous word representations and regard them as features to infer whether a word is a trigger or not by exploring neural network. Respectively, Nguyen et al. proposed a Convolutional Neural Network (CNN) with entity type information and word position information as extra features and Chen et al. presented a Dynamic Multi-pooling layer (DM-CNN) to capture information from sentence. Tang et al. [19] presented Bi-LSTM to model the preceding and following information of a word as people generally believe that LSTM is good at capturing long-term dependencies in a sequence. Some other models also effectively improve the performance of ED task, including Nguyen et al. [17] Bidirectional Recurrent Neural Network (Bi-RNN) and Feng et al. [6] Hybrid which combines the Bi-LSTM and CNN neural network.

3 Task Description

In this paper, we adopt the event extraction task defined in Automatic Content Extraction (ACE) evaluation. An event indicates a specific occurrence involving one or more participants. We introduce some ACE terminologies to facilitate the understanding of this task:

- **Event trigger**: the main word that most clearly expresses the occurrence of an event (typically, a verb or a noun).
- **Event argument**: the mentions that are involved in an event (participants).

- **Entity**: an object or a set of objects in one of the semantic categories.
- **Relation**: some relationship between entities in one sentence.

For easy to understand, we can see the instance of **S2**: an event extractor should detect an *End-Position* event, along with the event trigger "*leave*". Moreover, the entities in this sentence are *NCA* and *swenson*. The relation between the entities is *Membership*. In this paper, we formalize the ED problem as a multi-class classification problem following previous work [5]. Given a sentence, we classify every token of the sentence into one of the predefined event classes or non-trigger class.

In addition, since entity recognition and relation detection are difficult task in ACE evaluation and not the focus in the event extraction task, we directly leverage the ground-truth entity and relation labels.

4 Our Approach

We model the ED task as a multi-class classification task. In detail, given a sentence, we treat every token in this sentence as a trigger candidate, and our goal is to classify each of these candidates into one of 34 classes.

In this section, we illustrate details of our approach, including the sentences representation which is the input of the model, the attention strategies, the use of Bi-LSTM which is to encode semantics of each word with its preceding and following information, as well as the other details in the model.

4.1 Input

We follow Chen et al. [3] to take all tokens of the sentence as the input. Before feeding tokens into the network, we transform each of them into a real-valued vector D. The vector is formed by concatenating a word embedding, an entity type embedding with a relation type embedding. As shown in Fig. 1.

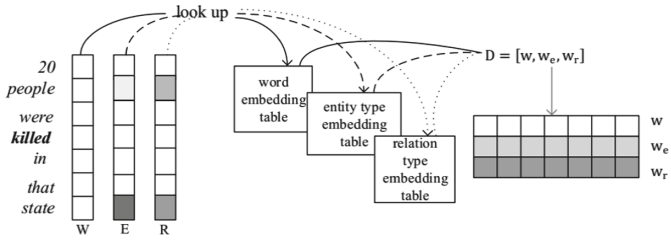

Fig. 1. Embedding.

Word Embedding. In this paper, we limit the context to a fixed length by trimming longer sentences and padding shorter sentences with special token. We let n be the fixed length. w_i is the current candidate trigger. So, we get the representation of the sentence: $W = \{w_1, w_2, ..., w_i, ..., w_n\}$. Then, looking up a pre-trained word embedding table to get the word embedding representation w. It is a fixed-dimensional real-valued vector which represents the hidden semantic properties of a token [4]. We use the Skip-gram model [16,20] to learn word embeddings on the NYT corpus[1].

Entity Type Embedding. Similarly, we limit the sentences to a fixed length n. However, we only label entities which have been annotated in corpus with specific symbols. Other non-entity words are labelled as 0. Thus, the entity representation as follows: $E = \{0, ...e_i, 0, ..., e_j, 0, ...\}$, where e_i is the i-th entity and e_j is the j-th entity. Then, we look up the entity type embedding table (initialized randomly). The result of the entity type embedding is w_e.

Relation Type Embedding. It is specially used to characterize the embedding of the ED model using relationship between two entities. For the sentence that has fixed length n, we set each word as 0 expect when there exists a word which is an entity e_i and is annotated relation type with another entity e_j in corpus, we set it as symbol r. In addition, we set the entity e_j as the same symbol with e_i. The relation type representation of this sentence is: $R = \{0, ..., r, 0, ..., r, 0...\}$. Similarly, we look up the relation type embedding table to get the relation type embedding w_r. We randomly initialize embedding vectors for each relation type (including the non-trigger type) and update them during training procedure.

We concatenate the above three embeddings as the input to the neural network. The process is shown in Fig. 1: $D = [w, w_e, w_r]$. We denote the final representation as $D = \{d_1, d_2, ...d_i, ...d_n\}$, where n is also the length of the sentence and d represents each word.

4.2 Attention Mechanism

In order to capture the information of important words as much as possible, and reduce the interference for modeling meaningless words, we leverage attention mechanism to the neural network. The specific attention structure is shown in Fig. 2. We follow Liu et al. [13] calculation method of attention value strictly. Given the sentence and its representation $D = \{d_1, d_2, ...d_i, ...d_n\}$, we treat each token as a candidate trigger, d_c represents the current candidate trigger. Firstly, we get the relatedness s_i between d_c and the other token representation d_i in the sentence by the following equation:

$$s_i = tanh({d_c}^{\mathrm{T}} W d_i + b) \tag{1}$$

[1] https://catalog.ldc.upenn.edu/LDC2008T19.

where W is the weight matrix and b is the bias. Then, we calculate the importance p_i of each token in the sentence relative to the current candidate trigger. Given all the importance weighs, we get the comprehensive information att_c conveyed by D regarding the candidate trigger by computing the weighted sum:

$$p_i = \frac{exp(s_i)}{\sum_{k=1}^{n} exp(s_k)} , \qquad att_c = \sum_{i=1}^{n} p_i * d_i \tag{2}$$

Furthermore, we come up with two attention strategies according to the different position of the attention mechanism in the model:

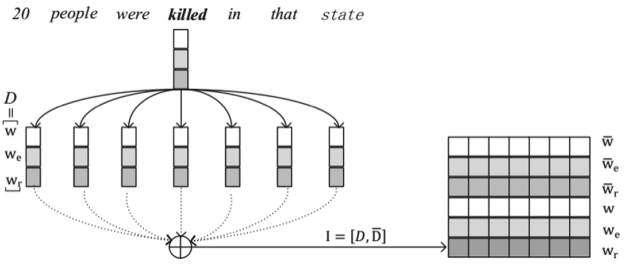

Fig. 2. Attention mechanism.

Att1: After obtaining the embedding D of the sentence, we apply the attention mechanism immediately. As shown in Fig. 2, through the above calculation method, we get the new embedding representation $\overline{D}$ after assigning attention weights. Then, we concatenate D and $\overline{D}$ as the input I to the Bi-LSTM.

Att2: We use embedding representation D as the input for Bi-LSTM firstly. And we utilize hidden output H of Bi-LSTM as the input for attention mechanism. Similarly, we can obtain the new hidden representation $\overline{H}$ after attention value calculation method. Then, we concatenate H and $\overline{H}$ for softmax.

4.3 Bi-LSTM

RNN with long short-term memory (LSTM) unit is adopted due to the superior performance in a variety of NLP tasks [12,14]. Furthermore, Bi-LSTM is a type of Bi-RNN, which can model word representation with its preceding and following information simultaneously.

Bi-LSTM consists of input gate, forget gate and output gate. At step t, input gate accepts a current input x_t, previous hidden state h_{t-1} and previous cell state C_{t-1} as Eq. 3. i_t controls how much new information can be conveyed to the cell state. $tildeC_t$ indicates new information at the current moment. Forget gate f_t controls how much information of the previous cell moment can be conveyed

to the current moment. C_t represents updated cell state. Output gate gets the current hidden state h_t of the step t as Eq. 5.

$$i_t = \sigma(W_i) \cdot [h_{t-1}, x_t] + b_i, \quad \tilde{C}_t = tanh(W_C \cdot [h_{t-1}, x_t] + b_c) \tag{3}$$

$$f_t = \sigma(W_f) \cdot [h_{t-1}, x_t] + b_f, \quad C_t = f_t * C_{t-1} + i_t * \tilde{C}_t \tag{4}$$

$$o_t = \sigma(W_o) \cdot [h_{t-1}, x_t] + b_o, \quad h_t = o_t * tanh(C_t) \tag{5}$$

The details of our Bi-LSTM architecture for ED are shown in Fig. 3. The figure shows the case of **Att1**, so the input of Bi-LSTM consists of six embedding representation: $\{w, w_e, w_r, \overline{w}, \overline{w_e}, \overline{w_r}\}$. We can see that Bi-LSTM is composed of two LSTM neural networks, a forward F-LSTM to model the preceding contexts, and a backward Bi-LSTM to model the following contexts. We can get the hidden representation h_f and h_b via running F-LSTM and B-LSTM respectively. Then, we concatenate the hidden embedding h_f and h_b as the output $H = \{h_1, h_2, ..., h_i, ...h_n\}$ of Bi-LSTM.

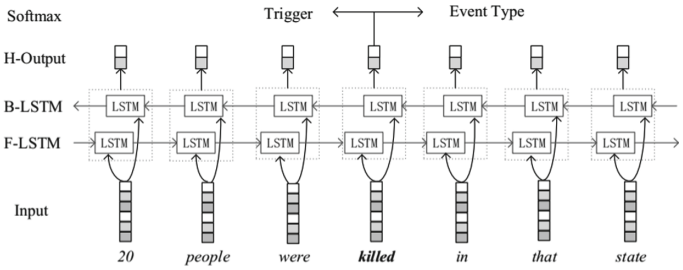

Fig. 3. Bi-LSTM architecture.

4.4 Output

AS mentioned earlier, we formulate ED as a multi-class classification problem. We predict each token of the sentence whether is an event trigger and assign event type to it. We use the hidden output H of the Bi-LSTM directly with **Att1** or combine H and $\overline{H}$ with **Att2** as the input to a softmax classifier. Thus, we can get the predicted probabilities of different types $P(y_j|x_i, \Theta)$, where Θ represents all the parameters of the model, and x_i is the i-th word and y_j is the j-th event type.

4.5 Training

We train the attention-based Bi-LSTM model by optimizing objective function which is defined as a multi-class cross-entropy loss:

$$L(\Theta) = -\sum_{k=1}^{N}\sum_{i=1}^{C} y_i log P(y_i|x_k, \Theta) + \lambda(\Theta) \tag{6}$$

where N denotes the all candidate triggers in the training process, C denotes the all event classes, and y_i is the real event type of the word x_i. λ is the regularization parameter and Θ represents the all parameters in the model.

In addition, we train the network via stochastic gradient descent (SGD) [18] with shuffled mini-batches. The gradients are computed using back propagation.

5 Experiments

5.1 Dataset and Evaluation Metric

We use the ACE-2005 corpus in the experiments. For comparison purpose, we follow Li et al. [10] to select 529 articles in English as the training data set, 30 as development set and 40 for test.

Following the previous work [9], we use Precision (P), Recall (R) and F_1 score (F_1) as the evaluation metrics of our approach.

5.2 Hyperparameter Settings

The word embeddings are initialized with the 300-dimensional real-valued vectors. The entity type embeddings are specified as the 50-dimensional real-valued vectors. And the relation type embeddings are initialized with the 20-dimensional vectors. We follow Feng et al. [6] to set the dropout rate and the batch size. Table 2 shows the specific setting of parameters used in our experiments.

Table 2. Hyperparameters used in our experiments.

Parameters	Values	Parameters	Values
Word embedding	300-dimensional	Dropout rate	0.2
Entity embedding	50-dimensional	Learning rate	0.3
Relation embedding	20-dimensional	Hidden size	200
Batch size	10	Coefficient	10^{-3}

5.3 Compared Systems

The state-of-the-art models proposed in the past are compared with ours. We divide the models into three classes:

Feature Based Approaches: ***Joint***: the method in Li et al. [10] which combines the local and global features and is based on the structured perceptron. ***Cross-Entity***: Hong et al. [8] model, which employs the name entities as the additional discriminant features to aid event extraction.

External Resource Based Approaches: Include Chen et al. [2] ***DMCNN-DS***, which utilizes world knowledge (Freebase) and linguistic knowledge (FrameNet) and Liu et al. [13] ***GMLATT*** which takes advantage of the multilingual information for the ED task.

Neural Network Based Approaches: ***DMCNN***: the method in Chen et al. [3], which uses CNN to do automatical feature extraction. In addition, also including Nguyen et al. [17] ***Bi-RNN***, Feng [6] ***Bi-LSTM***, Feng [6] ***Hybrid*** model which combines Bi-LSTM and CNN and Liu [15] ***ATT*** which exploited argument information to improve ED via supervised attention mechanisms.

5.4 Experimental Results

Table 3 shows the performance of all methods for both trigger identification and type classification. It can be observed that our approach outperforms other models, with a performance gain of no less than 0.5% $F1$ on event type classification. The performance mainly benefits from the higher recalls which are 78.5% and 76.3% in two subtasks respectively. In addition, comparing the three experiments that we did (the last three rows in the table), we found that no matter whether we merge relation or use attention mechanism, the performance has been improved to a small extent compared to using Bi-LSTM alone. But when we integrate two methods with Bi-LSTM, the performance will be greatly improved, which is 73.9%. The better performance of our approach can be further explained by the following reasons:

Table 3. Performance of the all methods (n/a: the paper did't list results of this task).

Methods	Trigger identification			Type classification		
	P	R	F1	P	R	F1
Joint [10]	76.9	65.0	70.4	73.7	62.3	67.5
Cross-Entity [8]	n/a	n/a	n/a	72.9	64.3	68.3
DMCNN-DS [3]	79.7	69.6	74.3	75.7	66.0	70.7
GMLATT [13]	80.9	68.1	74.1	78.9	66.9	72.4
DMCNN [2]	80.4	67.7	73.5	75.6	63.6	69.1
Bi-RNN [17]	68.5	75.7	71.9	66.0	73.0	69.3
Bi-LSTM [6]	80.1	69.4	74.3	81.6	62.3	**70.6**
ATT [15]	n/a	n/a	n/a	78.0	66.3	71.7
Hybrid [6]	80.8	71.5	75.9	84.6	64.9	**73.4**
Bi-LSTM+Att1 (Ours)	74.5	75.1	74.7	72.1	72.6	72.3
Bi-LSTM+Re (Ours)	72.9	77.5	75.1	69.6	74.1	71.8
Re+Bi-LSTM+Att1 (Ours)	73.7	**78.5**	76.1	71.5	**76.3**	**73.9**

- Compared with feature based methods, such as *Joint*, *Cross-Event* and *Cross-Entity*, neural network based methods, such as *CNN* and *Bi-LSTM*, perform better because they can make better use of word semantic information and avoid the errors propagated from NLP tools. Moreover, *Bi-LSTM* performs better than *CNN* due to the former can capture more complete information of the whole sentence, which reduce the loss of information.

- Table 4 lists embedding types used in each method. We can see that relation type can provide richer information for ED than position (*PSN*) and dependency (*DEP*). Because we merge the relation type embedding to the model, the recall has improved significantly, which is higher 11.4% than *Hybrid* (add *PSN* embedding) and higher 3.3% than *Bi-RNN* (add *DEP* embedding).
- Attention mechanism can make certain words get higher attention, capture more accurate information and ignore the interference of meaningless words.
- We would like to believe that using entity relation and attention simultaneously can enhance the performance further. Due to we use not only entity embedding but also relation embedding which labels two entities in a sentence with the same relation type label, when we employ attention mechanism in model, entities is equivalent to get twice attention. Accordingly, model can better capture the information of key words.

Table 4. Embedding types (PSN: Position; ET: Entity Type; DEP: Dependency; RT: Relation Type).

Methods	Embedding types	Methods	Embedding types
DMCNN-DS	word, PSN	ATT	word, ET
GMLATT	word, ET, PSN	DMCNN	word, PSN
Bi-RNN	word, ET, DEP	Bi-LSTM	word, PSN
Hybrid	word, PSN	**Ours**	word, ET, RT

In order to further prove the rationality of the above explanations, we conduct two extra experiments to do detailed analysis. We use ***TI*** and ***TC*** to stand for F_1 score of Event Trigger Identification and Event Type Classification respectively.

Effect of Different Features. We conduct the experiments with *Bi-LSTM+Att1* to exploit the effects of different feature combinations. We set the word embedding as the baseline, and then add entity embedding and relation embedding step by step. Results are shown in Table 5.

According to the result, we can find that both entity embedding and relation embedding can yield effectively improvement. It seems that, entity embedding is more effective than relation embedding (by 0.5%) in type classification. However, relation type embedding is more effective for trigger identification than the former (by 0.6%). An intuitive explanation is that: we label the entities which process relationship with the same relation type symbol as the relation type representation. Although the same labels provide complement information for trigger identification, it also causes interference to classify trigger.

Moreover, using all as embedding, *TI* and *TC* are 76.1% and 73.9% respectively, which integrates the advantages of entity and relation embedding, and reaches the optimal performance. Such as **S2**, entity embedding can capture the important information of "*Swenson*" and "*NCA*" rather than other words in sentence. And relation embedding will provide necessary information (*Membership* between two entities) to classify *End-Position* event correctly.

Table 5. Performance on different features.

Methods	Features	TI	TC
Bi-LSTM+Att1	word embedding	74.4	71.4
	+entity embedding	74.7	72.3
	+relation embedding	75.3	71.8
	all	76.1	73.9

Effect of Different Attention Strategies. In order to verify whether the attention mechanism plays a critical role, and compare attention strategies, we designed three comparison experiments. Taking *Re+Bi-LSTM* as the baseline, we add *Att1* and *Att2* on the basis respectively. Notes that all three methods combine word, entity and relation embedding. The results are shown in Table 6.

From Table 6, it can be observed that $F1$ score reduces 1.3% on event type classification relative to the baseline when we place attention mechanism after Bi-LSTM (+Att2). However, when we add attention mechanism before entering the Bi-LSTM model (+Att1), the $F1$ score improves 2.1% compared to baseline. This may because: although Bi-LSTM can capture sentence information as much as possible, it still can't avoid the loss of some parts of the information, or change the importance of each word in the original sentence. Thus, employing attention after the Bi-LSTM will reduce the information of the incomplete sentence again. By contrast, applying the attention mechanism before Bi-LSTM is equivalent to process sentences with original complete information, which can improve the importance of keywords and reduce the interference of meaningless words. Hereafter, Bi-LSTM model further selects effective information of the sentence to capture the key information perfectly. Thus, +Att1 can effectively improve the performance of ED system.

Table 6. Performance on different attention strategies.

Methods	TI	TC
Re+Bi-LSTM	75.1	71.8
Re+Bi-LSTM+Att1	76.1	73.9
Re+Bi-LSTM+Att2	73.6	70.5

6 Conclusions

In this paper, we verified that integrating an attention mechanism before the Bi-LSTM neural network, which can assign different attention to words and better capture the key information of sentences. Furthermore, we first use the entity relation as the feature for ED, and we confirmed it can provide additional information for the ED task. In the future, we will further explore the relationship between entity relation and ED, to unite them into a supporting model.

Acknowledgements. This work was supported by the national Natural Science Foundation of China via Nos. 2017YFB1002104, 61672368 and 61672367.

References

1. Ahn, D.: The stages of event extraction. In: Proceedings of the Workshop on Annotating and Reasoning about Time and Events, pp. 1–8. Association for Computational Linguistics (2006)
2. Chen, Y., Liu, S., Zhang, X., Liu, K., Zhao, J.: Automatically labeled data generation for large scale event extraction. In: Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), vol. 1, pp. 409–419 (2017)
3. Chen, Y., Xu, L., Liu, K., Zeng, D., Zhao, J., et al.: Event extraction via dynamic multi-pooling convolutional neural networks. In: ACL (1), pp. 167–176 (2015)
4. Collobert, R., Weston, J.: A unified architecture for natural language processing: deep neural networks with multitask learning. In: Proceedings of the 25th International Conference on Machine Learning, pp. 160–167. ACM (2008)
5. Doddington, G.R., Mitchell, A., Przybocki, M.A., Ramshaw, L.A., Strassel, S., Weischedel, R.M.: The Automatic Content Extraction (ACE) program-tasks, data, and evaluation. In: LREC, vol. 2, p. 1 (2004)
6. Feng, X., Huang, L., Tang, D., Ji, H., Qin, B., Liu, T.: A language-independent neural network for event detection. In: Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers), vol. 2, pp. 66–71 (2016)
7. Ghaeini, R., Fern, X., Huang, L., Tadepalli, P.: Event nugget detection with forward-backward recurrent neural networks. In: Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers), vol. 2, pp. 369–373 (2016)
8. Hong, Y., Zhang, J., Ma, B., Yao, J., Zhou, G., Zhu, Q.: 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, pp. 1127–1136. Association for Computational Linguistics (2011)
9. Ji, H., Grishman, R.: Refining event extraction through cross-document inference. In: Proceedings of ACL-08: HLT, pp. 254–262 (2008)
10. Li, Q., Ji, H., Huang, L.: Joint event extraction via structured prediction with global features. In: ACL (1), pp. 73–82 (2013)
11. Liao, S., Grishman, R.: Using document level cross-event inference to improve event extraction. In: Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, pp. 789–797. Association for Computational Linguistics (2010)
12. Lin, Z., et al.: A structured self-attentive sentence embedding. arXiv preprint arXiv:1703.03130 (2017)
13. Liu, J., Chen, Y., Liu, K., Zhao, J.: Event detection via gated multilingual attention mechanism. Statistics **1000**, 1250 (2018)
14. Liu, P., Qiu, X., Chen, J., Huang, X.: Deep fusion LSTMs for text semantic matching. In: Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), vol. 1, pp. 1034–1043 (2016)
15. Liu, S., Chen, Y., Liu, K., Zhao, J.: Exploiting argument information to improve event detection via supervised attention mechanisms, vol. 1, pp. 1789–1797 (2017)

16. Mikolov, T., Chen, K., Corrado, G., Dean, J.: Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781 (2013)
17. Nguyen, T.H., Cho, K., Grishman, R.: Joint event extraction via recurrent neural networks. In: HLT-NAACL, pp. 300–309 (2016)
18. Nguyen, T.H., Grishman, R.: Event detection and domain adaptation with convolutional neural networks. In: ACL (2), pp. 365–371 (2015)
19. Tang, D., Qin, B., Liu, T.: Document modeling with gated recurrent neural network for sentiment classification. In: Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pp. 1422–1432 (2015)
20. Turian, J., Ratinov, L., Bengio, Y.: Word representations: a simple and general method for semi-supervised learning. In: Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, pp. 384–394. Association for Computational Linguistics (2010)