Event Detection without Triggers

Shulin Liu, Yang Li, Xinpeng Zhou, Tao Yang, Feng Zhang
Tencent AI Lab
{forestliu, youngyli, xinpengzhou}@tencent.com
{rigorosyang, jayzhang}@tencent.com

Abstract

The goal of event detection (ED) is to detect the occurrences of events and categorize them. Previous work solved this task by recognizing and classifying event triggers, which is defined as the word or phrase that *most clearly* expresses an event occurrence. As a consequence, existing approaches required both annotated triggers and event types in training data. However, triggers are nonessential to event detection, and it is time-consuming for annotators to pick out the "*most clearly*" word from a given sentence, especially from a long sentence. The expensive annotation of training corpus limits the application of existing approaches. To reduce manual effort, we explore detecting events without triggers. In this work, we propose a novel framework dubbed as **T**ype-aware **Bi**as **N**eural **N**etwork with **A**ttention **M**echanisms (TBNNAM), which encodes the representation of a sentence based on target event types. Experimental results demonstrate the effectiveness. Remarkably, the proposed approach even achieves competitive performances compared with state-of-the-arts that used annotated triggers.

1 Introduction

This work tackles the task of event detection (ED), whose goal is to detect the occurrences of predefined events and categorize them. For example, consider the following sentence "*In Baghdad, a cameraman died when an American tank fired on the Palestine Hotel.*", an ideal event detection system should recognize two events, *Death* and *Attack*(suppose that both *Death* and *Attack* are in the predefined event set) .

Previous work typically solved this task by recognizing and classifying event triggers. According to ACE (Automatic Context Extraction) event evaluation program, event trigger is defined as the word or phrase that *most clearly* expresses an event occurrence. Take the following sentence as an example:

> ***S*:** In Baghdad, a cameraman **died** when an American tank **fired** on the Palestine Hotel.

"*died*" is the trigger word of *Death* event, and "*fired*" is the trigger word of *Attack* event. The majority of existing approaches modeled this task as word classification (Ji and Grishman, 2008; Liao and Grishman, 2010; Hong et al., 2011; Li et al., 2013; Nguyen and Grishman, 2015; Liu et al., 2016b,a; Chen et al., 2017), which predicted whether each word in a given sentence is an event trigger and what type of event it triggered. As a consequence, these approaches required both annotated triggers and event types for training.

However, event triggers are nonessential to this task. Remind that the goal of event detection is to recognize and categorize events, thus triggers could be viewed as intermediate results of this task. Furthermore, it is time-consuming for annotators to pick out the "*most clearly*" word from a given sentence, especially from a long sentence, which limits the application of existing ED approaches. To reduce manual effort, we explore detecting events without triggers. In this study, the only annotated information of each sentence is the types of events occurred in it. Consider the aforementioned example ***S*** again, its annotation is {*Death, Attack*}. On the contrast, previous work also required an annotated trigger for each event, which means the annotated information of ***S*** is {*Death*:died, *Attack*:fired} in previous work.

Without event triggers, it is intuitive to model this task via text classification. However, there are two challenges: (1) **Multi-label problem**: each sentence may contain arbitrary number of events, which means it could have zero or multiple target

Proceedings of NAACL-HLT 2019, pages 735–744
Minneapolis, Minnesota, June 2 - June 7, 2019.

labels. In machine learning, this problem is called multi-label problem. (2) **Trigger absence problem**: previous work illustrated that trigger words play important roles in event detection(Chen et al., 2015; Liu et al., 2016a). It is challenging to model this information without annotated triggers.

To solve the first challenge, we transform multi-label classification to multiple binary classification problems. Specifically, a given sentence s attached each pre-defined event type t forms an instance, which is expected to be labeled with 0 or 1 according to whether s contains an event of type t. For example, suppose there totally are 3 predefined types of events(denoted by t_1, t_2 and t_3), and sentence s contains two events of type t_1 and t_3, then it could be transformed to the following three instances:

instance	label
$< s, t_1 >$	1
$< s, t_2 >$	0
$< s, t_3 >$	1

Table 1: Example of instances in binary classifications for sentence s, which contains events of type t_1 and t_3.

In this paradigm, sentences that convey multiple events will yield multiple positive pairs, thus the multi-label problem could be well solved.

Furthermore, each type of events are usually triggered by a set of specific words, which are called event trigger words. For example, *Death* events are usually triggered by "die", "passed away", "gone", etc. Therefore, event trigger words are important clues to this task. Since existing work explicitly exploited annotated trigger words in their approaches, they can directly model this observation. However, in our case, annotated triggers are unavailable. To model this information, we propose a simple but effective model, called **T**ype-aware **Bi**as **N**eural **N**etwork with **A**ttention **M**echanisms (*TBNNAM*).

Figure 1 illustrates the framework of *TBNNAM*. The input is consisted of two parts: a tokenized sentence with NER tags and a target event type. The output o is expected to be 1 if the given sentence conveys an event of the target type, otherwise 0 (the output should be 1 for the example given in Figure 1). Specifically, given a sentence, the proposed model first transforms the input tokens into embeddings, and applies an LSTM layer to calculate a context-dependent representation for each token. Then it computes an attention vector, α, based on the target event type, where the trigger word is expected to obtain higher score. Finally, the sentence representation s_{att} is calculated based on α. Here, s_{att} is expected to focus on local information (trigger word). To capture global information, the final output, o, is also connected to the last LSTM units, which encodes the global information of the input sentence. Furthermore, to reinforce the influence of positive samples, we devise a bias objective function in our model . We call our model "type-aware" because the representation of a sentence, s_{att}, is calculated based on the target event type.

We have conducted experimental comparisons on a widely used benchmark dataset ACE2005[1]. The results illustrate that our approach outperforms all the compared baselines, and even achieves competitive performances compared with exiting approaches that used annotated triggers. We publish our code for further study by the NLP community.[2]

In summary, the main contributions of this work are: (1) To the best of our knowledge, this is the first work that focuses on detecting events without triggers. Compared with existing approaches, the proposed method requires less manual annotations. (2) Without triggers, this task encounters two challenges: multi-label problem and trigger absence problem. We propose a simple but effective model, which even achieves competitive results compared with approaches that using annotated triggers. (3) Since this is the first work on detecting events without triggers, we implement a series of baseline models for this task, and systematically evaluate and analyze them.

2 Background

2.1 Task Definition

Event detection task requires that certain specified types of events, which are mentioned in the annotated data, to be detected. The most common used benchmark dataset in previous work is ACE 2005 corpus. This corpus includes 8 types of events, with 33 subtypes. Following previous work(Ahn, 2006; Ji and Grishman, 2008; Liao and Grishman, 2010; Hong et al., 2011; Li et al.,

[1] https://catalog.ldc.upenn.edu/LDC2006T06

[2] https://github.com/liushulinle/event_detection_without_triggers

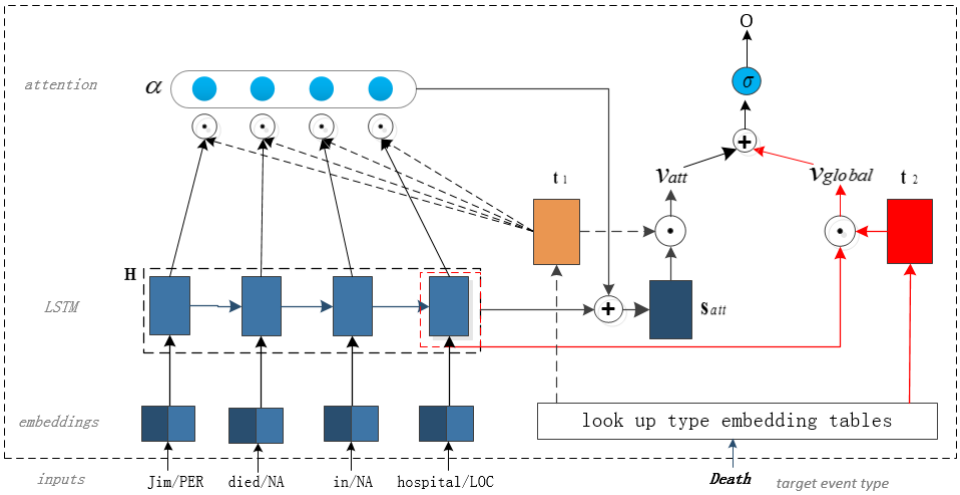

Figure 1: The framework of type-aware bias neural network with attention mechanisms. The input is consisted of two parts: a tokenized sentence with NER tags and a target event type. t_1 and t_2 are two different embedding vectors of the target event type. The output o is expected to be 1 if the given sentence conveys an event of the target type, otherwise 0 (the output should be 1 for the case in this figure). We call it "type-aware" because the representation of sentence, s_{att}, is calculated based on the target event type.

2013; Chen et al., 2015; Nguyen and Grishman, 2016), we treat them simply as 33 separate event types and ignore the hierarchical structure among them. Consider the following sentence "*In Baghdad, a cameraman died when an American tank fired on the Palestine Hotel*", an ideal event detector should detect two events from this sentence: a *Die* event and an *Attack* event.

2.2 Related Work

Event detection is one of important topics in NLP. Many approaches have been proposed for this task. Nearly all the existing methods on ACE event task follow supervised paradigm. We further divide them into feature-based methods and representation-based methods.

In feature-based methods, a diverse set of strategies has been exploited to convert classification clues into feature vectors. Ahn (2006) uses the lexical features(e.g., full word), syntactic features (e.g., dependency features) and external-knowledge features(WordNet (Miller, 1998)) to extract the event. Inspired by the hypothesis of One Sense Per Discourse (Yarowsky, 1995), Ji and Grishman (2008) combined global evidence from related documents with local decisions for the event extraction. To capture more clues from the texts, Gupta and Ji (2009), Liao and Grishman (2010) and Hong et al. (2011) proposed the cross-event and cross-entity inference for the ACE event task. Li et al. (2013) proposed a joint model to capture the combinational features of triggers and arguments. Liu et al. (2016b) proposed a global inference approach to employ both latent local and global information for event detection.

In recent years, representation-based methods have dominated the research. In this paradigm, candidate event mentions are represented by embeddings, which typically are fed into neural networks. Chen et al. (2015) and Nguyen and Grishman (2015) are the first work in this paradigm. Their models are based on CNNs (Convolutional Neural Networks). To model the dependency of triggers and arguments, Nguyen and Grishman (2016) proposed a joint event extraction approach based RNNs(Recurrent Neural Networks). Liu et al. (2017) proposed to encode argument information in event detection via supervised attention mechanisms. Recently, Nguyen and Grishman (2018) and Sha et al. (2018) proposed to exploit syntactic information for event detection.

All the existing approaches required annotated triggers. The expensive annotation of training data limits the application of these approaches. To reduce manual effort, we perform this task without event triggers.

3 Methodology

To deal the multi-label problem, we model this task via multiple binary classifications. Given a sentence, it will be fed into a binary classifier with each candidate event type. We add the label *NA* to

sentences that do not contain any events. To capture the hidden trigger information, we propose a simple but effective model, called **T**ype-aware **B**ias **N**eural **N**etwork with **A**ttention **M**echanisms (*TBNNAM*). Our model is "type-aware" because it calculates the representation of a sentence based on the target event type. Figure 1 illustrates the framework of *TBNNAM*. The input is consisted of two parts: a tokenized sentence with NER tags and a target event type. The output o is expected to be 1 if the given sentence conveys an event of the target type, otherwise 0. Next, we describe the structure of this model in bottom-up order.

3.1 Input Tokens

Given a sentence, we use Stanford CoreNLP tools[3](Manning et al., 2014) to convert texts into tokens. The ACE 2005 corpus annotated not only events but also entities for each given sentence. Following previous work, we exploit the annotated entity tags in our model(Li et al., 2013; Chen et al., 2015; Nguyen and Grishman, 2015, 2016; Liu et al., 2016b).

3.2 Word/Entity Embeddings

Word embeddings learned from a large amount of unlabeled data have been shown to be able to capture the meaningful semantic regularities of words(Bengio et al., 2003; Erhan et al., 2010). Much work(Socher et al., 2012; Zeng et al., 2014) has shown its power in many NLP tasks.

In this work, we use the Skip-gram model(Mikolov et al., 2013) to learn word embeddings on the NYT corpus[4]. Furthermore, we randomly initialized an embedding table for each entity tags. All the input word tokens and entity tags will be transformed into low-dimensional vectors by looking up these embedding tables. In this work, we denote the dimension of word embeddings by d_w, and that of entity embeddings by d_e.

3.3 Event Type Embeddings

As illustrated in Figure 1, an event type is transformed into two embedding vectors: $\mathbf{t}_1$ and $\mathbf{t}_2$. The first one (colored with brown) is designed to capture local information (hidden trigger word), and the latter one (colored with red) is designed to capture global information. Both of them are randomly initialized. The dimension of event type embeddings is denoted by d_{evt}.

[3]http://stanfordnlp.github.io/CoreNLP

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

3.4 LSTM Layer

As shown in Figure 1, the LSTM layer is run over the sequence of concatenation of word and entity embeddings. LSTM has three gates(input i, forget f and output o), and a cell memory vector c. The input gate can determine how incoming vectors $\mathrm{x(t)}$ alter the state of the memory cell. The output gate can allow the memory cell to have an effect on the outputs. Finally, the forget gate allows the cell to remember or forget its previous state.

3.5 Attention Layer

Each type of events are usually triggered by a set of specific words, which are called event trigger words. For example, *Death* events are usually triggered by "die", "passed away", "gone", etc. Therefore, event trigger words are important clues to this task. However, this information is hidden in our task, because annotated triggers are unavailable. To model the hidden triggers, we introduce attention mechanisms in our approach.

As illustrated in Figure 1, the attention vector α is calculated based on the target event type embedding $\mathbf{t}_1$ and the hidden states h yielded by LSTM. Specifically, the attention score for the k-th token in a given sentence is calculated by the following equation:

$$\alpha^k = \frac{\exp(\mathbf{h}_k \cdot \mathbf{t}_1^T)}{\sum_i \exp(\mathbf{h}_i \cdot \mathbf{t}_1^T)} \quad (1)$$

In this model, trigger words of the target event type are expected to obtain higher scores than other words. Finally, the representation of the sentence, $\mathbf{s}_{att}$, is computed by the following equation:

$$\mathbf{s}_{att} = \alpha^T \mathbf{H} \quad (2)$$

where $\alpha = [\alpha_1, ..., \alpha_n]$ is the attention vector, $\mathbf{H} = [\mathbf{h}_1, \mathbf{h}_2, ..., \mathbf{h}_n]$ is a matrix, $\mathbf{h}_k$ is the LSTM's output for the k-th token, and $\mathbf{s}_{att}$ is the representation of the given sentence.

3.6 Output Layer

As illustrated in Figure 1, the final output o is connected to two components: v_{att} and v_{global}. On one hand, v_{att} is calculated by the dot product of $\mathbf{s}_{att}$ and $\mathbf{t}_1$, which is designed to capture local features (specifically, features about hidden trigger words). On the other hand, the last output of the LSTM layer, $\mathbf{h}_n$, encodes global information of the whole sentence, thus $v_{global} = \mathbf{h}_n \cdot \mathbf{t}_2^T$ is expected

to capture global features of a sentence. Finally, o is defined as the weighted sum of v_{att} and v_{global}:

$$o = \sigma(\lambda \cdot v_{att} + (1 - \lambda) \cdot v_{global}) \quad (3)$$

where σ is the Sigmoid function, $\lambda \in [0, 1]$ is a hyper-parameter for trade-off between v_{att} and v_{global}.

3.7 Bias Loss Function

We devise a bias loss function to reinforce the influence of positive samples because of the following reasons. 1) **positive samples are much less than negative samples.** In our approach, each training sample is a <sentence,event type> pair, whose label is 1 or 0 according to whether the given sentence conveys an event of type t. For example, we totally have 33 target event types, if a sentence only contains one event, then it will be transformed into 32 negative pairs and 1 positive pair. The majority of sentences convey at most two events, thus negative samples are much more than positive samples. 2) **positive samples are more informative than negatives.** A positive pair $< s, t >$ means that s conveys an event of type t, whereas negative pair means s does not convey any event of type t. Apparently, the former is more informative.

Given all of the (suppose T) training instances $(x^{(i)}, y^{(i)})$, the loss function is defined as follows:

$$J(\theta) = \frac{1}{T}\sum_{i=1}^{T}(o(x^i) - y^{(i)})^2(1 + y^{(i)} \cdot \beta) + \delta||\theta||^2 \quad (4)$$

where x is a pair consisted of a sentence and a target event type, $y \in \{0, 1\}$, θ is the parameter of our model and $\delta > 0$ is the weight of L2 normalization term. $(1 + y^{(i)} \cdot \beta)$ is the bias term. Specifically, the value of this term is 1 for negative samples ($y^{(i)}$ is 0) and 1 + β for positive samples ($y^{(i)}$ is 1), where $\beta \geq 0$.

3.8 Training

We train the model by using a simple optimization technique called stochastic gradient descent (SGD) over shuffled mini-batches with the Adadelta rule (Zeiler, 2012). Regularization is implemented by a dropout and L2 norm.

Given a instance x, the model assign it a label $\widetilde{y}$ according to the following equation:

$$\widetilde{y} = \begin{cases} 0 & o(x) < 0.5 \\ 1 & otherwise \end{cases} \quad (5)$$

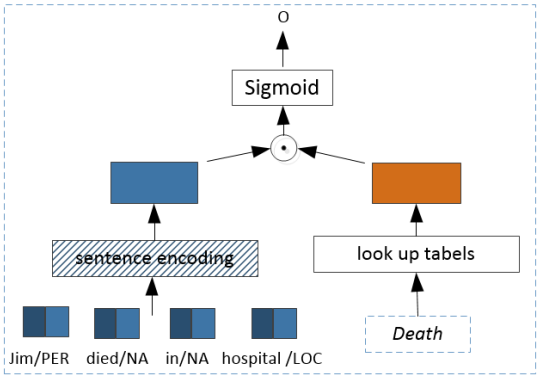

Figure 2: The framework of binary classification based approaches. The output o is expected to be 1 if the given sentence conveys an event of the target type, otherwise 0.

where x is a pair $< s, t >$, $o(x)$ is the output of the model for x, and $\widetilde{y}$ is the final predicted result.

4 Baseline Systems

Since this is the first work to perform event detection without triggers, we implement a series of baseline systems for comparisons, which could be divided into two categories: binary classification based methods and multi-class classification based methods.

4.1 Binary Classification

Similar with the proposed approach, baseline systems in this group solved this task via binary classification. Figure 2 illustrates the framework of these methods. These models take a sentence and a target event type as input. Then all the inputs are transformed into embeddings by looking up embedding tables. These models have the same loss function as the proposed approach (see Equation 4). The key component of these models is sentence encoder. According to the strategy of encoding sentence, we implement three models for comparison: *BC-CNN, BC-LSTM*$_{last}$, *BC-LSTM*$_{avg}$.

- *BC-CNN* employs a CNN model to encode sentence.
- *BC-LSTM*$_{last}$ employs LSTM model, and use the hidden state of the last token as the representation of a given sentence.
- *BC-LSTM*$_{avg}$ also employs LSTM model, but use the average of all hidden states as the representation of a given sentence.

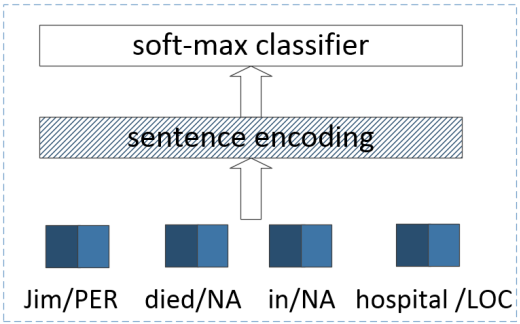

Figure 3: The framework of multi-class classification based approaches.

4.2 Multi-class Classification

All existing approaches model the task of event detection (with triggers) via multi-class classification[5]. Given a sentence, these methods predict whether each token is an event trigger and what type of event it triggered. We also implement several multi-class classification based systems for comparison. Since annotated triggers are unavailable in our task, the sentence is the input of our model. Figure 3 illustrates the framework of these models. Following existing work(Chen et al., 2015; Liu et al., 2017), we employ a negative log-likelihood loss function in the soft-max classifier: $J(\theta) = -\frac{1}{T}\sum_{i=1}^{T}\log(p(y^{(i)}|x^{(i)},\theta))$, where $(x^{(i)}, y^{(i)})$ is a training sample, $y^{(i)}$ is a label from the valid label set (all the predefined event types plus a *NA* for none event), T is the total number of training instances, θ is the parameters of the model. According to the strategy of encoding sentence, we implement three models: *MC-CNN*, *MC-LSTM*$_{last}$ and *MC-LSTM*$_{avg}$.

- *MC-CNN* employs a CNN model to encode sentence.
- *MC-LSTM*$_{last}$ employs LSTM model, and use the hidden state of the last token as the representation of a given sentence.
- *MC-LSTM*$_{avg}$ also employs LSTM model, but use the average of all hidden states as the representation of a given sentence.

[5]Multi-class classification means a classification task with more than two classes, but each sample belongs to only one class. "multi-class" is different from "multi-label".

5 Experimental Results

5.1 Experimental Setup

In this section, we introduce the dataset, evaluation metrics and the settings of hyper parameters.

5.1.1 Dataset

Our experiments are conducted on ACE 2005 dataset. Following the evaluation of previous work(Li et al., 2013; Chen et al., 2015; Nguyen and Grishman, 2016; Liu et al., 2017), we randomly selected 30 articles from different genres as the development set, and subsequently conducted a blind test on a separate set of 40 ACE 2005 newswire documents. We used the remaining 529 articles as our training set.

This work focuses on detecting events without triggers. Therefore, we remove trigger annotations from the corpus. Specifically, we employ Stanford CoreNLP Toolkit to split each document into sentences, and assign each sentence with a set of labels according to the original annotations in ACE 2005 corpus. If a sentence does not contain any event, we assign it with a special label, *NA*. If a sentence contains multiple events of the same type (less than 3% in ACE corpus), we only keep one label for each type. Table 2 shows several samples of the our corpus.

sentence	labels
They got married in 1985.	{*Marry*}
They got married in 1985, and divorced 3 years latter.	{*Marry*, *Divorce*}
They are very happy every day.	{*NA*}

Table 2: Examples of instances in our corpus (without event trigger annotations).

5.1.2 Evaluation Metrics

Following previous work (Liao and Grishman, 2010; Li et al., 2013; Chen et al., 2015; Liu et al., 2017), we use precision (P), recall (R) and F_1-measure (F_1) to evaluate the results.

Precision: the proportion of correctly predicted events in total predicted events.

Recall: the proportion of correctly predicted events in total gold events of the dataset.

F_1-measure: $\frac{2\times P\times R}{P+R}$

5.1.3 Hyper Parameters

Hyper parameters are tuned on the development dataset via grid search. In all experiments, we

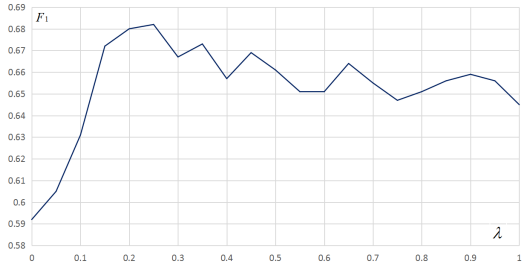

Figure 4: Experimental results on development dataset with different setting of λ.

methods	P(%)	R(%)	F_1(%)
MC-CNN	73.3	46.3	56.8
BC-CNN	**76.6**	**52.9**	**62.6**
MC-LSTM$_{last}$	57.9	42.3	48.9
BC-LSTM$_{last}$	**69.8**	**52.2**	**59.7**
MC-LSTM$_{avg}$	60.3	42.7	50.0
BC-LSTM$_{avg}$	**68.1**	**49.2**	**57.1**

Table 3: Experimental results on ACE 2005 corpus. Methods with name *MC-** are based on multi-class classification, and methods with name *BC-** are based on binary classification.

set the dimension of word embeddings as 200, the dimension of entity type embeddings as 50, batch size as 100, the hyper parameter for the L_2 norm as 10^{-5}, β in the bias term as 1.0. Furthermore, we also tune λ in Equation 3 on the development dataset. Figure 4 illustrates experimental results with different settings of λ, finally we set λ as 0.25. And in all the CNN-based baseline systems, the sizes of filter windows are set to 1, 2, 3 with 100 feature maps each.

5.2 Multi-class Classification vs. Binary Classification

Table 3 illustrates the experimental results, where methods with name *MC-** are based on multi-class classification, and methods with name *BC-** are based on binary classification. According to the strategy to encode a sentence, methods in Table 3 are grouped into three parts. From the table, we make the following observations:

- In each group, binary classification based approach significantly outperforms multi-class classification based approach. The reason is that *BC-** can solve the multi-label problem, but *MC-** can not. Moreover, *MC-** achieve much lower recall than *BC-**, because they predict at most one event for each sentence.

methods	P(%)	R(%)	F_1(%)
BC-CNN	76.6	52.9	62.6
BC-LSTM$_{last}$	69.8	52.2	59.7
BC-LSTM$_{avg}$	68.1	49.2	57.1
BC-LSTM$_{att}$	68.3	64.5	66.3
our TBNNAM	76.2	64.5	**69.9**
Nguyen's CNN†	71.8	66.4	69.0
Chen's DMCNN†	75.6	63.6	69.1
Liu's PSL†	75.3	64.4	69.4
DS-DMCNN‡†	75.7	66.0	70.5

Table 4: Experimental results on ACE 2005 corpus. Methods in the first group are baseline systems. Methods in the second group are the proposed approaches. Methods in the last group are state-of-the-art ED systems. † requiring annotated triggers, ‡ using external data

- Methods with CNN as sentence encoder achieve better performance than that with LSTM. The reason is that trigger words are important clues to event detection, and CNN is good at extracting such local features.

5.3 Overall Performances

In this section, we illustrates the results of the proposed approach (see Table 4). The results of baseline systems are listed in the first group. Methods in the second group are the proposed approaches. They have the same model structure as Figure 1. In *BC-LSTM$_{att}$*, λ (see Equation 3) is set as 1.0, which is designed to show the effects of the proposed attention strategy. In *TBNNAM*, λ is set as 0.25, which is designed to employ both local information (captured by the attention mechanism) and global information (captured by the last output of LSTM). Methods in the last group are state-of-the-art ED systems on ACE 2005 dataset. We give a brief introduction of them as follows:

1). *Nguyen's CNN*: the CNN model proposed by Nguyen and Grishman (2015)

2). *Chen's DMCNN*: the dynamic multi-pooling CNN model proposed by Chen et al. (2015)

3). *Liu's PSL*: the soft probabilistic soft logic model proposed by Liu et al. (2016b)

4). *DS-DMCNN*: the DMCNN model augmented with automatic labeled data, proposed by Chen et al. (2017)

From the table, we make the following observations:

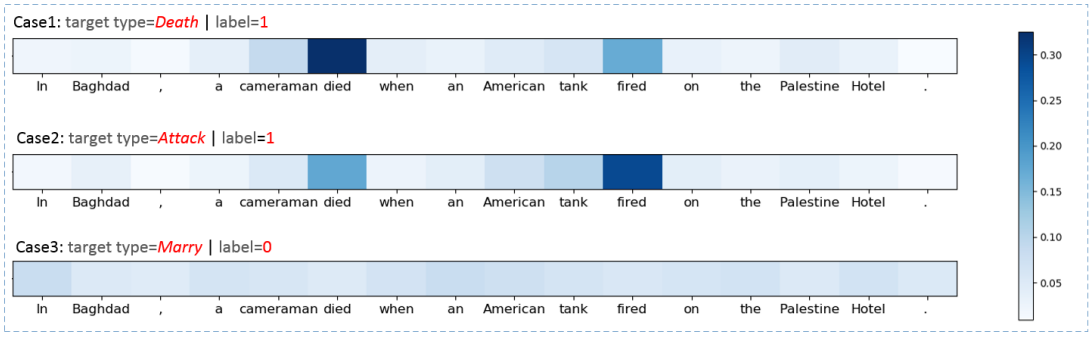

Figure 5: Visualization of attention weight vector α of sample instances learned by our model.

- *BC-LSTM$_{att}$* outperforms all the baseline systems with remarkable gains, which demonstrates the effectiveness of the proposed attention mechanism.

- *TBNNAM* achieves better performance than *BC-LSTM$_{att}$* (69.9% vs. 66.3%), which means that global information captured by the last state of LSTM is also important to this task. Such global information and local information captured by the attention mechanisms are complementary to each other.

- All state-of-the-art ED systems require annotated triggers. Without trigger annotations, our approach achieves competitive results, even outperforms some of them.

5.4 Analysis of Weight α

Figure 5 shows several examples of the attention vector α learned by our model. In the first case, "died" is the most significant keyword for the *Death* event, and our model succeeded to capture this feature by assigning it with a large attention score. Similarly, in the second case, "fired" is a key clue of *Attack* event, and our model also learned it and assigned it with a large attention score. Actually, "died" and "fired" are the trigger words of *Death* and *Attack* events, respectively. Therefore, we argue that, although annotated triggers are unavailable, our model still can exploit trigger information for this task. Moreover, our approach also could model the dependencies among different events, which has been demonstrated useful for this task(Liao and Grishman, 2010; Liu et al., 2016b). For example, *Attack* events often co-occur with *Death* events. In Case1 and Case2 (Figure 5), our approach models such information by paying attention on both words "died" and "fired". Furthermore, the 3-rd case is a negative sample, thus there is no key clues. Our model assigned each token with nearly equivalent attention score.

methods	P(%)	R(%)	F_1(%)
BC-LSTM$_{att}$\Bias	**74.5**	57.2	64.7
BC-LSTM$_{att}$	68.3	**64.5**	**66.3**
TBNNAM\Bias	**76.6**	59.8	67.2
TBNNAM	76.2	**64.5**	**69.9**

Table 5: Results of systems without/with bias term in loss function, where **Bias* do not use bias term.

5.5 Effects of Bias Term in Loss Function

In this section, we illustrates the effectiveness of the bias term in Equation 4. Table 5 shows experimental results. Methods named with "**Bias*" do not use bias term. From the table, we observe that systems with bias term in loss function significantly outperform those without bias term, which demonstrates the correctness of our analysis in Section 3.7 that positive samples should be reinforced during training.

6 Conclusions

Existing event detection approaches required annotated triggers, which limits their applications because of the expensive annotations. To reduce manual effort, we investigate performing this task without event triggers. In this setting, event detection task encounters two challenges: multi-label problem and trigger absence problem. We propose a simple but effective model to solve them, which computes the representation of a sentence according to the target event type. Experimental results demonstrate its effectiveness. Remarkably, the proposed approach even achieves competitive performances compared with state-of-the-arts that used annotated triggers.

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. Association for Computational Linguistics.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. 2003. A neural probabilistic language model. *The Journal of Machine Learning Research*, 3:1137–1155.

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 (Volume 1: Long Papers)*, pages 409–419, Vancouver, Canada.

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*, pages 167–176. Association for Computational Linguistics.

Dumitru Erhan, Yoshua Bengio, Aaron Courville, Pierre-Antoine Manzagol, Pascal Vincent, and Samy Bengio. 2010. Why does unsupervised pre-training help deep learning? *The Journal of Machine Learning Research*, 11:625–660.

Prashant Gupta and Heng Ji. 2009. Predicting unknown time arguments based on cross-event propagation. In *Proceedings of the ACL-IJCNLP 2009 Conference Short Papers*, pages 369–372. Association for Computational Linguistics.

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*, pages 1127–1136. Association for Computational Linguistics.

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

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)*, 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.

Shulin Liu, Yubo Chen, Shizhu He, Kang Liu, and Jun Zhao. 2016a. Leveraging framenet to improve automatic event detection. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, volume 1, pages 2134–2143. Association for Computational Linguistics.

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*, pages 1789–1798, Vancouver, Canada. Association for Computational Linguistics.

Shulin Liu, Kang Liu, Shizhu He, and Jun Zhao. 2016b. A probabilistic soft logic based approach to exploiting latent and global information in event classification. In *Proceedings of the thirtieth AAAI Conference on Artificail Intelligence*, pages 2993–2999. Association for Computational Linguistics.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*, pages 55–60.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

George Miller. 1998. *WordNet: An electronic lexical database*. MIT press.

Huu Thien 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*, pages 365–371. Association for Computational Linguistics.

Huu Thien 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. Association for Computational Linguistics.

Thien Huu Nguyen and Ralph Grishman. 2018. Graph convolutional networks with argument-aware pooling for event detection.

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 *AAAI*.

Richard Socher, Brody Huval, Christopher D Manning, and Andrew Y Ng. 2012. Semantic compositionality through recursive matrix-vector spaces. In *Proceedings of the 2012 joint conference on empirical methods in natural language processing*, pages 1201–1211. Association for Computational Linguistics.

David Yarowsky. 1995. Unsupervised word sense disambiguation rivaling supervised methods. In *33rd Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics.

Matthew D Zeiler. 2012. Adadelta: An adaptive learning rate method. *arXiv preprint arXiv:1212.5701*.

Daojian Zeng, Kang Liu, Siwei Lai, Guangyou Zhou, and Jun Zhao. 2014. Relation classification via convolutional deep neural network. In *Proceedings of COLING 2014*, pages 2335–2344.