Exploiting the Entity Type Sequence to Benefit Event Detection

Yuze Ji,[1,2] **Youfang Lin,**[1,2] **Jianwei Gao,**[1,2] **Huaiyu Wan**[1,2*]
[1]School of Computer and Information Technology, Beijing Jiaotong University, Beijing, China
[2]Beijing Key Laboratory of Traffic Data Analysis and Mining, Beijing, China
{yuzeji, yflin, gaojianwei, hywan}@bjtu.edu.cn

Abstract

Event Detection (ED) is one of the most important tasks in the field of information extraction. The goal of ED is to find triggers in sentences and classify them into different event types. In previous works, the information of entity types are commonly utilized to benefit event detection. However, the sequential features of entity types have not been well utilized yet in the existing ED methods. In this paper, we propose a novel ED approach which learns sequential features from word sequences and entity type sequences separately, and combines these two types of sequential features with the help of a trigger-entity interaction learning module. The experimental results demonstrate that our proposed approach outperforms the state-of-the-art methods.

1 Introduction

Event Extraction (EE) is one of the essential tasks of Information Extraction, which aims to extract structured events from unstructured texts. According to ACE (Automatic Context Extraction) event annotation guideline[1], an event is represented by an event trigger, which is often a single verb or noun, and a set of event arguments, the participants of the event. Event Detection (ED), as a crucial step in EE task, focuses on finding event trigger words and classifying them into different event types. As pointed out in (Liu et al., 2019; Ritter et al., 2012), the ambiguity in natural languages makes ED a challenging task. On the one hand, various expressions can be used to represent the same event type; on the other hand, the same event triggers, when placed in different context, can be categorized in totally different event types. To illustrate the second phenomenon, we present two examples from the widely used ACE 2005[2] dataset:

1) A Russian Soyuz capsule (VEH) **dropped** (*Transport*) the astronauts (PER) off this morning.

2) U.S. planes (VEH) **dropped** (*Attack*) a bomb (WEA) near northern Iraq.

Notice that the same annotated event trigger **dropped** has different meanings in the two sentences above and thus evokes entirely different event types. In the first sentence, **dropped** evokes a *Transport* event, but in the second sentence, **dropped** represents an *Attack* event. According to Liu et al. (2018a), in the ACE 2005 dataset, 57% of the event triggers are ambiguous. How to alleviate the ambiguity of event triggers has become a crucial problem in the ED task.

In several previous works, researchers have proved that entity mentions could play a positive role on alleviating the ambiguity of event triggers (Hong et al., 2011; Li et al., 2013; Feng et al., 2016; Liu et al., 2017, 2019, 2018a). An entity mention, as described in ACE 2005 dataset, is a reference to an object or a set of objects in the world. Back to the two sentences above, entity mentions are the underlined word tokens. In the first sentence, before classifying the event trigger **dropped**, if an event detector can obtain the information that "Soyuz capsule" is an entity mention with the type VEHICLE, and "astronauts" has the entity type PERSON, the event detector may tend to consider the potential link between VEHICLE and PERSON, which makes the event more likely to be a *Transport* event. In the second sentence, besides "planes" which is a VEHICLE type entity mention, another entity mention "bomb", which has the type WEAPON, can largely effect on the

*Corresponding author: hywan@bjtu.edu.cn
[1]https://www.ldc.upenn.edu/sites/www.ldc.upenn.edu/files/english-events-guidelines-v5.4.3.pdf
[2]https://catalog.ldc.upenn.edu/LDC2006T06

Proceedings of the 23rd Conference on Computational Natural Language Learning, pages 613–623
Hong Kong, China, November 3-4, 2019.

ED result. The appearance of WEAPON tells the event detector that there is a weapon in the context, and this clue leads the detector to recognize an *Attack* event with less ambiguity.

Liu et al. (2017) utilize raw entity types directly as local context of words to calculate the attention value between entity types and candidate triggers, aiming to catch the most important entity type information. After that, Liu et al. (2018b, 2019); Nguyen and Grishman (2018) apply entity types as a kind of supplementary information of the word tokens. They concatenate these two types of features and feed them into neural networks to learn mixed representations.

Intuitively, we understand that the sequential features of words are very important in modeling the sentence information, since the order of tokens can largely influence the meaning of the sentence. In the previous works, different neural network architectures have been employed to capture the sequential features from word sequences (Chen et al., 2015; Lin et al., 2018; Sha et al., 2018). Similar to the word sequences, entity type sequences, which consist of entity type annotations for each token in the word sequences, also contain sequential features, because the position of an entity mention's type in the sequences may affect its importance in the ED process. However, to the best of our knowledge, there is no study which regards the entity type sequence as an independent sequence to capture the sequential features and discusses what influence the entity types' sequential features would take to the ED task.

In order to make use of the information from both entity type sequences and word sequences, in this paper, we propose a novel ED approach Entity-Type-Enhanced-Event-Detection (**ETEED**). We consider that the word sequence and the entity type sequence have equal importance and thus the representation of each sequence should be learned separately. In this procedure, the Transformer Encoder structure (Vaswani et al., 2017) is utilized to capture the sequential features. Besides, an attention based trigger-entity interaction learning module is proposed to learn the correlation between triggers and entity types. Different from previous works which calculate the attention value between a candidate trigger and the whole entity type sequence, this module only learns the relation between entity mentions' types and the candidate trigger, and returns weighted summed entity mention type representations to benefit the classification. In this way, we can avoid the disturbance from irrelevant entity types, and focus only on the effect brought by entity mentions.

In summary, our contribution in this work is as follows:

- we propose to learn entity type representations separately from word representations, in order to make full use of the sequential features from entity type sequences.
- we propose an attention-based trigger-entity interaction learning module, which focuses only on the relation between entity mentions and candidate triggers, thus can eliminate the influence brought by irrelative entity types.
- we extensively evaluate our approach on a widely used benchmark dataset ACE 2005, and the evaluation result shows that our method can achieve competitive results compared with the state-of-the-art methods.

2 Approach

In this section, we elaborate the proposed **ETEED** method. Similar to the existing works, we regard ED as a classification problem. Specifically, in each sample, there is a candidate trigger, and our goal is to classify this candidate trigger into 34 event types (33 event subtypes and a NA type). We present the overall framework of our method in Figure 1. To well illustrate our model, we divide this section into three different parts: i) token representation learning, which involves representing the word sequences and entity type sequences, ii) attention-based feature learning, which interprets how the attention module works to learn the relation between entity mentions and candidate triggers, and iii) trigger classification, which concatenates all the continuous representation together and produce the final output for the trigger classification.

2.1 Token Representation Learning

For each sample with length n in the dataset, we represent its word sequence as $w = \{w_1, w_2, .., w_n\}$, in which w_i means the i-th word in the sentence. Similarly, let $e = \{e_1, e_2, ..., e_n\}$ be the entity types

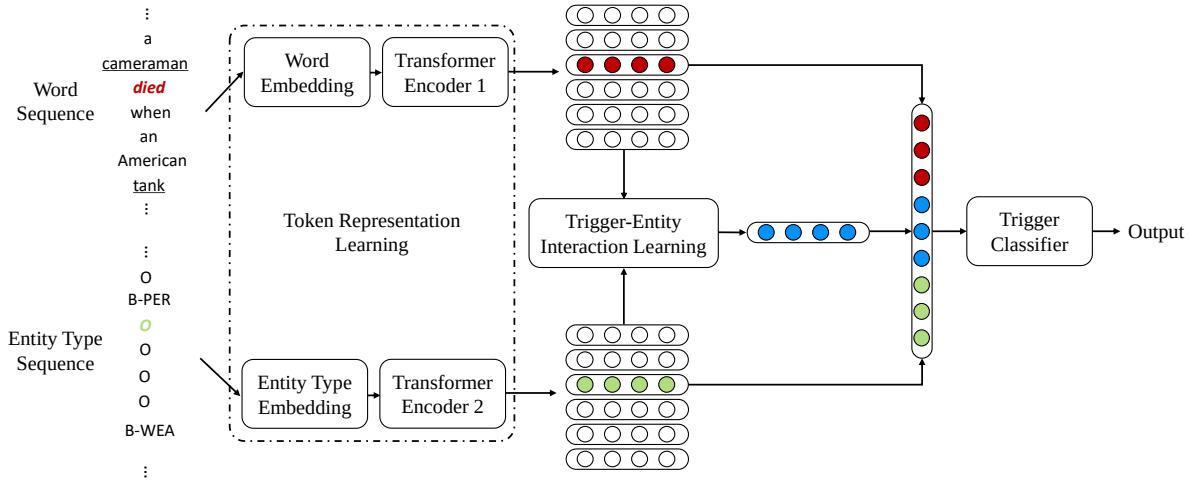

Figure 1: Global structure of our method.

corresponding to w. In consideration that cross-word entity mentions occur frequently in the ACE 2005 dataset, we apply BIO annotation schema to assign entity types for each token in the word sequence. Besides, we use c to represent the position of a candidate trigger in the sequence, so w_c and e_c show the word and entity type information of the candidate trigger respectively. In order to learn vector representations in local semantic context for the word and entity type tokens, firstly, we use embedding layers to transform the symbolic representations w and e to real-value vectors. Then, the Transformer Encoders are applied to capture the semantic relation between tokens, and learn a specific vector representation for each token.

Embedding Layer

With the help of the embedding layer, we transform the word token w_i and entity type token e_i in the input sequences into real-value representations. By looking up pre-trained word embedding matrix for w_i, a fixed sized vector representation x_{w_i} can be obtained. On the other side, for embedding entity type tokens, following existing works (Li et al., 2013; Chen et al., 2015; Liu et al., 2017; Nguyen and Grishman, 2015), we randomly initialize the real-value representation for each entity type and update it during the training process. The vector representation of e_i is marked as x_{e_i}.

Transformer Encoder Structure

Proposed by Vaswani et al. (2017), the Transformer has proved its effectiveness on the machine translation task. Different from most neural network based machine translation models (Cho et al., 2014; Bahdanau et al., 2015; Gehring et al., 2017), the Transformer is solely based on attention mechanisms, dispensing with recurrences and convolutions entirely. One of the most important reasons is that it is easier for attention mechanisms to learn long-range dependencies, which is a key challenge in sequential data modeling, than recurrent and convolutional neural networks. The Transformer has an encoder-decoder structure, the encoder maps the input sequence to new continuous representations, then the decoder receives the representations and generates an output sequence. The encoder-decoder structure is suitable for the machine translation task, however, in our approach, we only need to produce token representations for input sequences based on their local context. So, only the Transformer Encoder is used in our model.

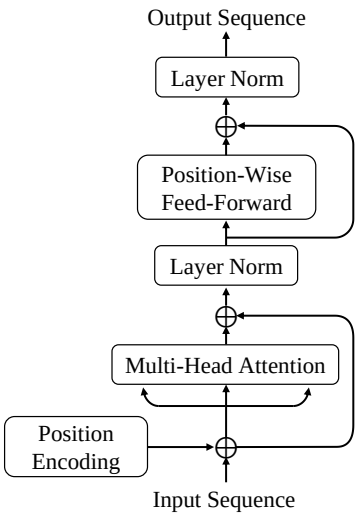

Figure 2: Architecture of the Transformer Encoder.

Figure 2 shows the architecture of the Transformer Encoder employed in our approach. For an input sequence consists of vector representations $\{x_1, x_2, ..., x_n\}$ with $x_i \in \mathbb{R}^d$, the Transformer Encoder produces the output sequence $\{z_1, z_2, ..., z_n\}$ of the same dimension with the input sequence. There are two sub-layers in the Transformer Encoder, one is a multi-head self attention layer, and the ohter is a position-wise feed-forward layer. Each sub-layer is followed by a residual connection (He et al., 2016) and a layer normalization (Lei Ba et al., 2016).

Based on single attention function, multi-head attention mechanism jointly captures the information from different representation subspaces. It firstly does h times different linear projections on the same input, then performs h single attention functions in parallel. Finally, in order to integrate all the information together, h output values from single attention functions are concatenated and projected to the same dimension with the input.

$$A_i(x) = softmax\left(\frac{\left(xW_i^Q\right)\left(xW_i^K\right)^T}{\sqrt{d_k}}\right) \quad (1)$$

$$head_i = A_i(x) \cdot \left(xW_i^V\right) \quad (2)$$

$$MultiHead(x) = [head_1, ..., head_h] \cdot W^O \quad (3)$$

where $x \in \mathbb{R}^{n\times d}$ is the input of the multi-head attention layer, $W_i^Q \in \mathbb{R}^{d\times d_k}$, $W_i^K \in \mathbb{R}^{d\times d_k}$, $W_i^V \in \mathbb{R}^{d\times d_v}$ are parameters to perform linear projections on input vectors, $W^O \in \mathbb{R}^{hd_v\times d}$ projects the concatenation of h single attention results to the same dimension with x. d_k, d_v are hyper-parameters determining projection dimensions.

In addition to the multi-head attention layer, a position-wise feed-forward layer is employed to enhance the representation capability of the Transformer Encoder.

$$FFN(x') = relu\left(x'W_1 + b_1\right)W_2 + b_2 \quad (4)$$

where $x' \in \mathbb{R}^{n\times d}$ is the input of the feed-forward layer, $W_1 \in \mathbb{R}^{d\times d_{hidden}}$, $b_1 \in \mathbb{R}^{d_{hidden}}$, $W_2 \in \mathbb{R}^{d_{hidden}\times d}$, $b_2 \in \mathbb{R}^d$, in which d_{hidden} is a hyper-parameter. Besides, since the Transformer Encoder contains neither recurrence nor convolution, in order to utilize the order information of sequence, position encodings are added to the input sequence at the beginning of the Transformer Encoder. The position encodings are calculated with the following equations:

$$PE(pos, 2i) = sin\left(pos/10000^{2i/d}\right) \quad (5)$$

$$PE(pos, 2i+1) = cos\left(pos/10000^{2i/d}\right) \quad (6)$$

where pos is the position and i is the dimension.

With the Transformer Encoder architecture described above, we obtain the vector representations from word sequence x_w and entity type sequence x_e separately, which will be marked as $z_w = \{z_{w_1}, ..., z_{w_n}\}$, $z_{w_i} \in \mathbb{R}^{d_w}$ and $z_e = \{z_{e_1}, ..., z_{e_n}\}$, $z_{e_i} \in \mathbb{R}^{d_e}$ in the following paragraph.

2.2 Trigger-Entity Interaction Learning

After the token representation learning, we get two sequences z_w and z_e. In order to encode the interaction between entity types and the candidate trigger, we introduce a trigger-entity interaction learning module in this subsection. It is an attention-based module which calculates the attention factors between the candidate trigger and entity types. In this procedure, we notice that there are many tokens with type O in the entity type sequence, which may prevent the method from explicitly modeling the relation between entity mentions' types (non-O entity types) and candidate trigger. Inspired by Nguyen and Grishman (2018), in this paper, we exclusively calculate the relation between the type of entity mentions and the candidate trigger. Given the vector representation of the candidate trigger z_{w_c} and the entity type sequence z_e, the attention values will be calculate as:

$$z_{em} = \{z_{e_i} | 1 \le i \le n \text{ and } e_i \neq O\} \quad (7)$$

$$\alpha = \sigma\left(\frac{(z_{w_c}U_1)\cdot(z_{em}U_2)^T}{\sqrt{d_w}} + b_3\right) \quad (8)$$

$$\alpha' = softmax\left(\alpha\right) \quad (9)$$

where $z_{em} \in \mathbb{R}^{k\times d_e}$ with supposing that there are k entity mentions in the sample, $U_1 \in \mathbb{R}^{d_w\times d_w}$, $U_2 \in \mathbb{R}^{d_e\times d_w}$, $b_3 \in \mathbb{R}^k$. As mentioned by Vaswani et al. (2017), to counteract the effect that the large values of d_w may impact the softmax result, we employ a coefficient $\frac{1}{\sqrt{d_w}}$ to scale the dot product.

Finally, with the help of the attention values α' and the entity mention type sequence z_{em}, we can calculate the vector representation for the trigger-entity interaction. Before doing the dot product

with α', in order to reinforce the capability of the model, we employ a fully connected layer on z_{em} with ReLU as activation function.

$$r = \alpha' \cdot relu(z_{em}U_3 + b_4) \quad (10)$$

where $U_3 \in \mathbb{R}^{d_e \times d_w}$, $b_4 \in \mathbb{R}^{d_w}$, and $r \in \mathbb{R}^{d_w}$ represents the output of trigger-entity interaction learning.

2.3 Trigger Classification

As illustrated in Figure 1, for each sample, we concatenate the trigger-entity interaction r with the candidate trigger z_{w_c} and the corresponding entity type representation z_{e_c}, to form the complete candidate trigger representation. Then, a multi-layer perceptron is employed as the trigger classifier to model the candidate trigger representation. The activation function SELU (Klambauer et al., 2017) is utilized in this multi-layer perceptron. Finally, the softmax function is applied to calculate the conditional probabilities that the candidate trigger belongs to each event type.

$$H_{c_1} = selu\left([z_{w_c}, z_{e_c}, r]W_{c_1} + b_{c_1}\right) \quad (11)$$
$$H_{c_2} = selu\left(H_{c_1}W_{c_2} + b_{c_2}\right) \quad (12)$$
$$O = softmax\left(H_{c_2}W_o + b_o\right) \quad (13)$$

where $W_{c_1} \in \mathbb{R}^{(d_e+2d_w)\times d_{c_1}}$, $W_{c_2} \in \mathbb{R}^{d_{c_1}\times d_{c_2}}$, $W_o \in \mathbb{R}^{d_{c_2}\times d_T}$ are weight metrics, in which d_{c_i} $(i \in \{1, 2\})$ are dimensions of the hidden states, d_T demonstrates the number of event types.

During the training procedure, we set the cross-entropy error as the loss function of our model, and the Adam optimizer (Kingma and Ba, 2015) is utilized to update the parameters. To keep the scale of gradients roughly the same in all layers, all the parameters are initialized by Xavier initializer (Glorot and Bengio, 2010). Table 1 shows the hyper-parameter settings in our experiments.

Module	Parameter	Value
Embedding	word	200
	entity	128
Transformer Encoder (Common)	head number	4
	layer number	1
	d_{hidden}	2048
Transformer Encoder (Word)	d_k	200
	d_v	200
Transformer Encoder (Entity)	d_k	128
	d_v	128
Trigger Classifier	d_{c_1}	256
	d_{c_2}	64
Adam Optimizer	lr	5e-5
	β_1	0.9
	β_2	0.999
	ϵ	1e-8

Table 1: Hyper-parameter settings in our experiments.

3 Experiments

3.1 Dataset

We evaluate our approach on a widely used benchmark dataset ACE 2005. In this dataset, event triggers are categorized into 33 subtypes (e.g., Be-Born, Marry, Attack). Besides, we annotate the candidate triggers which have no event types with the NA type. So, in total 34 event types are involved in our experiments. We also utilize the golden entity mentions annotated in ACE 2005 with BIO schema to produce the entity type sequences. Following the previous studies (Liu et al., 2019; Hong et al., 2018; Ji and Grishman, 2008), from the ACE 2005 English corpus, we choose randomly 40 newswire articles as the test set, 30 other articles as the development set, and pick the remaining 529 articles as the training set.

Following the ACE 2005's guideline document and Liu et al. (2019), we enumerate every noun, verb and adjective in sentences as candidate triggers. The NLP toolkit NLTK[3] is employed to parse and annotate the POS tags for sentences. We use pre-trained GloVe (Pennington et al., 2014) vectors as the embeddings for word tokens, and randomly initialize the embeddings for entity types then update them during the training procedure.

3.2 Overall Performance

We compare our ETEED model with the following state-of-the-art methods:

1) **JointBeam** is a feature-based method proposed by Li et al. (2013), which combines the manually designed local and global features to extract events.

2) **RBPB** is proposed by Sha et al. (2016), which simultaneously utilizes patterns and elaborately designed features to extract event triggers. In addition, a regularization method is applied to further improve the performance of the model.

3) **JRNN** is proposed by Nguyen et al. (2016), which combines the manually designed features with BiGRU to jointly extract triggers and arguments.

4) **HNN** is a language independent neural net-

[3]http://www.nltk.org/

Method	Trigger Identification(%)			Trigger Classification(%)		
	P	R	F1	P	R	F1
JointBeam	76.9	65.0	70.4	73.7	62.3	67.5
RBPB		N/A		70.3	67.5	68.9
JRNN	68.5	75.7	71.9	66.0	73.0	69.3
HNN*	**80.8**	71.5	75.9	**84.6**	64.9	73.4
SELF	75.3	78.8	77.0	71.3	74.7	73.0
DEEB-RNN		N/A		72.3	75.8	74.0
TEACHER		N/A		76.8	72.9	74.8
ETEED (ours)	78.1	**82.5**	**80.2**	74.1	**78.2**	**76.1**

Table 2: Overall performance with golden entity labels. * represents the model doesn't utilize entity type information.

work architecture proposed by Feng et al. (2016). With the structure which combines BiLSTM with CNN, this method can capture both the sequence and chunk information of words to benefit ED.

5) **DEEB-RNN** is proposed by Zhao et al. (2018), which incorporates the document-level clues with BiGRU to enhance ED.

6) **SELF** is proposed by Hong et al. (2018), which integrates BiLSTM into GAN structure, in order to distinguish the authentic information from spurious features.

7) **TEACHER** is an adversarial imitation based knowledge distillation approach proposed by Liu et al. (2019). This model contains two modules, one is a "teacher" module which combines the word sequences with the golden annotations of entity types and argument roles to learn knowledge representations. The other one is a "student" module which tries to imitate the representations from the "teacher" module.

We evaluate the performance via Precision (P), Recall (R) and F1-score (F1). Table 2 shows the overall performance of different approaches on the ED task. From the results, it can be found that our approach outperforms the state-of-the-art methods in both trigger identification and trigger classification. In the trigger identification, our approach achieves better results than all the previous methods in recall and F1-score (promote respectively 3.7% and 3.2% against the best baseline model SELF). Besides, although lower in precision by 2.8% , ETEED is 11% higher than the HNN model in recall. The same conclusion can be made in the trigger classification, our methods produces the highest recall and F1-score, and the improvement of F1-score is 1.3% over the best baseline model TEACHER. To summarize, on the one hand, ETEED significantly improves the recall values compared with the state-of-the-art methods. On the other hand, ETEED produces relatively comparable precisions to the existing methods, which ensures the good F1-score.

Method	Classification F1(%)
JointBeam	65.6 (↓1.9)
RBPB	67.8 (↓1.1)
TEACHER	71.2 (↓3.6)
ETEED (ours)	**74.8** (↓1.3)

Table 3: Performance with predicted entity labels. ↓ represents the performance drop from golden annotations.

To further verify the performance of our model in the real testing scenario, where the golden entity annotations are missing, we utilize the predicted entity type labels in the test procedure. Following Liu et al. (2019), we train a BiLSTM-CRF model on the training set, then apply it on the test set to get the predicted entity type sequences. The F1-score of the BiLSTM-CRF model on the test set is 82.7%. The JointBeam, RBPB and TEACHER are selected as baseline methods. Table 3 shows that our approach has significant improvement compared with the baseline methods. Besides, our model has relatively small performance descent with using predicted annotations than using golden annotations.

3.3 Effect of Entity Type Representation

In order to evaluate the effect of the entity type representation, we design the two following experiments:

Method	Trigger Identification(%)			Trigger Classification(%)		
	P	R	F1	P	R	F1
ETEED$_{no_entity}$	82.1	74.4	78.0	77.8	70.5	74.0
ETEED$_{concat}$	**83.2**	73.9	78.3	**78.4**	69.7	73.8
ETEED	78.1	**82.5**	**80.2**	74.1	**78.2**	**76.1**

Table 4: Effect of entity type sequence representation.

Method	Trigger Identification(%)			Trigger Classification(%)		
	P	R	F1	P	R	F1
ETEED$_{no_interaction}$	**82.6**	75.2	78.7	**79.3**	72.2	75.6
ETEED$_{all_entity}$	82.3	77.4	79.7	78.2	73.5	75.8
ETEED	78.1	**82.5**	**80.2**	74.1	**78.2**	**76.1**
TEACHER	N/A			76.8	72.9	74.8
ETEED$_{argument}$	87.4	82.9	85.1	**86.0**	**81.6**	**83.8**

Table 5: Effect of trigger-entity interaction.

1) **ETEED**$_{no\_entity}$ utilizes only word token sequences, which means there is no more entity information used in the model. For each sample, firstly the word sequence is encoded by the Transformer Encoder, then the candidate trigger's representation x_{w_c} is fed into the trigger classifier to get its event type.

2) **ETEED**$_{concat}$ follows some previous works (Liu et al., 2019, 2017), this model utilizes the concatenation of the word representations and the entity type representations as the input of downstream structures. In ETEED$_{concat}$, the concatenated representations are sent into the Transformer Encoder, then the features of the candidate triggers are picked from the output results to be classified by the trigger classifier.

The hyper-parameter settings keep the same with the ETEED. Table 4 shows that, our approach significantly outperforms the baseline models on F1-score (1.9% on identification and 2.1% on classification). In the meanwhile, our method has higher recall values but slightly lower precision values than the baseline models. This phenomenon shows that our method can largely improve the recall values with few precision loss. Especially, when compared with ETEED$_{concat}$, ETEED produces large performance gain on F1-score (1.9% and 2.1% on identification and classification respectively), which proves the effectiveness of our method. With the entity types' sequential features, our method can bring more entity information to trigger classifier than the baseline models.

3.4 Effect of Trigger-Entity Interaction

In this subsection, we conduct three comparison experiments to evaluate the effect of the trigger-entity interaction learning.

1) **ETEED**$_{no\_interaction}$ removes the trigger-entity learning from the complete model.

2) **ETEED**$_{all\_entity}$ uses all entity type information rather than the entity mentions' type information to learn trigger-entity interaction.

3) **ETEED**$_{argument}$ is a model designed to evaluate the effect of the argument role sequence. As mentioned in (Liu et al., 2019), TEACHER needs to utilize both the entity type and the argument role information to get the best performance. In the ETEED$_{argument}$, we replace the entity type sequences by the argument role sequences to compare with the TEACHER model.

Table 5 shows the results. Compared with ETEED$_{no_interaction}$, the complete ETEED model performs larger improvement on identification than classification on F1-score (1.5% to 0.5%). This result reveals that the use of entity mentions' types is obviously helpful when judging the occurrence of an event, but when it comes to trigger classification, the entity mention information brings less improvement. In order to explicitly capture the entity mention information, we only use entity mention features in our approach. To evaluate its effect, we compare ETEED$_{all_entity}$ with ETEED. Although the performance gain are

not significant, focusing on entity mentions can still bring 0.5% and 0.3% F1-score improvement on trigger identification and classification.

When comparing ETEED$_{argument}$ with ETEED, we can find that the use of argument role information can dramatically improve the performance of ED, because argument roles contain more information than entity types. Besides, some special argument roles can point to specific event types. As methods which use argument role information, ETEED$_{argument}$ significantly outperforms TEACHER, this result further proves the effectiveness of our model. However, in the real testing and application scenarios, we can hardly obtain the arguments role information before getting the event types of the candidate triggers. Instead, using predicted entity type information is more feasible.

4 Related Work

Event Detection is an important task in Information Extraction. The majority of existing approaches regard this task as a classification problem, and we summarize these approaches into two categories globally.

Feature-based methods are proposed as the first kind of approach to tackle the ED task by introducing feature-engineering to convert the classification clues like POS tags and dependency features into feature vectors (Ahn, 2006; Ji and Grishman, 2008; Hong et al., 2011; Li et al., 2013; Patwardhan and Riloff, 2009; Gupta and Ji, 2009; Liao and Grishman, 2010; Liu et al., 2016). This kind of approach depends heavily on expert knowledge and manual feature design, which makes these approaches time-consuming and low adaptability on different datasets. Chambers and Jurafsky (2011) design a weakly supervised system, which can automatically induce the event templates and extract event information from unlabeled corpus, to alleviate the need of expert knowledge. However, the required external resources are not always available for some low-resource languages.

In recent years, deep learning methods have proved their effectiveness on the ED task. (Chen et al., 2015; Nguyen and Grishman, 2016; Lin et al., 2018) utilize CNN to automatically capture the high-level vector representations of sentences. Nguyen et al. (2016) apply RNN in their model in order to capture the sequential features in the sentences. Feng et al. (2016) combine CNN with RNN and propose a hybrid neural network. Araki and Mitamura (2018) make use of the distant supervision mechanism to detect the events regardless of domains. Liu et al. (2017); Zhao et al. (2018) utilize attention mechanisms aiming to fuse the external sentence features (i.e. entity type features, document features) with the word features. (Hong et al., 2018; Liu et al., 2019) implement adversarial training to distinguish effective information from spurious features. GCN is a powerful neural network architecture on graphs, Nguyen and Grishman (2018) use this architecture to represent dependency relations in sentences. Compared with the feature-based methods, the deep learning methods need no more feature-engineering, which means less financial/time cost and better adaptability. Among them, there are several approaches which utilize the entity type information in their neural networks. Liu et al. (2017) utilize the entity type embedding directly as local context of the current word, and calculate the attention values between them; others (Liu et al., 2018b, 2019; Nguyen and Grishman, 2018) concatenate the entity type embeddings with the work token embeddings, in order to integrate these two types of features into mixed representations with the help of neural networks. However, these existing studies ignore the entity types' sequential features which may benefit the ED task.

In our approach, we learn the word features and the entity type features separately, which allows us to capture the sequential features of entity types and thus make full use of the entity type information. Besides, an attention-based trigger-entity interaction learning is introduced in our work to learn relations between the candidate trigger words and the entity mentions' type features.

5 Conclusion

In this work, we propose a novel neural network architecture ETEED for the ED task. In order to capture the sequential features from both the word sequences and the entity type sequences, our approach proposes to model these two types of sequences separately. The two types of sequences are firstly modeled by two isolated Transformer Encoders, then, an attention-based trigger-entity interaction learning module is applied to capture the correlations between the candidate trigger's word representation and the entity type representations of the sequence. Aiming to obtain a more

accurate interaction representation, this module utilizes only the entity mentions' type information rather than the whole entity type sequence to calculate the attention values. Finally, the concatenation of the candidate trigger's word, entity type representations and the trigger-entity interaction representation is fed into the trigger classifier to obtain the final event category. In the future, we plan to extend our method to the Event Extraction task, which means not only to extract triggers from sentences, but also to identify and classify the corresponding event arguments.

Acknowledgments

This paper is supported by the National Key R&D Program of China (No. 2018YFC0830200). We thank anonymous reviewers for their valuable comments.

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, Sydney, Australia. Association for Computational Linguistics.

Jun Araki and Teruko Mitamura. 2018. Open-domain event detection using distant supervision. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 878–891, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

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*, pages 976–986, Portland, Oregon, USA. 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, Beijing, China. Association for Computational Linguistics.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, Doha, Qatar. Association for Computational Linguistics.

Xiaocheng Feng, Lifu Huang, Duyu Tang, Heng Ji, Bing Qin, and Ting Liu. 2016. 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)*, pages 66–71, Berlin, Germany. Association for Computational Linguistics.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N Dauphin. 2017. Convolutional sequence to sequence learning. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1243–1252. JMLR. org.

Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256.

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, Suntec, Singapore. Association for Computational Linguistics.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

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, Portland, Oregon, USA. Association for Computational Linguistics.

Yu Hong, Wenxuan Zhou, Jingli Zhang, Guodong Zhou, and Qiaoming Zhu. 2018. Self-regulation: Employing a generative adversarial network to improve event detection. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 515–526, Melbourne, Australia. 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, Columbus, Ohio. Association for Computational Linguistics.

Diederik P Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Günter Klambauer, Thomas Unterthiner, Andreas Mayr, and Sepp Hochreiter. 2017. Self-normalizing neural networks. In *Advances in neural information processing systems*, pages 971–980.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. 2016. Layer normalization. *Computing Research Repository*, arXiv:1607.06450. Version 1.

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, Sofia, Bulgaria. Association for Computational Linguistics.

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, Uppsala, Sweden. Association for Computational Linguistics.

Hongyu Lin, Yaojie Lu, Xianpei Han, and Le Sun. 2018. Nugget proposal networks for Chinese event detection. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1565–1574, Melbourne, Australia. Association for Computational Linguistics.

Jian Liu, Yubo Chen, and Kang Liu. 2019. Exploiting the ground-truth: An adversarial imitation based knowledge distillation approach for event detection. In *AAAI*, Honolulu, Hawaii, USA.

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

Shulin Liu, Yubo Chen, Kang Liu, and Jun Zhao. 2017. Exploiting argument information to improve event detection via supervised attention mechanisms. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1789–1798, Vancouver, Canada. Association for Computational Linguistics.

Shulin Liu, Kang Liu, Shizhu He, and Jun Zhao. 2016. A probabilistic soft logic based approach to exploiting latent and global information in event classification. In *Thirtieth AAAI Conference on Artificial Intelligence*, pages 2993–2999. AAAI Press.

Xiao Liu, Zhunchen Luo, and Heyan Huang. 2018b. Jointly multiple events extraction via attention-based graph information aggregation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1247–1256, Brussels, Belgium. Association for Computational Linguistics.

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, San Diego, California. Association for Computational Linguistics.

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, Beijing, China. Association for Computational Linguistics.

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, Austin, Texas. Association for Computational Linguistics.

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*, pages 5900–5907. AAAI Press.

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*, pages 151–160, Singapore. Association for Computational Linguistics.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar. Association for Computational Linguistics.

Alan Ritter, Oren Etzioni, Sam Clark, et al. 2012. Open domain event extraction from twitter. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1104–1112. ACM.

Lei Sha, Jing Liu, Chin-Yew Lin, Sujian Li, Baobao Chang, and Zhifang Sui. 2016. RBPB: Regularization-based pattern balancing method for event extraction. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1224–1234, Berlin, Germany. Association for Computational Linguistics.

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 *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, pages 5916–5923.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5998–6008. Curran Associates, Inc.

Yue Zhao, Xiaolong Jin, Yuanzhuo Wang, and Xueqi Cheng. 2018. Document embedding enhanced event detection with hierarchical and supervised attention. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 414–419, Melbourne, Australia. Association for Computational Linguistics.