ISBN: 978-981-18-5852-9
WCSE 2022 Spring Event: 2022 9th International Conference on Industrial Engineering and Applications
doi: 10.18178/wcse.2022.04.180

Event Detection via Graph-based Multi-hop Neighbors' information Fusion

Chuan Li [1], Guoqiang Tian [2 +], Xintong Sun [3] and Fang Yang [4]

[1] Xi'an University of Posts and Telecommunications

[2] Xi'an University of Posts and Telecommunications

Abstract. Event detection (ED) proves to be a crucial subtask of event extraction. There is a close connection between event trigger word and its related neighboring words in dependency parse tree, neighbors' information has been widely used in event detection task. However, only neighbors' information with a direct arc (i.e., only one hop) to the trigger candidate in the dependency parse tree is used in many existing graph convolutional network (GCN)-based methods. As a result, the multi-hop neighbors' information is not fully utilized, so that it is difficult to further improve the performance of ED. Therefore, this paper proposes an ED model of graph convolutional network based on dependency parse tree. This model introduces graph attention network (GAT) to learn the multi-hop neighbors' information of each adjacent node in the syntactic graph, and uses a multi-label attention fusion mechanism to fuse the extracted multi-hop semantic information and reduce the complexity of the model. The results of experiments on the ACE2005 dataset show that compared with other methods based on graph convolutional neural networks, the F1-socre of this method reaches 75.6%, and the experimental results reflect the effectiveness of the method proposed in this paper.

Keywords: event detection, dependency parse tree, Graph convolution network, Graph attention network, attention fusion mechanism

1. Introduction

Event detection (ED) is defined as a key subtask of event extraction (EE) in natural language processing, which aims to recognize specified types of events in text. Each event mention in a sentence can be identified by a word or a phrase, called an event trigger, which is reckoned as the core word to express the occurrence of an event. The event detection task seeks to detect event triggers and have them classified into specific types of interest. Considering the following sentence as an instance: “Nathan divorced wallpaper salesman Bruce Nathan in 1992.” An efficient ED system should have the capacity of identifying the word “divorced” in the sentence as the trigger of event type Divorce.

Traditional feature-based methods mainly rely on manually designed features and sophisticated natural language processing tools. Their state-of-the-art methods usually utilize many elaborately designed features. Li *et al.* [1] use lexical and syntactic features to help detect events. Liao *et al.* [1] use document level information to address the problem that ambiguity in sentence-level event extraction depending on local information. Hong *et al.* [2] use cross-entity inference for event extraction. However, two problems may caused by feature-based methods: (1) The performance of feature-based methods rely on the quality of manually designed features. Once the quality of designed features is unqualified, the performance will be bad. (2) These manually designed features limits the generalization of feature-based methods because of their low reusability [3].

Some researchers focus on sequence-based methods by using deep neural networks. The existing groundbreaking approaches based on deep neural networks for ED are as follows. Nguyen *et al.* [4] firstly use convolutional neural networks (CNN) on event detection task, incorporating semantic and grammatical information of words, and employing type features for auxiliary ED. Chen *et al.* [5] propose a DMCNN model by constructing a dynamic multi-pooling mechanism to further improve the performance on ED.

+ Corresponding author. Tel.: + 13629277130.
E-mail address: tgq_123@stu.xupt.edu.cn.

Nevertheless, when handling the situation that a sentence consists of multiple events, CNN may lose some valuable information. RNN structures have the ability to capture long-term and short-term memory information. Nguyen *et al.* [6] propose an RNN-based model called JRNN, which utilizes a bidirectional RNN to learn rich context information. Feng *et al.* [7] proposed a hybrid neural network to capture both sequence and chunk information from contexts. Sha *et al.* [8] propose a novel method called dbRNN. They augment their model with dependency bridges and prove that applying both tree structure and sequence structure in RNN can achieve better performance than only use sequential RNN. With the rapid development of deep learning, graph neural networks (GNN) have been widely employed in ED, such as graph convolution networks (GCNs). Dependency parse trees can connect current word to its context to capture rich syntactic information for ED. GCN-based models have achieved excellent performance in ED by exploiting the dependency parse tree to efficiently learn the syntactic relations between the trigger candidate words and its related arguments. Nevertheless, many of the GCN-based models only use the information of the immediate neighbors in the dependency trees[9][10], which may ignore some long-range dependencies. For instance, considering the sentence "*Mohamad fired Anwar, his former protege, in1998.*" shown in Fig. 3, we need to recognize that the trigger word "*fired*" actually triggers an *End-Position* event instead of an *Attack* event by the phrase "*former protege*". Obviously, there are more than one hop to go from "*fired*" to "*former protege*". According to Yan's statistics [11], there are about 51% (4977/9793) sentences belong to this category.

One way to address the above problem is to stack multiple graph convolution layers. Nevertheless, according to the Liu's research [12], stacking multiple GCN layers will result in over-smoothing.

In this paper, we propose to tackle above problem by employing graph attention network (GAT) [13], which is an improved network of GCN. GAT can assign different importance to nodes of a same neighborhood in dependency graph. In order to make use of multi-hop neighbors' information of each node in dependency graph, we model multi-order representations of each word with its multi-hop neighbors' information by employing GAT. Subsequently, we introduce a multi-label attention mechanism to fuse the multi-order representations of each word which contains its multi-hop neighbors' information[14].

We evaluate our proposed method on the ACE 2005 dataset and widely compare our model to many previous state-of-the-art approaches. Experimental consequences show that the proposed method achieves the best performance among all comparison methods.

2. Task Description

Event detection (ED) is a key subtask of event extraction (EE). An event refers to something that occurs at a specific time and place, involving one or more participants [15]. Some terminologies of Automatic Content Extraction (ACE) are introduced to help understanding as follows:

- Entity: an object or a group of objects in a semantic category.
- Event mention: a phrase or sentence which can describe the event.
- Event type: the category of which the event corresponds.
- Event trigger: the core word which indicates the occurrence of an event, usually a verb or a noun.
- Event argument: the main properties of an event, including entities, nonentity participants, time, and so on.

Event detection aims to identify the event trigger which characterizes an event and to classify it into specific event type.

3. The Proposed Method

Following the previous studies [4], we cast the event detection as a multi-class classification task.

To be more specific, given a sentence, each word is associated with the context contained in the sentence to constitute an event trigger candidate. The goal of the ED is to predict the event type for each trigger candidate. A trigger candidate may be one of the event types which are pre-defined in the ACE 2005 dataset or NONE to represent it belongs to NONE class. To address the situation when an event trigger consists of more than one word, we employ the BIO annotation schema to assign a label for each word. The "BI" labels are used to represent the position information of each word in the event trigger. Tag “O” denotes the “Other” tag, which indicates that the corresponding word is irrelevant of the target events. So we have 2N+1 labels for ED where N is the number of pre-defined event types.

Our ED model consists of four modules as shown in Fig. 1: (i) Word embedding layer, which encodes each word in the input sentence as a continuous vector; (ii) Bi-GRU layer, which utilizes a Bidirectional Gated Recurrent Unit (Bi-GRU) to encode the semantics of each word in the forward direction and the backward direction, respectively; (iii) Multi-hop graph attention layer, in which we employ the graph attention network (GAT) to capture multi-order word representations with their multi-hop neighbors’ information in a sentence; (iv) Multi-label attention and Classification layer, in which we adopt the multi-label attention mechanism [14] to fuse the word representations with multi-hop neighbors’ information and then complete the classification task.

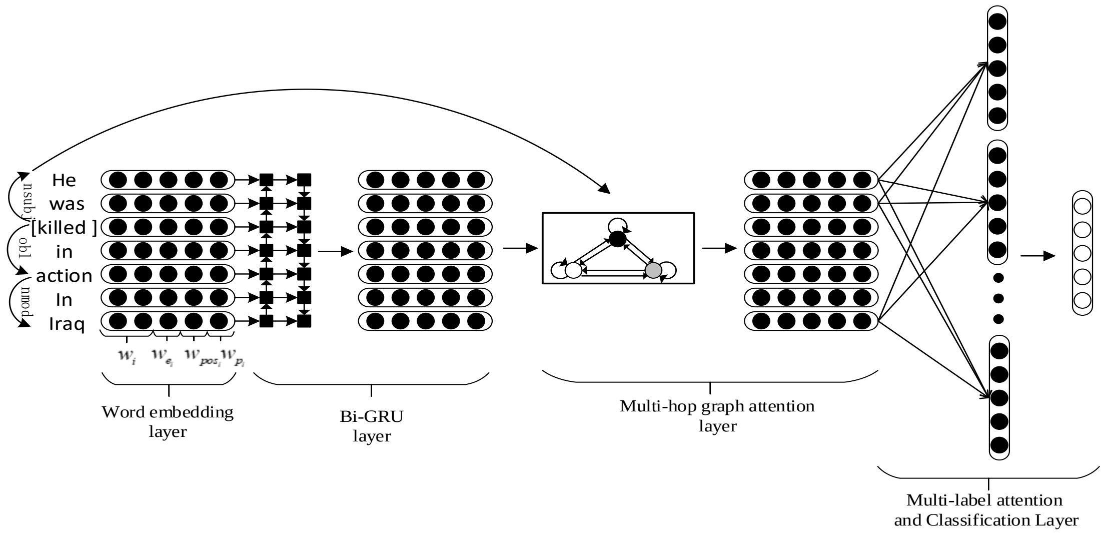
nsubj
dobj
nmod
He
was
[killed]
in
action
In
Iraq
w_i w_{e_i} w_{pos_i} w_{p_i}
Word embedding layer
Bi-GRU layer
Multi-hop graph attention layer
Multi-label attention and Classification Layer

Fig. 1: The architecture of our proposed model

3.1. Word Embedding Layer

In the word embedding layer, given a sentence of n words $W = \{w_1, w_2, ..., w_n\}$,we first transformed each word w_i into a real-valued vector x_i by concatenating the following vectors:

- The word embedding vector w_i: It is a real-valued vector which contains the semantic information of each word w_i. We employ the Skip-gram [16] model to train the word embeddings on the NYT corpus.
- The entity type embedding vector w_{e_i} : We adopt BIO annotation schema to annotate the entities mentioned in the sentence, and convert the entity type labels into real-valued random vectors by looking up an embedding table. Label “B-X” indicates the beginning of an entity mention of type “X”, the word with “I-X” label means the current word is inside of an entity mention of type “X”. The word with label “O” represents the current word does not belong to any entity mentions.
- The POS-tagging label embedding vector w_{pos_i} : POS-tagging represents the part-of-speech tagging. This is acquired by looking up an POS-tagging label embedding table which is randomly initialized.
- The position embedding vector w_{p_i} : Suppose w_t is the current word, we encode the relative distance $t-i$ between w_i and w_t as a real-valued vector to enhance the initial word representation x_i. Similar to the POS-tagging label embedding vector, the position embedding vector is also acquired by looking up the randomly initialized position embedding table which maps the relative positions into real-valued vectors.

Consequently, each word can be represented by the vector $x_i = [w_i : w_{e_i} : w_{pos_i} : w_{p_i}] \in R^d$,where $d = d_w + d_e + d_{pos} + d_p$, d_w , d_e , d_{pos} and d_p indicate the word embedding dimension, the entity type embedding dimension, the POS-tag embedding dimension and the position embedding dimension, respectively. Then the input sentence W can be mapped as a sequence of vectors $X = \{x_1, x_2, ..., x_n\}$ and X is used as input of the Bi-GRU layer in the following step.

3.2. Bi-GRU Layer

Gated Recurrent Unit (GRU) is a variant of simple RNN network, which can effectively solve the problems that the gradients vanish or explosion caused by simple RNNs [17]. Compared with the overly complex gated network structure of LSTM, the structure of GRU is relatively simple. By reducing the gated signal of LSTM to two, they are called update gate and reset gate, respectively. Therefore, compared with LSTM, GRU has fewer parameters, can achieve faster convergence speed, and is more suitable for ED task. Bi-GRU is a bidirectional GRU. It does not change the internal structure of the GRU, but applies the GRU twice from two different directions. It is used to capture some specific pre-features or post-features in the context to enhance the semantic association, and make up for the single insufficient GRU, so this paper uses Bi-GRU to capture contextual semantic features, and the GRU structure is shown in Fig. 2.

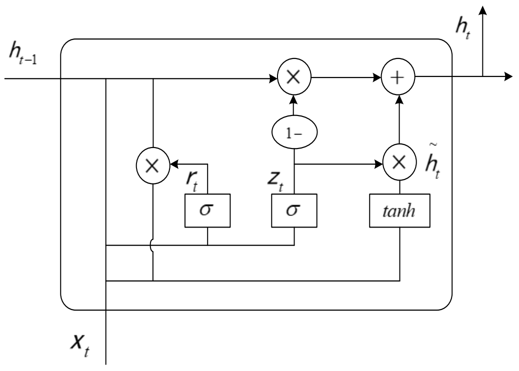

Fig. 2: The architecture of GRU network

Assume that the input at time step *t* is x_t, the calculation formulas of GRU are as follows:

$$z_t = \sigma(W_z \cdot [h_{t-1}, x_t] + b_z) \tag{1}$$

$$r_t = \sigma(W_r \cdot [h_{t-1}, x_t] + b_r) \tag{2}$$

$$\tilde{h}_t = \tanh(W_h \cdot [r_t \odot h_{t-1}, x_t] + b_h) \tag{3}$$

$$h_t = (1 - z_t) \odot h_{t-1} + z_t \odot \tilde{h}_t \tag{4}$$

where h_{t-1} denotes the hidden vector in the former moment, z_t is called an update gate and r_t is called a reset gate, respectively. h_t is the final output vector. σ is the nonlinear activation function Sigmoid, tanh is the hyperbolic tangent function. W_z, W_r and W_h represent weight matrices, b_z, b_r and b_h represent bias items. Here we adopt a forward GRU and a backward GRU and concatenate the hidden vectors of them, thus the final output of Bi-GRU is:

$$h_t = [\overrightarrow{GRU}(x_i) : \overleftarrow{GRU}(x_i)] \tag{5}$$

where $h_t \in R^{n \times 2d_h}$, which is the vector representation containing the contextual feature information of the sentence after Bi-GRU layer, and d_h is the dimension of a single direction of GRU.

3.3. Multi-hop Graph Attention Layer

This paper employs a dependency parse tree with its corresponding adjacency matrix for the input sentence so that we can make full use of the interactive information between each word and its neighbors to improve the performance of ED. According to the previous study [18], Each dependency parse tree can be transformed to a dependency graph which contains the nodes with first-order syntactic relations. Thus, only immediate neighbors' information in dependency graph is used. Let $G = (V, E)$ be the dependency parse tree of input sentence W, where $V = \{v_1, v_2, ..., v_n\}$ denotes the sets of nodes and E denotes the sets of edges, respectively. V consists of n nodes which represent n words $w_1, w_2, ..., w_n$ in W. Each edge (v_i, v_j) in E is a directed syntactic arc from word w_i to word w_j with its dependency label $L(v_i, v_j)$.

Consider sentence "*Mohamad fired Anwar, his former protege, in 1998.*" as an example, whose corresponding dependency parse tree is shown in Fig. 3, there exists a directed arc from the node of the word w_i = "*fired*" to the node of the word w_j ="*Mohamad*" with the type label $L(v_i, v_j) = L(\text{"}fired\text{"},\text{"}Mohamad\text{"}) = nsubj.$

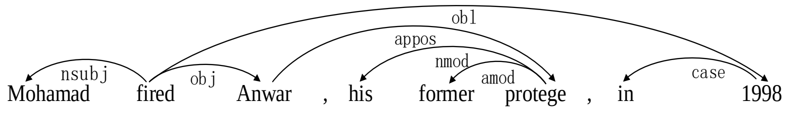

Fig. 3: An example of dependency parse result

In order to overcome the shortcomings of one-way transmission of information in the initial dependency graph, we append the reverse edges (v_j, v_i) and the self-loop edges (v_i, v_i) into the original edge sets E [19]. Thus, we have another two kinds of type labels $L(v_j, v_i) = L'(v_i, v_j)$ and $L(v_i, v_i) = loop$ .We use E' to represent the new edge sets. Then the new dependency graph can be denoted as $G' = (V, E')$. We use A to represent the adjacency matrix of the dependency graph.

Each adjacency matrix A can be divided into three sub-matrices of $n \times n$ dimensions: A_{along} , A_{rev} and A_{loop} , respectively. The definitions of three sub-matrices are as follows:

$$A_{along}(i, j) = \begin{cases} 1, (v_i, v_j) \in E \\ 0, (v_i, v_j) \notin E \end{cases} \tag{6}$$

$$A_{rev} = A_{along}^T \tag{7}$$

$$A_{loop} = I \tag{8}$$

where *I* represent an identity matrix. We represent the dependency graph's adjacency matrix with *m*-th-hop neighbors' information as $A_{hop}^m = (A_{hop})^m$,where *hop* can be *along*, *rev*, or *loop*. A_{hop}^m contains *m*-hop paths of A_{hop} . For convenience, we denote A_{along}^m , A_{rev}^m and A_{loop}^m as a^m , r^m , l^m , respectively. To fully utilize the multi-hop neighbors' information of each node, we employ several parallel GAT layers to learn multi-hop syntactic relation. Aim to get semantic information with stronger expressive ability, a linear transformation is applied to each input node to convert the input feature into a higher-level semantic feature, and then the self-attention mechanism f is executed on each node to calculate the attention coefficient, as formula (10) shows:

$$e_{ij} = f(W_h h_i, W_h h_j) \tag{9}$$

where the attention mechanism *f* can be regarded as a single-layer feedforward neural network. It is used to learn the association between node pairs as follows:

$$e_{ij} = \varphi(W_a[W_h h_i \,||\, W_h h_j]) \tag{10}$$

where φ is the Nonlinear activation function LeakyReLU (with negative input slope α = 0.2), W_a and W_h are weight matrices. e_{ij} can reflect the importance of node *j*'s features to node *i*. For convenience of making the coefficients easier to compare between different nodes, we apply the softmax function to normalize all options of j :

$$s_{ij} = \text{softmax}(e_{ij}) = \frac{\exp(e_{ij})}{\sum_{j \in \mathrm{N}_i} \exp(e_{ij})} \tag{11}$$

where s_{ij} will be the normalized coefficient of the subsequent graph attention convolution function as follows:

$$g(h_i, a^m) = \sigma \sum_{j=1}^{n} (s_{ij} a_{ij}^m (W_{a,m} h_j + b_{a,m})) \tag{12}$$

and N_i is the set of some neighbors of node *i* in the graph. In formula(13), σ is nonlinear activation function ELU [20], $W_{a,m}$ and $b_{a,m}$ are the weight matrix and bias vector of a^m . We associate the three subgraphs' representations of A^m to calculate the word representations h_i^m with multi-hop neighbors' information as follows [11]:

$$h_i^m = g(h_i, a^m) \oplus g(h_i, r^m) \oplus g(h_i, l^m) \tag{13}$$

where $\oplus$ is element-wise addition.

3.4. Multi-label Attention and Classification Layer

After obtain multi-order representations h_i^m with multi-hop neighbors' information of each word w_i, we adopt a multi-label attention layer proposed in AttentionXML [14] , which can make full use of the context information of the current word and capture the importance of the trigger candidate for each classification label by fusing the representations h_i^m:

$$h_i = \sum_{m=1}^{M} \alpha_{im} h_i^m \tag{14}$$

where α_{im} is the normalized coefficient of h_i^m, which is calculated in formula(15). M is the highest hop of neighbors.

$$\alpha_{im} = \frac{\exp(W_{att} h_i^m)}{\sum_{j=1}^{M} \exp(W_{att} h_i^j)} \tag{15}$$

where W_{att} is the weight matrix, which is used here as the attention weight parameter. Finally, we feed the fused vector into a fully connected network and apply the softmax function to predict the trigger label for each trigger candidate in formula (16):

$$p_i^t = \mathrm{softmax}(W_2 \psi(W_1 h_i + b_1) + b_2) \tag{16}$$

where p_i^t denotes the final output of the *i*-th trigger label, ψ is a non-linear activation function RELU. $W_{1,2}$ and $b_{1,2}$ are trainable weight matrices and bias items, respectively.

3.5. Loss Function

Following the existed studies [18] [21], we employ a bias loss function $J(\theta)$ to intensify the effectiveness of event type labels during the training process for the reason that the number of tags "O" own a larger proportion than the number of event type labels. The bias loss function is calculated as follows:

$$\begin{aligned} J(\theta) = \max \sum_{i=1}^{N_{st}} \sum_{j=1}^{n_i} & (\log p(y_j^t \mid s_i, \theta) \cdot I(O) \\ & + \omega \log p(y_j^t \mid s_i, \theta) \cdot (1 - I(O))) \end{aligned} \tag{17}$$

where N_{st} is the number of sentences, n_i is the number of words in sentence s_i. $I(O)$ is a switching function which is used to distinguish the loss of tag "O" and event type labels, and is defined as follows:

$$I(O) = \begin{cases} 1, & if \quad tag = "O" \\ 0, & \text{otherwise} \end{cases} \tag{18}$$

where ω performs as a bias weight, it is proportional to the influence of event type labels on the model.

4. Experiments

4.1. Dataset and evaluation matrices

We evaluate our model on the extensively used ACE 2005 dataset, which consists of 599 documents annotated with 33 event types. Following the previous work [8], we divide ACE 2005 dataset into 3 portions: 40 documents (881 sentences) as the test set, 30 documents (1087 sentences) as the development set and the rest 529 documents (21090 sentences) as the training set. We employ the Stanford Core NLP toolkit for data preprocessing, including sentence splitting, tokenizing, pos-tagging and generating dependency parse trees.

Finally, we use Precision (P), Recall (R) and F1-score (F1) to evaluate our model as the same as previous studies.

4.2. Hyper-parameter setting

We tuned the hyper-parameter on the development dataset of the ACE 2005 dataset. Our selected values of the parameters include the word embedding vectors with dimension = 100. The rest three embeddings including event type embedding, POS-tag embedding and position embedding are randomly initialized. We set the dimension of event type embedding vectors, the dimension of POS-tag embedding vectors and the dimension of position embedding vectors to 50. The hidden state size of one-layer BiGRU is set to 250, the hop of neighbors M is set to 3 and the number of hidden units of the graph attention layers is 150. We perform the stochastic gradient descent algorithm and the AdaDelta update rule [22] to optimize parameters with learning rate = 0.001 and batch size = 20. We apply L2 regularization with a parameter of 1e-5 to avoid overfitting. The dropout rate is set to 0.5 and the bias parameter ω is set to 5. We also set the max length of sentence to 50 by padding the shorter sentences and cutting longer ones. We perform all the experiments by using Pytorch 1.5.0 on the Nvidia GeForce RTX 3090 GPU, with Intel Xeon Silver 4208 CPU.

4.3. Overall performance

In order to evaluate our proposed model comprehensively, we compare it with some state-of-the-art methods as follows:

1 MaxEnt is proposed by Li *et al.* [1], which exploits local features and global features to improve the performance significantly on EE task.

2 Cross-Event is proposed by Liao and Grishman [1], which uses the document level cross-event to improve the performance on ACE event extraction.

3 Cross-Entity is proposed by Hong *et al.* [2], which uses cross-entity inference to extraction events.

4 DMCNN is proposed by Chen *et al.* [5],which constructs a dynamic multi-pooling layer to reserve more crucial information.

5 DMCNN+ is also proposed by Chen *et al.* [23], which exploits argument information to improve event detection via supervised attention mechanisms.

6 JRNN is proposed by Nguyen *et al.* [6], which presents a joint model for event extraction by using a bidirectional RNN and manually designed features.

7 dbRNN is proposed by Sha *et al.* [8], which adds dependency bridges over BiLSTM for event extraction.

8 GCN-ED is proposed by Nguyen and Grishman [9], which firstly applies graph convolution network on event detection.

9 JMEE is proposed by Liu *et al.* [10], which introduce syntactic arcs and GCN with self-attention aggregation mechanism to improve the performance on event extraction.

10 BGCN is proposed by Cheng *et al.* [24], which utilize the BERT model to strengthen the feature representation and introduce GCN network with syntactic structure to capture long-distance dependencies to detect events.

Table 1 shows the overall performance of different models on the ACE 2005 dataset. According to the result of the table, our model achieves the best F1-score among all of the competing models. Our proposed model gets a significant improvement over feature-based models. Compared with three feature-based models, MaxEnt, Cross-Event and Cross-Entity, our method achieves 9.7%, 6.8%, 7.3% improvement on F1-score, respectively. We can also see that feature-based models cannot perform better than sequence-based models such as DMCNN, etc. It indicates that human-designed features are not effective enough for event detection while automatic feature extraction based on neural networks can capture richer semantic information. In addition, the proposed model could also outperform all sequence-based models with remarkable improvements of 6.5%, 5.1%, 6.3% and 3.7% on F1 score over the four sequence-based baseline model DMCNN, DMCNN+, JRNN and dbRNN, which suggests that graph-based models are able to make use of syntactic dependency relations over dependency parse trees. Finally, compare with the GCN-based model

GCN-ED, JMEE and BGCN, our proposed method still achieves 2.5%, 1.9% and 1.4% improvement on F1-score. It indicates the superiority of exploiting multi-hop neighbors' information.

Table 1: Comparison to the-state-of-art methods

Model	P	R	F1
MaxEnt	74.5	59.1	65.9
Cross-Event	68.7	68.9	68.8
Cross-Entity	72.9	64.3	68.3
DMCNN	75.6	63.6	69.1
DMCNN+	75.7	66.0	70.5
JRNN	66.0	73.0	69.3
dbRNN	74.1	69.8	71.9
GCN-ED	77.9	68.8	73.1
JMEE	76.3	71.3	73.7
BGCN	75.9	72.5	74.2
Ours	**78.2**	**73.1**	**75.6**

4.4. Effect of network frameworks

In this section, we analysis different model frameworks to prove the effectiveness of GAT and GAT with Bi-GRU. To be more specific, we compare the full model with its incomplete model where the Bi-GRU layers are eliminated. For the version with the GAT layers, we gradually increase the hop of neighbors by executing graph attention convolution over different order dependency graphs until the performance drops. The performance of different models is shown in Table 2 on the ACE 2005 dataset, which contains two modules. The first module corresponds to the complete models; in the next module, the Bi-GRU layers are not included. From the table, we can know that both the full model and incomplete model achieve the best performance when the hop of neighbors M is 3. The full model we proposed achieves an F1-score of 75.6%, is better than the model without Bi-GRU with F1-score of 73.9%. Therefore, we can draw a conclusion that Bi-GRU can capture some useful feature that are not contained in GATs. So Bi-GRU can be regarded as a supplement to GAT on ED task. Combine the Bi-GRU and GAT could further improve the performance than only use GAT. We would use the best network framework in the next experiment.

Table 2: Performance of different frameworks

Model	P	R	F1
Bi-GRU + GAT (M=1)	72.1	70.9	71.5
Bi-GRU + GAT (M=2)	73.1	**73.4**	73.2
Bi-GRU + GAT (M=3)	78.2	73.1	**75.6**
Bi-GRU + GAT (M=4)	**80.3**	65.7	72.3
GAT only (M=1)	**80.0**	63.6	70.9
GAT only (M=2)	75.8	68.5	72.0
GAT only (M=3)	73.2	**74.6**	**73.9**
GAT only (M=4)	74.5	71.2	72.8

4.5. Effect of multi-label attention mechanism

To evaluate the effectiveness of the multi-label attention mechanism in this model, we replace it with mean pooling to fuse the multi-order word representations with multi-hop neighbors' information. The result is shown in Table 3. From the table we can see that the model with multi-label attention mechanism outperforms the mean pooling method. It achieves 1.9% improvement on F1-score. Thus, we can draw the conclusion that the multi-label attention mechanism can distinguish the importance of the word representations with different hop neighbors' information.

Table 3: Performance of different attention fusion methods

Method	P	R	F1
Mean Pooling	77.1	70.6	73.7
Multi-label attention	**78.2**	**73.1**	**75.6**

5. Conclusion

This paper proposes a novel neural network model for event detection. We utilize graph attention network to capture the word representations with multi-hop neighbors' information in dependency parse tree and adopt the multi-label attention mechanism to capture various intensive parts of the context. We widely compare our proposed model with a series of state-of-the-art models, experiments show that the proposed model achieves the best performance on the ACE 2005 dataset. For future research topics, we intend to apply our proposed model to the other related tasks, such as relation extraction, argument extraction, etc.

6. References

[1] Q. Li, H. Ji and L. Huang. Joint event extraction via structured prediction with global features. Proc. 51st Ann. Meet. of the Asso. for C. Ling, vol. 1,pp. 73-82, 2013.

[2] S. Liao, R. Grishman. Using document level cross-event inference to improve event extraction. Proc. 48th Ann. Meet. of the Asso. for C. Ling, pp. 789-797, 2010.

[3] Y. Hong, J. Zhang, B. Ma, J. Yao, G. Zhou and Q. Zhu. Using cross-entity inference to improve event extraction. Proc. 49th Ann. Meet. of the Asso. for C. Ling, pp. 1127-1136, 2011.

[4] J. Liu, L. Min and X. Huang. An overview of event extraction and its applications. arXiv preprint arXiv:2111.03212, 2021.

[5] T. H. Nguyen, R. Grishman. Event detection and domain adaptation with convolutional neural networks. Proc. 53rd Ann. Meet. of the Asso. for C. Ling. and the 7th Inter. J. Confer. on Nat. Lan. Pro, vol. 2, pp. 365-371, 2015.

[6] Y. Chen, L. Xu, K. Liu, D. Zeng and J. Zhao. Event extraction via dynamic multi-pooling convolutional neural networks. Proc. 53rd Ann. Meet. of the Asso. for C. Ling. and the 7th Inter. Joint Confer. on Nat. Lan. Pro, vol.1, pp. 167-176, 2015.

[7] T. H. Nguyen, K. Cho and R. Grishman. Joint event extraction via recurrent neural networks. Proc. 2016 Confer. of the Nor. American Chap. of the Asso. for C. Ling, pp. 300-309, 2016.

[8] X. Feng, B. Qin and T. Liu. A language-independent neural network for event detection. Sci. Ch. Inf. Sci, pp. 1-12, 2018.

[9] L. Sha, F. Qian, B. Chang and Z. Sui. Jointly extracting event triggers and arguments by dependency-bridge rnn and tensor-based argument interaction. AAAI Confer. on Art. Intel, 2018.

[10] T. H. Nguyen, R. Grishman. Graph convolutional networks with argument-aware pooling for event detection. AAAI Confer. on Art. Intel, 2018.

[11] X. Liu, Z. Luo and H. Huang. Jointly multiple events extraction via attention-based graph information aggregation. arXiv preprint arXiv:1809.09078, 2018.

[12] H. Yan, X. Jin, X. Meng, J. Guo and X. Cheng. Event detection with multi-order graph convolution and aggregated attention. Proc. 2019 Confer. on Emp. Meth. in Nat. Lan. Proc. and the 9th Inter. Joint Confer. on Nat. Lan. Pro, pp. 5766-5770, 2019.

[13] J. Zhou, G. Cui, S. Hu, Z. Zhang, C. Yang, Z. Liu, L. Wang, C. Li and M. Sun. Graph neural networks: A review of methods and applications. AI Open, pp. 57-81, 2020.

[14] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio and Y. Bengio. Graph attention networks. arXiv preprint arXiv:1710.10903, 2017.

[15] R. You, Z. Zhang, Z. Wang and S. Dai. Attentionxml: Label tree-based attention-aware deep model for high-performance extreme multi-label text classification. Adv. in Neu. Inf. Pro. Sys, 2019.

[16] Q. Li, J. Li, J. Sheng, S. Cui, J. Wu and Y. Hei. A Compact Survey on Event Extraction: Approaches and Applications. arXiv preprint arXiv:2107.02126, 2021.

[17] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado and J. Dean. Distributed representations of words and phrases and their compositionality. Ad. in Neu. Inf. Pro. Sys, pp. 3111-3119,2013.

[18] J. Chung, C. Gulcehre, K. H. Cho and Y. Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. arXiv preprint arXiv:1412.3555, 2014.

[19] S. Cui, B. Yu, T. Liu, Z. Zhang, X. Wang and J. Shi. Edge-enhanced graph convolution networks for event detection with syntactic relation. arXiv preprint arXiv:2002.10757, 2020.

[20] D. Marcheggiani, I. Titov. Encoding sentences with graph convolutional networks for semantic role labeling. arXiv preprint arXiv:1703.04826, 2017.

[21] D. A. Clevert, T. Unterthiner and S. Hochreiter. Fast and accurate deep network learning by exponential linear units (elus). arXiv preprint arXiv:1511.07289, 2015.

[22] Y. Chen, H. Yang, K. Liu, J. Zhao and Y. Jia. Collective event detection via a hierarchical and bias tagging networks with gated multi-level attention mechanisms. Proc. 2018 Confer. on Emp. Meth. in Nat. Lan. Pro, pp. 1267-1276, 2018.

[23] M. D. Zeiler. Adadelta: an adaptive learning rate method. arXiv preprint arXiv:1212.5701, 2012.

[24] Y. Chen, S. Liu, X. Zhang, K. Liu and J. Zhao. Automatically labeled data generation for large scale event extraction. Proc. 55th Ann. Meet. of the Asso. for C. Ling, Vol.1, pp. 409-419, 2017.

[25] S. Cheng, W. Ge, Y. Wang and J. Xu. BGCN: Trigger word detection based on BERT and graph convolutional networks. C. Sci, pp. 292-298, 2021.