Multi-Graph Convolution Network with Jump Connection for Event Detection

1st Xiangbin Meng
School of Computer Science, Beijing University of Posts and Telecommunications
Beijing, China
mengxb@bupt.edu.cn

2nd Pengfei Wang
School of Computer Science, Beijing University of Posts and Telecommunications
Beijing, China
wangpengfei@bupt.edu.cn

3rd Haoran Yan
Key Laboratory of Network Data Science and Technology, Institute of Computing Technology, Chinese Academy of Sciences
Beijing, China
yanhaoran17s@ict.ac.cn

4th Liutong Xu
School of Computer Science, Beijing University of Posts and Telecommunications
Beijing, China
xliutong@bupt.edu.cn

5th Jiafeng Guo
Key Laboratory of Network Data Science and Technology, Institute of Computing Technology, Chinese Academy of Sciences
Beijing, China
guojiafeng@ict.ac.cn

6th Yixing Fan
Key Laboratory of Network Data Science and Technology, Institute of Computing Technology, Chinese Academy of Sciences
Beijing, China
fanyixing@ict.ac.cn

***Abstract*—Event detection is an important information extraction task in nature language processing. Recently, the method based on syntactic information and graph convolution network has been wildly used in event detection task and achieved good performance. For event detection, graph convolution network (GCN) based on dependency arcs can capture the sentence syntactic representations and the syntactic information, which is from candidate triggers to arguments. However, existing methods based on GCN with dependency arcs suffer from imbalance and redundant information in graph. To capture important and refined information in graph, we propose Multi-graph Convolution Network with Jump Connection (MGJ-ED). The multi-graph convolution network module adds a core subgraph splitted from dependency graph which selects important one-hop neighbors' syntactic information in breadth via GCN. Also the jump connection architecture aggregate GCN layers' representation with different attention score, which learns the importance of neighbors' syntactic information of different hops away in depth. The experimental results on the widely used ACE 2005 dataset shows the superiority of the other state-of-the-art methods.**

***Index Terms*—event detection, multi-graph convolution network, jump connection aggregation, bias loss function, syntactic information in breadth and depth**

I. Introduction

Event extraction program defined by ACE is to identify events from given texts. An event is represented as a structure comprising an event trigger with specific types and a set of arguments with different roles. Generally, the event extraction (EE) task includes two sub-tasks, event detection (ED) and argument extraction (AE). Event detection, which aims to detect the event trigger, is an crucial part of event extraction as the main words to the corresponding events. For example, in the sentence "The company fired Anwar who was an engineer in 1998 .", an ED system is excepted to detect an $End{-}Position$ event with the trigger word $fired$". Argument extraction aims to identify all of the participants of each event, which are entities involved in events. In the above sentence, the arguments of the event include $comany$ (Role = Entity), $Anwar$ (Role = Person), $engineer$ (Role = Position) and 1998 (Role = Time). In this paper, we only focus on the event detection (ED) task.

Previous works employ sentence level sequential model methods (Chen et al., 2015) [1]; nguyen, 2016 [2]). These sequential methods only capture contextual information which ignores syntactic information and suffers from the low efficiency in capturing long range information. On the contrary, the methods based on dependency tree can capture syntactic information and the syntactic relation between triggers and arguments. For event detection, the syntactic information in sentence can help to identify the candidate trigger. Also, the syntactic relation between the candidate triggers and related arguments can effectively classify the event type of candidate triggers. For example, in the Fig. 1 sentence, the dependency path '$company-fired-Anwar$' to figure out the verb word $fired$ as the trigger word. The trigger word $fired$ can be identified to an $End-Position$ (a person fired from an organization) type or an $Attack$ (a person attack someone) type. With the auxiliary of the information between trigger word

This research work was supported by the National Natural Science Foundation of China under Grant No.61802029, the fundamental Research for the Central Universities No.500418800, the Youth Innovation Promotion Association CAS under Grants No. 20144310, the National Key R&D Program of China under Grants No. 2016QY02D0405, and the Foundation and Frontier Research Key Program of Chongqing Science and Technology Commission (No. cstc2017jcyjBX0059).

2375-0197/19/$31.00 ©2019 IEEE
DOI 10.1109/ICTAI.2019.00108

$fired$ and its related argument $engineer$, the trigger word $fired$ has more possibilities to be judged as $End-Position$ type instead of $Attack$ type. The statistics of dependency arcs hops from triggers to its related arguments as the Fig. 2 shows, 99.6% (9757/9793) triggers has the dependency relation with its related arguments in the ACE 2005 dataset, which shows that dependency tree arcs from triggers and related arguments are significant for event detection.

To capture syntactic information in the dependency tree of given sentence, we employ graph convolution network architecture (Kipf and Welling, 2016 [4]). There are several methods based on graph convolution network in dependency tree graph for event detection. Nguyen 's method [3] use dependency tree as a graph and adopt graph convolution network with novel pooling; (Liu et al., 2018 [5]) use graph convolution based dependency tree with self attention for event detection.

However, for event detection, the candidate trigger in graph needs to select important neighbors, which mainly explore the syntactic information and relation from the candidate trigger to its related arguments, which we say it needs different neighbors in breadth. Also, the candidate trigger needs dependency arcs' information with different hops away in graph to extract and filter useful and redundant information, which we say it need different syntactic information in depth. These GCN methods adopt syntactic information via the same neighbors information in the breadth of graph and the same dependency arcs' information with the same hops away in depth of the graph, which includes more redundant information. The redundant information will damage the generate of useful information and influence the identification of an event. For instance, for the example sentence shows in Fig. 1 and the given dependency tree graph of sentence in Fig. 3, the trigger word $fired$ can capture the syntactic information of its neighbor $Anwar$ as an related argument by graph convolution network. However, $fired$ can also capture the redundant neighbor punctuation '.' syntactic information which shows that the triggers need different neighbors in the breath of graph. Also $fired$ can capture the syntactic information with the argument $engineer$ at least 2 hops in the dependency tree graph, but capture the syntactic information with arguments $Anwar$ or $company$ only need one hop, which shows that triggers in graph need neighbors with different hops away in the depth of graph. Therefore, it is necessary to explicitly employ syntactic information in the breadth and depth of dependency tree graph for event detection.

In this paper, we propose a Multi-Graph Convolution Network with Jump Connection for Event Detection which called **MGJ-ED**. We use multi-graph convolution network based on dependency tree graph to select important one-hop neighbors' syntactic information in the breadth of graph. Also we use jump connection aggregation method (Xu et al., 2018) [6] to adaptively learn the neighbors' importance of syntactic information with different hops away in the depth of graph.

We evaluate the performance of our proposed MGJ-ED method on the wildly used ACE 2005 dataset to demonstrate the superiority of our method. Our main contributions can be summarized as follows:

- We propose a method to select neighbors in the breadth of graph via multi-graph convolution network based on dependency tree graph.
- We use the architecture of jump connection to aggregate graph representations, which can adaptively learn the importance of candidate triggers' neighbors with different hops away in depth of the graph.
- The modules we proposed use graph information in breadth and depth, which is the first time to use different direction information in graph for event detection.
- We achieve the state-of-the-art performance on the widely used ACE 2005 dataset for event detection.

II. Task Description

Event detection is a subtask of event extraction defined by ACE. An event is defined as an occurrence with one or more participants. We firstly introduce some ACE terminologies for understanding the task:

- **Entity mention**: reference of an entity (typically a noun phrase).
- **Event mention**: a phrase or sentence within which an event is described, including event triggers and related arguments.
- **Event trigger**: the main word which can most clearly express the event occurrence (typically a noun or a verb)
- **Event argument**: the entity mentions that are involved in an event.
- **argument role**: the relationship between an event and its related argument in which it participates.

The event detection task aims to identify event triggers and categorize their event types. For instance, in the sentence "The company fired Anwar who was an engineer in 1998 .", the event detection task excepts us to identify trigger words "$fired$" and classifies the event type as $Attack$. The extraction of event arguments "$company$"(Role = Entity), "$Anwar$"(Role = Person), "$engineer$(Role = Position)" and "1998"(Role = Time-Within) is not involved in the ED task. The ACE 2005 dataset annotate 8 super event types with 33 subtypes. Following previous work, we use these 33 subtypes as our event type labels. Also, we directly use the whole entity extent in ACE 2005 dataset.

III. The proposed Method

In general, event detection can be cast as a multi-class classification problem to decide whether a word in the sentence forms event triggers. Let $W = w_1, w_2, ., w_n$ be a sentence where n means the length of sentence and w_i means the i-th token. We apply BIO annotation schema to assign trigger labels since triggers may be multiple tokens.

Our MGJ-ED framework mainly consists of four modules: (i) encoder module that encode each token to vectors, (ii) multi-graph convolution network module that performs graph convolution network with syntactic graph and core subgraph, (iii) jump connection module, which aggregates

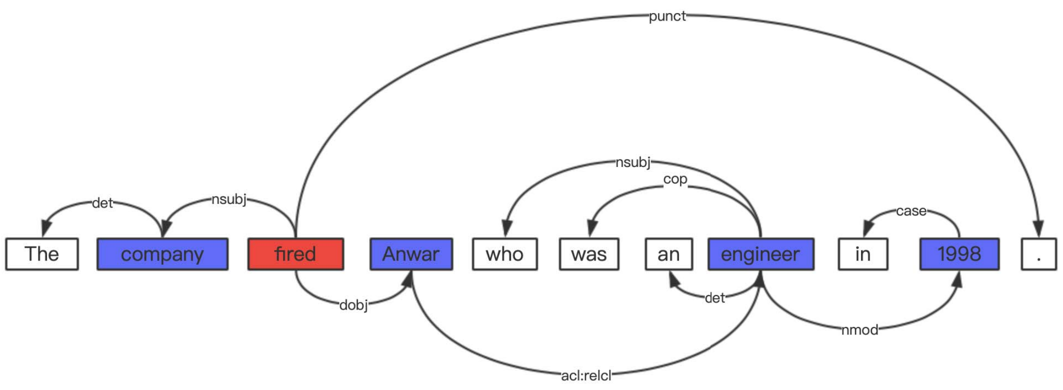

Fig. 1. The dependency parsing result of an example sentence, where red words represent event trigger and blue words represent arguments

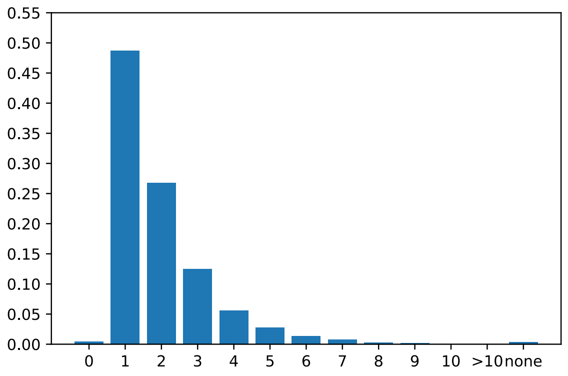

Fig. 2. The statistics of shortest path distance based on dependency tree in ACE 2005 dataset

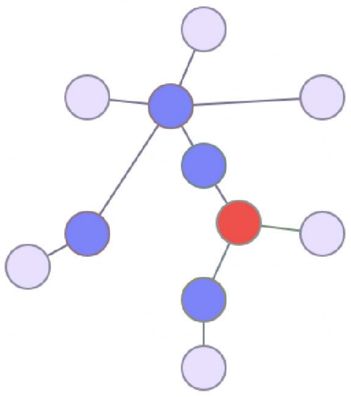

Fig. 3. Dependency tree graph of the example sentence, where the red node represents the trigger word '$fired$' and blue nodes represent related arguments.

graph representation by jump connection architecture, (iv) trigger classification module that predicts the event types of each candidate trigger.

A. Encoder

In the Encoder Module, each token w_i in sentence is transformed to a vector x_i by looking up in embedding tables and concatenating following embedding vectors:

- Word embedding vector $word_i$: $word_i$ is implemented by looking up a pretrained word embedding matrices.
- POS-tagging embedding vector pos_i: pos_i is generated by looking up a random initialized POS-tagging label embedding matrices.
- The entity type embedding vector e_i: Similarly to the POS-tagging embedding, we encode entity label by a random initialized POS-tagging label embedding matrices. We use the whole entity extent in ACE 2005 dataset as JMEE framework [5] did.
- Positional embedding vector pt_i: we encode the relative distance of the current candidate trigger word by a random initialized embedding matrices (Nguyen and Grishman, 2016 [2]; Liu et al., 2017 [8]).

The input sentence W can be represented to a sequence of vector $X = x1, x2, , xn$. Since GCN can only capture local graph limited by the number of the GCNs' layer, we use LSTM to expand the sentence information with the limited number of GCNs' layer. Following JMEE framework [5], we use bidirectional long-short term memory network(BiLSTM) [9] to encode the word representation X for expanding the sentence information:

$$\overrightarrow{p_i} = \overrightarrow{LSTM}(\overrightarrow{p_{i-1}}, x_i) \tag{1}$$

$$\overleftarrow{p_i} = \overleftarrow{LSTM}(\overleftarrow{p_{i-1}}, x_i), \tag{2}$$

The i-th token representation $\bar{x_i} = [\overrightarrow{p_i}, \overleftarrow{p_i}]$ will be the input of mutli-graph convolution network module as the first layer, where [ ] is the concatenation operation. We set $\bar{X} = \bar{x_1}, \bar{x_2}, ..., \bar{x_n}$to the encoder output representation.

B. Mutli-graph convolution network module

Each dependency tree can be represented to a graph with n nodes by its adjacency matrix $A = n * n$. Let $G = \{V, E\}$ be the dependency tree graph of sentence W, Where $V(|V| = n)$ as the sets of n nodes and E as the edges of G respectively. In V, each v_i is the encoder module output representation of token w_i in W. Each $edge(v_i, v_j) \in E$ is a directed dependency arc from token w_i to w_j, where $A[i, j] = 1$. As for an GCN in l-th layer, we denote h_i^{l-1} as the input of GCN layer and h_i^l as the output of GCN layer, where h_i^0 is the encode module representation of i-th token in W. The graph convolution equation as follows:

$$h_i^l = f(\sum_{j=1}^{n} A_{ij} W_g^l h_j^{l-1} + b_g^l) \tag{3}$$

Where W_g^l is the weight matrix in linear transformation, b_g^l is the bias term in l-th layer. f is the non-linear function.

To flow information in opposite direction, we add reversed edge $E(v_j, v_i)$ of directed edge $E(v_i, v_j)$ in the graph G, where $A[j, i] = 1$. For capturing the information of the node itself, we add self-loops in graph, where $A[i, i] = 1$. Therefore, we denote graph $\bar{G}$ with reversed edges and self-loops. For example, in the dependency arcs shown in Fig. 1, there are four dependency arcs with two nodes "$fired$" and "$Anwar$": the directed arc ("$fired$", "$Anwar$") , the reversed arc ("$Anwar$", "$fired$") and two self loop arc ("$fired$", "$fired$"), ("$Anwar$", "$Anwar$"). Also, we use degree normalization to solve the high degree bias problem. Therefore, the equation can be written as follows:

$$h_i^l = f(\sum_{j=1}^{n} \bar{A_{ij}} W_g^l h_j^{l-1} / d_i + b_g^l) \tag{4}$$

where $\bar{A}$ is the adjacency matrix with reversed edges and self-loops in dependency graph $\bar{G}$, and $d_i = \sum_{j=1}^{n} \bar{A_{ij}}$ is the degree of token i in the resulting graph.

To select the neighbors which can explore syntactic information from the candidate trigger and its related argument, we split the dependency graph $\bar{G}$ to a core subgraph $\bar{G_c}$ as the Fig. 4 shows. The core subgraph $\bar{G_c}$ only remains the dependency path from candidate triggers to entities with its self-loops, where the arguments are entity mentions in sentence and not all event includes arguments. The core subgraph's representation in l-th layer of GCN can be written as follows:

$$h_{ci}^l = f(\sum_{j=1}^{n} \bar{A_{cij}} W_{cg}^l h_{cj}^{l-1} / d_{ci} + b_{cg}^l) \tag{5}$$

Where h_c^l is the output of l-th GCN layer, $\bar{A_c}$ is the adjacency matrix and $d_{ci} = \sum_{j=1}^{n} \bar{A_{cij}}$ in the core subgraph $\bar{G_c}$. Also W_{cg}^l is the weight matrix, b_{cg}^l is the bias item and f is the non-linear function.

As Fig. 5 shows, we use the encoder modules' representation $\bar{X}$ as the input of graph convolution network based on the dependency graph $\bar{G}$ and coregraph $\bar{G_c}$. Also we stack L over layers of GCN, which can generate the graph representation in L hops. For dependency graph $\bar{G}$, it can represent the whole dependency relation in the sentence. The GCN architecture based on graph $\bar{G}$ can capture the syntactic information in sentence. As for the core subgraph $\bar{G_c}$, the weight of dependency arcs along with the dependency path from candidate trigger to entities including arguments are 1 and the others are 0 compared with the original graph. The GCN architecture based on the core subgraph $\bar{G_c}$ can strongly select the neighbors in the dependency path from candidate triggers to arguments, which captures important syntactic information in the breadth of graph.

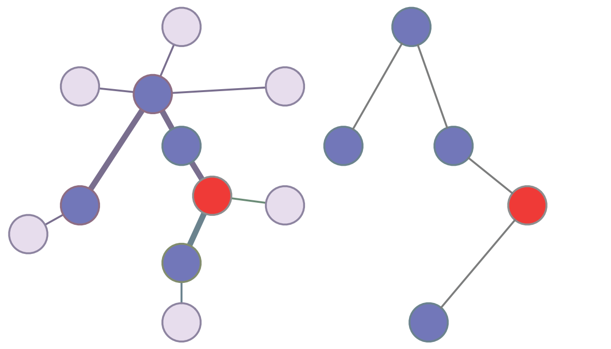
Fig. 4. The dependency graph and the core subgraph splited based on graph of example sentence, where token $'fired'$ as the candidate trigger. We ignore the self-loops for display.

C. Jump connection module

Since each token in dependency tree graph need different hops of neighbor arcs' information in depth for event detection, we employ a jump connection aggregation mechanism to extract and filter useful syntactic information in the depth of graph by learning the importance of different GCN layers' representation and aggregate the GCN layers' representation. Since jump connection aggregation mechanisms in different graph are same, we ignore different graphs in jump connection mechanism for convenience. The vector of aggregation with GCN layers' representation in two graph can be written as follows:

$$\bar{h_i} = \sum_{l=1}^{L} \alpha_i^l h_i^l, \tag{6}$$

Where $\bar{h_i}$ is the aggregation' vector with GCN layers' representation, L is the number of GCN layers, α_i is the attention score of the token i for each layer l in both graph, $\sum_{l=1}^{L} \alpha_i^l = 1$. The attention score represents the importance of the graph representation learned on l-th layer for token i. We use LSTM attention method to calculate the attention score. We input h_i^1 , ..., h_i^L into a bi-directional LSTM and generate the forward and backward representations f_i^l and b_i^l for each layer l.The attention score α_i^l will be generated by a linear mapping of the concatenated representations $[f_i^l, b_i^l]$:

$$\alpha_i^l = softmax(s_i^l) = \frac{exp({s_i^l}^T W_a)}{\sum_{j=1}^{L} exp({s_{ij}^j}^T W_a)} \tag{7}$$

$$s_i^l = f(W_b[f_i^l, b_i^l] + b_{item}) \tag{8}$$

Where W_a and W_b are weight matrix and b_{item} is the bias term in both graphs, f is the non-linear activation function.

To aggregate the graph representations from two graph, we use max pooling to generate representation $g_i = max(\bar{h_i}, \bar{h_{ci}})$ for each token i, where $\bar{h_i}$ is the vector of aggregation with GCN layers' representation based on graph $\bar{G}$ and $\bar{h_{ci}}$ is the

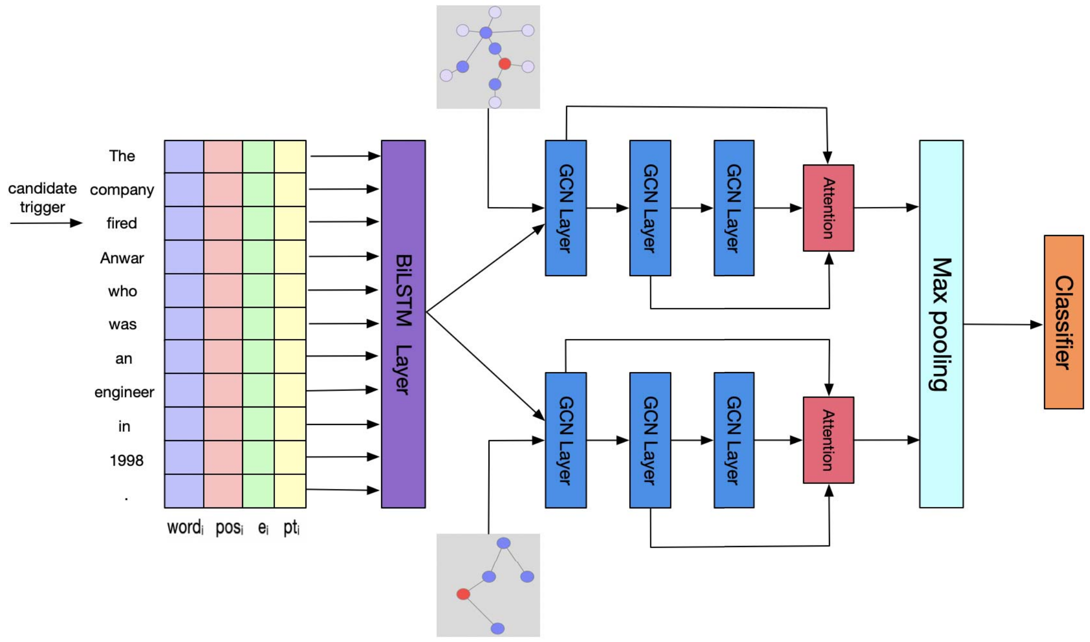

Fig. 5. The architecture of the MGJ-ED method, the number of GCN layer is set to 3.

vector of with GCN layers' representation based on the core subgraph $\bar{G}_c$.

D. Trigger Classification

In the trigger classification module, we taking the representation g_i of each token i from the multi-graph convolution network with jump connection. We feed the token representation g_i in to a fully connected network to predict the trigger label:

$$y_i^t = softmax(O_i^t) = \frac{exp(O_i^t)}{\sum_{j=1}^{2N+1} exp(O_i^j)}, \tag{9}$$

$$O_i = W_o g_i + b_o \tag{10}$$

Where W_o is the weight matrix and b_o is the bias term; N is the number of trigger labels defined by ACE 2005. As for the BIO annotation, the number of trigger labels which we will predict is $2N+1$, including $None$ class. y_i^t is the output of the t-th trigger label for i-th token.

E. Biased Loss Function

Since the number of $None$ label is much more than other trigger labels, we adapt negative log-likelihood biased loss function :

$$J(\theta) = -\sum_{i=1}^{N_s}\sum_{j=1}^{N_t}(\log p(y_j^t|s_i,\theta)\cdot I(y_i^t)) \tag{11}$$

Where N_s is the number of the sentence in dataset, N_t is the number of tokens in sentence s_i. $I(y_i^t)$ is an indicating function, if y_i^t is $None$ class, it will set the bias number λ be one, otherwise larger than one. λ is also a hyper parameter.

IV. Experiments

A. Dataset, Resources and Evaluation Metric

We evaluate our MGJ-ED method by using widely used dataset ACE 2005 for event detection. The ACE 2005 dataset annotate 33 event subtypes, along with the BIO annotation shcema and $None$ Class, we will classify each token into 67 categories. To comply with previous work [2] [5], we use the same data split includes 40 newswire documents for the test set, 30 other documents for the dev set and 529 remaining documents for the training set.

In the data preprocessing, we deploy the Stanford CoreNLP toolkit for tokenizing, sentence splitting, pos-tagging and dependency parsing. We use the pretrained word embedding from NYT corpus with Skip-gram algorithm(Mikoloev et al., 2013 [10]; Chen et al., 2018 [11]) .

Also, following the previous work [5] [7] [11], we use Precision (P) , Recall (R) and F measure (F_1) to report the performance of the model.

B. Hyper parameter Setting

The hyper parameters are tuned by the performance on the dev dataset of the ACE 2005 dataset. In encoder module, we set 100 dimensions for word embedding, 50 dimensions for entity type embedding, pos-tagging embedding and positional embedding, 300 dimensions for the hidden units of BiLSTM. In multi-graph convolution network module, we set the hidden units for graph convolution to 300 dimensions and the number of graph convolution layer L to 3. In jump connection module, we set bi-lstm hideen units dim to 300. For loss function, we set the loss function bias weight λ to 5.

Also, we set the batch size to 32 and we utilize a fixed length n = 50. This implies that need to cut off the longer

TABLE I
PERFORMANCE OF DIFFERENT MODULES BASED ON MGJ-ED .

Method	P	R	F_1
GCN-naive	**76.6**	67.3	71.6
GCN-multi	74.6	72.1	73.3
GCN-jump	72.8	73.6	73.2
MGJ-ED	73.9	**78.3**	**76.0**

sentence and pad the shorter ones. We use the RELU (Glorot et al., 2011 [12]) as our nonlinear activate function. We set learning rate to 0.001, dropout rate to 0.5 and use the Adam update rule(Kingma et al., 2014 [13]) during training. We also fine-tuned all the embedding table during training.

C. Effect of our modules

To evaluate the impact of capturing syntactic information in breadth and depth with multi-graph convolution network module and jump connection module, we design models with different architectures based on dependency arcs with graph convolution network: 1) **GCN-naive**: it uses graph convolution network based dependency graph; 2): **GCN-multi**: it uses multi-graph convolution network module based on GCN; 3): **GCN-jump**: it adopts jump connection architectures based on GCN; 4): **MGJ-ED**: which combines both multi-graph convolution module and jump connection module.

Table I shows the experimental results that the GCN-multi improves 1.7% F_1 scores and GCN-jump improves 1.6% F_1 score compared with GCN-naive. The module of multi-graph convolution network adds a core subgraph which can selects the neighbors and lead the candidate trigger word to explore the syntactic information with argument. The jump connection aggregation module learns the importance of dependency arcs information with different range in depth. Also combining the two modules as MGJ-ED shows, multi-graph module and jump connection module can effectively improve performance each other for event detection, which improves 2.7% and 2.8% F_1 measure score compared with GCN-multi and GCN-jump. The methods of multi-graph convolution network and jump connection get better performance than the method based on GCN, which proves that these method can effectively identify event triggers in event detection task.

D. overall performance

We compare the performance one the test set with the following state-of-the-art methods :

1). **Cross Event**: which uses the document information to improve the performance for event extraction in sentence level. (Liao and Grishman., 2011) [14].

2). **CNN**: a CNN model for event detection.(Nguyen and Grishman, 2015) [15].

3). **DMCNN**: which uses dynamic multi-pooling method in convolution neural network for capturing multiple events' information (Chen et al., 2015) [16].

4). **DMCNN+**: the dynamic multi-pooling model with automatic labeled data (Chen et al., 2017) [17].

5). **JRNN**: a joint model with a bidirectional RNN and manually designed features (Nguyen et al., 2016) [18].

TABLE II
OVERALL PERFORMANCE OF DIFFERENT THE-STATE-OF-ART METHOD ON TEST SET

Method	P	R	F_1
Cross Event	68.7	68.9	68.8
CNN	71.8	66.4	69.0
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
ANN-S2	**78.0**	66.3	71.7
GCN-ED	77.9	68.8	73.1
JMEE	76.3	71.3	73.7
DEEB-RNN	72.3	75.8	74.0
GCN-naive	76.6	67.3	71.6
GCN-multi	74.6	72.1	73.3
GCN-jump	72.8	73.6	73.2
MGJ-ED	73.9	**78.3**	**76.0**

6). **dbRNN**: which adds dependency arcs' information over Bi-LSTM to improve performance for event extraction (sha et al., 2018) [7].

7). **ATT-S2**: which uses supervised mechanism to explicitly exploits argument information for event detection (Liu et al., 2017) [8].

8). **GCN-ED**: which uses novel pooling method based on GCN for event detection (Nguyen et al., 2018) [3].

9). **JMEE**: which uses GCN with highway network and self attention aggregate mechanism for event detection (Liu et al., 2018) [5].

10). **DEEB-ED**: which used hierarchical superviesd attention based bidirectional RNN with document information for event detection (Zhao et al., 2018) [19].

Table II shows the performance comparing to the state-of-the art methods. From the table we can see that our MGJ-ED model achieve the best recall and F_1-measure score among all the compared methods. Our method improve significantly performance, which is higher over 2.5% on recall and 2.0% on F_1-measure score with the best baseline performance. These results show the effectiveness of capturing information in breadth and depth with multi-graph convolution network module and jump connection module.

E. Multi-graph convolution network module analysis

The result in Table I shows that multi-graph convolution network module can improve performance. For the multi-graph convolution module, we split the dependency tree graph $\bar{G}$ with reversed arcs and self-loops to a core subgraph $\bar{G_c}$. Subgraph $\bar{G_c}$ only remains the dependency path from candidate triggers to related entities. Since the nodes in original dependency graph $\bar{G}$ have the same weight of neighbors, the candidate triggers in graph can not select important neighbors, such as arguments.

The core subgraph we splited can better select the important nodes of the candidate trigger compared with some attention method, such as GAT [27] which is widely used in GCN models. In the core subgraph $\bar{G_c}$, the arcs' weight in dependency graph from candidate triggers to entities including arguments are set to 1, others are set to 0, which can strongly to select neighbors and lead the candidate triggers to capture the information in the path with arguments. However, if we only employ the core subgraph via GCN for event detection, it will lost some syntactic information in sentence. Therefore, the multi-graph convolution network module employ GCN based on both two graphs, where the core graph via GCN lead the nodes to choose their neighbors, which we say it capture syntactic information for event detection in breadth. The experiments in Table III shows that the multi-graph convolution network module can effectively improve 1.8% F_1 measure score and 0.8% score compared with the GCN-naive model and the GCN-core model which only employs the core subgraph via GCN.

TABLE III
PERFORMANCE OF GCN METHOD BASED ON DIFFERENT GRAPHS

Method	P	R	F_1
GCN-naive	**76.6**	67.3	71.6
GCN-core	70.0	**75.2**	72.5
GCN-multi	76.3	71.3	**73.3**

F. Jump connection module analysis

For the jump connection module, we weight the importance of different GCN layers' representation by calculating lstm attention score. Traditional graph convolution network in graph $\bar{G}$ uses the last layer graph representation as the input of classifier, which involves neighbors' information in L hops (L is the number of GCN layer). However, traditional graph convolution network will cause over-smoothing problem (zhou et al., 2018graph) [20]. Also the nodes need dependency arcs' information with different hops away in depth of graph for event detection. To solve the problem, the jump connection module aggregates all GCN layers' representation and learns the importance of nodes with different hops away by calculating the graph convolution layer attention score, which we say it improves performance in depth for event detection.

We use the example sentence ”The company fired Anwar who was an engineer in 1998 .” as an example to illustrate the advantage of our jump connection module. There is an event in the sentence: an $End-position$ triggered by $fired$ with arguments $Company$, $Anwar$, $Engineer$ and 1998, where Fig. 3 shows the dependency tree graph. We expect the candidate trigger $fired$ could more focus on capturing syntactic information from arguments ($Company$, $Anwar$) in one hop and $engineer$ in two hops, while the argument 1998 is a supplement of the event with three hops. We gather the statistics of attention score from jump connection module. The Fig. 6 shows that the first two GCN layers get more attention score when token $fired$ as the candidate word, which proves that our model can capture more syntactic information within 2-hops local graph via jump connection module. Compared the traditional GCN method which uses the last layer output as the graph representation, the jump connection module capture neighbors' syntactic information with different hops away by different layers' importance. Also Table I illustrates that the jump connection module can improve 1.7% F_1 measure score compared with the GCN-naive model and the MGJ-ED model can improve 2.8% F_1 measure score compared with the GCN-multi. The attention scores and performance compared with other models proves the superiority of jump connection module.

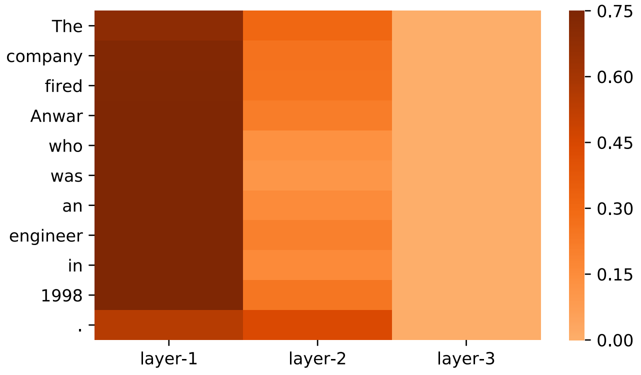

Fig. 6. Visualization of the attention score of the example sentence in different graph convolution layer based on original graph $\bar{G}$, where token $fired$ as the candidate trigger. Darker red mean higher score.

G. Parameter GCN-layer analysis

As the Fig. 2 shows, we count that about 88.4% (8863/9793) dependency path from triggers and arguments in 3 hops based on ACE 2005 benchmark dataset, which means that 3-layers graph convolution network can capture most syntactic information of the sentence. Since the overfull layers may lead the serious over-smooth problem and the Bi-LSTM encoder architecture can expand the representation in sentence, we believe $L = 3$ can have a better performance for event detection. From Table IV we can see the MGJ-ED with 3 layers have the best performance.

TABLE IV
PERFORMANCE OF MGJ-ED WITH DIFFERENT LAYERS

Method	P	R	F_1
MGJ-ED ($L = 2$)	70.2	**78.5**	74.3
MGJ-ED ($L = 3$)	**73.9**	78.3	**76.0**
MGJ-ED ($L = 4$)	71.4	78.0	74.6

V. RELATED WORK

Event detection is an important task in nature language processing. There are several existing approaches in the event detection task. The early methods use different statistical models with hand-design feature for event detection task achieve good performance for event detection (Ahn 2006; Ji and Grishman 2008; Hong et al. 2011) [21] [22] [23]. These feature-based methods depend on the extensive human engineering which will influence the model performance.

Recently, the neural network models are widely utilized in event detection task. (Nguyen et al., 2015;Nguyen and Grishman 2016) [15] [2] employ CNN models; (Chen et al. 2015) [1] uses dynamic multiple pooling CNN model for event detection. Recurrent neural networks (Nguyen, Cho, and

Grishman 2016) [18] are also employed for event detection. Also (Liu et al.2017) [8] uses supervised argument attention method to identify event triggers. (Liao and Grishman, 2010; Ji and Grishman, 2008; Hong et al., 2011; Reichart and Barzilay, 2012; Lu and Roth, 2012; Zhao et al., 2018) [14] [23] [20] [25] [26] use document-level information to improve the classification of trigger words. However, these sequential modeling methods suffer from the less information between candidate triggers and its related arguments. Also, it can not capture the dependency syntactic information.

There are also methods based on syntactic information for event detection. (Sha et al., 2018) [7] uses dependency arcs over Bi-LSTM for event detection. (Nguyen and Grishman, 2018; Liu et al. 2018) [3] [5] employed graph convolution network based on dependency tree graph for event detection and achieve good performance. However, these dependency tree and GCN methods can not explicitly capture the syntactic information from the dependency tree. Different from them, our method explores refined information in breadth and depth of the dependency graph to improve the performance of event detection. This is the first work to employ graph convolution based on dependency tree graph in different direction for event detection.

VI. Conclusion

The paper proposed a Multi-Graph Convolution Network with Jump Connection framework (**MGJ-ED**) for event detection task. In our framework we introduce multi-graph convolution network module and jump connection module to captures syntactic information in breadth and depth of the dependency tree graph. The modules we proposed is the first time to use syntactic information in different direction of the dependency tree graph for event detection. The experiment results compared with other the-state-of-the-art method baseline prove the superiority of our proposed method.

References

[1] Chen, Yubo and Xu, Liheng and Liu, Kang and Zeng, Daojian and Zhao Jun. Event extraction via dynamic multi-pooling convolutional neural networks. Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing. volume 1, pages 167-176, 2015.

[2] Thien Huu Nguyen and Ralph Grishman. Modeling skip-grams for event detection with convolutional neural networks. In Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 886891. 2016.

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

[4] Kipf, Thomas N and Welling, Max. Semi-supervised classification with graph convolutional networks. arXiv preprint arXiv:1609.02907. 2016.

[5] Liu, Xiao and Luo, Zhunchen and Huang, Heyan. Jointly multiple events extraction via attention-based graph information aggregation. arXiv preprint arXiv:1809.09078. 2018.

[6] Keyulu Xu, Chengtao Li, Yonglong Tian, Tomohiro Sonobe, Ken-ichi Kawarabayashi, and Stefanie. Representation learning on graphs with jumping knowledge networks. In International Conference on Machine Learning (ICML), pp. 54535462, 2018.

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

[8] Shulin Liu, Yubo Chen, Kang Liu, and Jun Zhao. Exploiting argument information to improve event detection via supervised attention mechanisms. Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, pages 1789 1798. 2017.

[9] Sepp Hochreiter and Ju rgen Schmidhuber. Long short-term memory. Neural Computation, 9(8):17351780. 1997.

[10] Mikolov, Tomas and Sutskever, Ilya and Chen, Kai and Corrado, Greg S and Dean, Jeff. Distributed representations of words and phrases and their compositionality. Advances in neural information processing systems, pages 3111-3119. 2013

[11] Chen, Yubo and Yang, Hang and Liu, Kang and Zhao, Jun and Jia, Yantao. Collective event detection via a hierarchical and bias tagging networks with gated multi-level attention mechanisms. Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 1267-1276. 2018.

[12] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. eep sparse rectifier neural networks. Proceedings of the 14th International Conference on Artificial Intelligence and Statistics, pages 315323. 2011.

[13] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. CoRR, abs/1412.6980. 2014

[14] Shasha Liao and Ralph Grishman. Using document level cross-event inference to improve event extraction. In Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, pages 789797. 2010.

[15] Nguyen, T. H., and Grishman, R. Event detection and domain adaptation with convolutional neural networks. In ACL-IJCNLP. 2015.

[16] Yubo Chen, Liheng Xu, Kang Liu, Daojian Zeng, and Jun Zhao. 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 of the Asian Federation of Natural Language Processing, pages 167176. 2015.

[17] Yubo Chen, Shulin Liu, Xiang Zhang, Kang Liu, and Jun Zhao.Automatically labeled data generation for large scale event extraction. In Proceedings of ACL . 2017.

[18] Thien Huu Nguyen, Kyunghyun Cho, and Ralph Grishman. 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 300309. 2016.

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

[20] Zhou, Jie and Cui, Ganqu and Zhang, Zhengyan and Yang, Cheng and Liu, Zhiyuan and Sun, Maosong. Graph neural networks: A review of methods and applications. arXiv preprint arXiv:1812.08434. 2018.

[21] Ahn, D. 2006. The stages of event extraction. In Proceedings of the Workshop on Annotating and Reasoning about Time and Events.

[22] Heng Ji and Ralph Grishman. Refining event extraction through cross-document inference. In Proceedings of ACL, pages 254262. 2008.

[23] Yu Hong, Jianfeng Zhang, Bin Ma, Jian-Min Yao, Guodong Zhou, and Qiaoming Zhu. Using cross-entity inference to improve event extraction.In Proceedings of ACL, pages 11271136. 2011.

[24] Heng Ji and Ralph Grishman. Refining event extraction through cross-document inference. In Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics, pages 254262. 2008.

[25] Roi Reichart and Regina Barzilay. Multi-event extraction guided by global constraints. In Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pages 70 79. 2012.

[26] Wei Lu and Dan Roth. Automatic event extraction with structured preference modeling. In Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics, pages 835844. 2012.

[27] Veličković, Petar and Cucurull, Guillem and Casanova, Arantxa and Romero, Adriana and Lio, Pietro and Bengio, Yoshua. Graph attention networks. arXiv preprint arXiv:1710.1090. 2017.