SemEAGAT: A Novel Approach by Incorporating Semantic Dependency Graph in Event Detection

Yuqi Xi[1], **Jiamin Lu**[1]*, **Tingting Hang**[1], **Yunfei Zhang**[1], **Zhongyi Wang**[1] **and Jun Feng**[1]

[1]Key Laboratory of Water Big Data Technology of Ministry of Water Resources School of Computer and Information, Hohai University

E-mail: 282281397@qq.com

Abstract. Event detection (ED) is a task that requires capturing deep semantic information in text to correctly identify specific types of events. Semantic dependency graph aims to recover sentence-internal predicate-argument relationships. However, recent ED researches often use syntactic dependency tree in GNNs, while we believe that the semantic dependency graph can further improve the ED task's effectiveness since it has already been used in many other NLP tasks such as relation extraction, machine translation and abstractive summarization to provide effective semantic information, which is exactly required for event detection. In this paper, we propose a novel semantic dependency edges aware graph attention network (SemEAGAT). It incorporates the semantic dependency graph with an additional multi-head attention in an edge-aware way. Experiments on ACE2005 show our proposed method can achieve better effectiveness by comparing with the state-of-the-art methods.

1. Introduction

Event Detection (ED) is a crucial subtask in information extraction of Natural Language Processing, which aims to detect event instances of specific types that occur in a given text. Each event instance usually appears in a single sentence in which an event trigger (most often is a single verb or nominalization) is selected to associate with that event instance. ED task, more precisely stated, is to identify event triggers and classify them into specific event subtypes of interest. In this paper, we focus on the event detection task defined in Automatic Content Extraction (ACE) evaluation1. Taking the following sentence in ACE2005 as an example: The plane arrived back to base safely. An ED system should be able to recognize the word "arrive" as a trigger and classify it to the pre-defined event subtype Movement:Transport.

In earlier ED researches, traditional feature-based methods such as [1] [2] focused on designing human feature engineering to perform ED, and achieved a relatively high performance at that time. However, the performance of these feature-based methods largely depends on the quality of elaborately designed features. In recent years, as the advantages of automatically capturing effective features, neural networks have been employed in ED by many researchers. The typical neural ED

* indicates corresponding author

[1] http://projects.ldc.upenn.edu/ace

methods such as DMCNN [3], JRNN [4] belong to the category of sequence modeling methods, which treat each sentence as a sequence and automatically learn effective feature representations from the sequence. Recently, ED researches intend to employing Graph Convolutional Network (GCN) [5] and its variants such as Graph Recurrent Network (GRN) [6], Graph Attention Network (GAT) [7] to perform the task. Among these graph neural networks, our method is based on GAT since its ability that weights the importance of neighbors during the information aggregation process. Compared to the above sequence modeling methods, these GCN-based ED researches incorporate the dependency structure information conveyed by syntactic dependency trees, which achieve better performance [8] [9] [10].

However, the deep semantic information in semantic depen- dency graph has not been utilized in event detection, which may provide accurate and effective information to help perform ED more precisely. Take another sentence as an example: *Two former Congressmen now stepping into the CROSSFIRE to debate this, Maine Democrat Tom Andrews, now national director of Win Without War*. In multiple common perspectives like the co-occurrence relationship, semantic similarity relationship, and shallow semantic relationship (i.e., syntactic relationship), the word "*War*" either has a strong bonding withthe word "*Win*", or is affected by the overemphasis on its literal meaning, hence it is often adopted by ED systems as a golden trigger for a *Conflict:Attack* event. Nevertheless, in this sentence, the truth is that there is only one actual trigger "*former*" which evokes an *End-Position* event, and the word "*War*" is a non-trigger instead.

Such misprediction can be avoided by utilizing the deep se- mantic information in semantic dependency graph. As shown in Fig. 1, the word "*War*" is strongly influenced by the word "*Without*" according to the semantic dependency graph within the sentence. Specifically, there is one (also only one) incoming edge labeled *ARG2* from "*Without*" to "*War*", which lets the deep semantic (i.e., semantic relationship *ARG2*) information flow from "*Without*" to "*War*". Hence ED systems have a high degree of confidence to overcome both the overemphasizing on the word "*War*"'s literal meaning in the syntactic dependency tree and the misleading from the two common perspectives, i.e., the co-occurrence relationship and semantic similarity relationship, and naturally predicting the word "*War*" to be a non-trigger so as to avoid such misprediction.

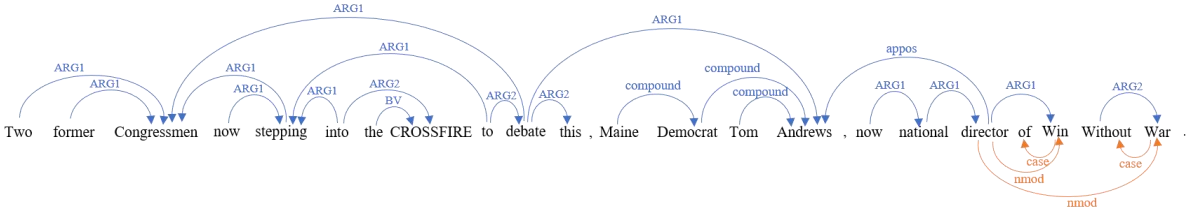

Figure 1. An example shows both the semantic dependency parsing result (top in blue) and the syntactic dependency parsing result (bottom in orange) of this sentence. In multiple common perspectives, ED systems tend to identify the word "*former*" as a trigger evoking an *End-Position* event, and adopt the word "*War*" as a trigger evoking a *Conflict:Attack* event. While there is sufficient evidence in the semantic dependency graph to prove the word "*War*" is actually a non-trigger. The semantic edges are all shown, and the syntactic edges not connected to "*Win*", "*Without*", or "*War*" are omitted for brevity.

In addition to directly mitigating the mispredictions, incorporating semantic dependency graph can also reduce the irrelevant information for ED, which may further improve the precision of event detection. This benefit can be attributed to the better partial connectivity of semantic dependency graph, indicating that semantic dependency graph allows function words, e.g., the word "*of* " in Fig. 1, to have no semantic contribution of their own. Besides, the high degree of non-projectivity [11] also means that semantic dependency graph has the ability to further strengthening the information flow within the sentence. Based on these advantages of semantic dependency graph, our intuition is that incorporating semantic dependency graph can provide effective deep semantic information to perform

event detection better. To our knowledge, we are the first to explore semantic dependency graph for event detection.

In this paper, we propose a novel semantic dependency edges aware graph attention network (SemEAGAT) to incorporate semantic dependency graph for event detection. In specific, for a constructed semantic dependency graph, we not only employ GAT to exploit the connectivity of semantic dependency edges between trigger candidates, but also apply an additional multi-head attention [7] [12] to integrate the semantic relationship information on these edges, which indicates EAGAT performs the information aggregation process in an edge-aware way to make better use of semantic dependency structure information. Experiments conducted on ACE2005 show that our proposed method outperforms other baseline methods, achieving a competitive performance.

2. Related work

Event detection (ED) has received extensive attention from researchers in the past decades, and these ED approaches can be broadly classified into feature-based approaches and neural network approaches. Feature-based approaches mostly focused on performing handcraft feature engineering. For example, Ahn [13] used lexical features, syntactic features, and external knowledge features for ED. Hong et al. [1] proposed the cross-entity inference method for ED, which regards entity-type consistency as the key feature. Li et al. [2] presented a joint framework that considers both local features and global features. Although these feature-based approaches achieved a relatively high performance at that time, the performance are sensitive to the quality of the manually designed features.

In the last several years, as the advantages of automatically capturing effective features, neural networks have gained rapid momentum in various fields, and more ED researchers turned their attention on conducting neural event detection. The typical models such as, Chen et al. [3] applied a novel DMCNN for keeping more contextual information benefit to ED. Nguyen et al. [4] first proposed a novel JRNN framework, and further proved the effectiveness of memory vectors (matrices) in ED. Moreover, attention mechanisms have also been introduced into ED thanks to its effectiveness in focusing on more important information. For example, Chen et al. [14] proposed the gated multi-level attention mechanisms to solve document-level ED. We also note some researches applying new learning strategies to ED, such as Hong et al. [15] adopted GAN to conduct self-regulation for improving ED, Nghia et al. [16] applied the hard attention with Gumbel-Softmax trick for ED.

The aforementioned typical neural networks in ED such as CNNs, RNNs are dedicated to handling sequential structures within the sentence but are hard to deal with the graph structures, so many graph-based ED researches emerged in more recent years, and they achieved a great success. Nguyen et al. [8] proposed GCN-ED to transform the syntactic dependency tree within sentences into a graph and employed GCN [9] to conduct ED. Liu et al. [9] proposed a novel JMEE framework to exploit the syntactic dependency tree by introducing GCN. Furthermore, Yan et al. [10] proposed MOGANED through employing GAT and further studied explicitly gathering information from both first-order and multi-order neighbors in the syntactic dependency tree. Lv et al. [17] proposed a novel Hierarchical Graph Enhanced Event Detection (HGEED) framework to make better use of both syntax and document information for the ED task, achieving a newly high performance. These graph-based ED researches achieved better performance than the typical neural methods in general, it is mainly due to the combination of GCNs and syntactic dependency structure. However, the semantic dependency structure within sentences has not been explored for ED, which may provide ED systems with deeper semantic information so that performing better ED.

We fill the gap in this work by adopting Graph Attention Network (i.e., GAT) and the multi-head attention mechanism to incorporate semantic dependency graph for ED.

3. The proposed method

Following previous researches [3] [4] [8], we formulate event detection as a multi-class classification problem. Our task is to identify trigger candidates in the given sentence and assign a label to each identified. The label can be one of the pre-defined event subtypes or NONE which indicates the

candidate is a non-trigger of any event. Thus, we have an equivalent problem of ($N_{et}+1$)-class classification for ED where N_{et} is the number of pre-defined event subtypes. There are 33 pre-defined event subtypes in ACE2005.

Fig. 2 shows the overall architecture of SemEAGAT. It is mainly composed of three modules: encoding module, graph information module, and classifier module. Firstly, the encoding module transforms and encodes the input sentence for obtaining the local contextual representation of each word. Then, based on the contextualized representations, the graph information module utilize EAGAT to capture deep semantic-aware word representations from the semantic dependency graph structure. Lastly, the deep semantic-aware word representations are used for classification. In the following sections, we introduce each module in detail respectively.

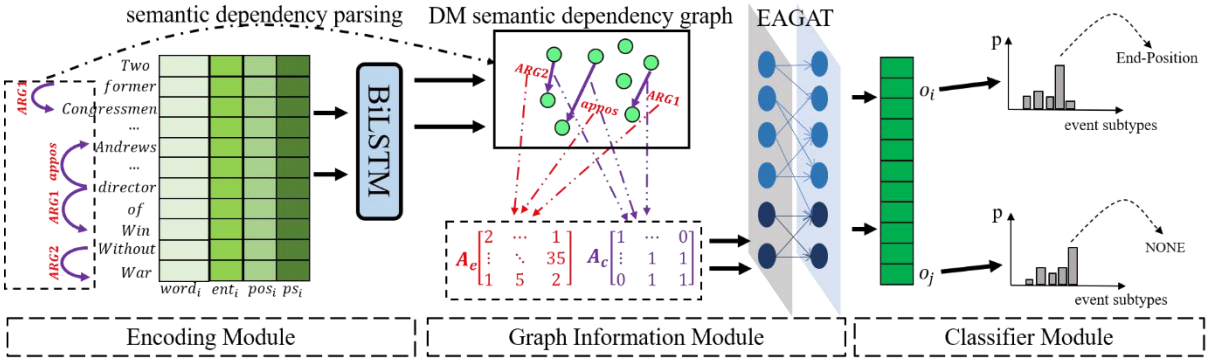

Figure 2. Overall architecture of the proposed SemEAGAT method. First, an encoding module that includes embeddings and a contextual encoder, transforms each input sentence into the local contextualized feature representations. Then, a graph information module follows, which consists of a graph construction component used for constructing the semantic dependency graph within the sentence, and a graph encoder component incorporates the adjacency matrix A_c and the edge embedding matrix A_e of the graph with EAGAT for capturing informative structure information. Finally, a classifier module is employed for predicting the task-related classification result. The final feature representation o_i and o_j correspond to "former" and "War" respectively.

3.1 Encoding module

The encoding module includes the following two components, namely embedding layer and contextual encoder.

Embedding Layer. For a sentence of n words, we denote this sentence $S=\{w_1, w_2, ..., w_i, ..., w_n\}$, where n is the length of sentence S. Each token w_i is transformed to a real-valued vector x_i by concatenating the following vectors:

- The word embedding vector of w_i: we use the word embedding pre-trained by Skip-gram [18] on the NYT Corpus following previous researches [10] [14] [17].
- The POS-tagging label embedding vector of w_i: we obtain this by looking up the randomly initialized POS-tagging label embedding table following previous [9].
- The entity type embedding vector of w_i: we obtain this through looking up the randomly initialized entity type embedding table.
- The position embedding vector of w_i: to indicate that w_c is the current word, we encode the relative distance $i-c$ from w_i to w_c as the position embedding vector following [8] [9]. And we obtain this vector through looking up the randomly initialized position embedding table.

Contextual Encoder. BiLSTM has been proven to have advantages in capturing contextual semantic information of words in the sentence [14] [19]. Besides, the ability of LSTMs and GCNs to capture semantic information may be complementary to a degree [8] [9], thus, before feeding into the

next graph information module, we employ BiLSTM to encode the representation vector sequence X for capturing local contextual semantic information of each token in the sentence.

To be specific, we conduct a forward LSTM and a backward LSTM over the representation vector sequence $X = \{x_1, x_2, ..., x_i, ..., x_n\}$ to generate the forward hidden vector sequence $\overrightarrow{h} = \{\overrightarrow{h_1}, \overrightarrow{h_2}, ..., \overrightarrow{h_i}, ..., \overrightarrow{h_n}\}$ and the backward hidden vector sequence $\overleftarrow{h} = \{\overleftarrow{h_1}, \overleftarrow{h_2}, ..., \overleftarrow{h_i}, ..., \overleftarrow{h_n}\}$ respectively. For each token x_i, we concatenate its forward representation $\overrightarrow{h}$ and backward representation $\overleftarrow{h}$, producing a new contextualized representation vector sequence $h_c = \{[\overrightarrow{h_1}, \overleftarrow{h_1}], [\overrightarrow{h_2}, \overleftarrow{h_2}], ..., [\overrightarrow{h_i}, \overleftarrow{h_i}], ..., [\overrightarrow{h_n}, \overleftarrow{h_n}]\}$, where $[,]$ means the concatenation operation. The new contextualized representation h_c will be fed into the next module as the initial graph node feature representations.

$$\overrightarrow{h} = \overrightarrow{LSTM}(x_1, x_2, ..., x_i, ..., x_n) \tag{1}$$

$$\overleftarrow{h} = \overleftarrow{LSTM}(x_1, x_2, ..., x_i, ..., x_n) \tag{2}$$

$$h_c = [\overrightarrow{h}; \overleftarrow{h}] \tag{3}$$

3.2 Graph information module.

The graph information module consists of two components which are the graph construction component and the graph encoder component.

Graph Construction. Semantic dependency graphs represent sentence-internal predicate-argument relationships between content words, having various semantic representation formalizations such as DELPH-IN MRS-Derived Bi-Lexical Dependencies (DM), Enju Predicate-Argument Structures (PAS), Prague Semantic Dependencies (PSD), corresponding to different annotation systems [11]. In this paper, we leverage DM as the representation formalization, and we generate the semantic dependency graphs by applying the state-of-the-art semantic dependency parsing model of Wang et al. [20].

We denote $G = (V, E)$ as the semantic dependency graph for sentence S where $V = \{v_1, v_2, ..., v_i, ..., v_n\}$ and E is the set of nodes and edges of G respectively. In V, each node v_i is correspond to each token w_i in S. Each edge (v_i, v_j) in E indicates there existing a directed edge from the head word w_i to the dependent word w_j with a label denoted $L(w_i, w_j)$. For instance, in Fig. 1, there is a directed edge from "*Without*" to "*War*" labeled "*ARG2*".

For the purpose of allowing information to flow from each word w_i to its governor word as well as itself, and for preventing the constructed input graph from too sparse, we follow [9] [17] to add all self-loops and the inverse edges into the original edge sets E, producing a new edge sets E' as:

$$\begin{aligned} E' = E \cup \{(w_i, w_i) | 1 \le i \le n\} \\ \cup \{(w_j, w_i) | (w_i, w_j) \in E\} \end{aligned} \tag{4}$$

The newly added edges also have directions and labels. We further consider that when the direction of an edge changes, influences produced by the label on the edge (representing the specific dependency relationship from head word to dependent word) should also change. Therefore, we expand our label set (we denote the original label set L) by adding new reverse labels corresponding to the original labels, resulting in a nearly doubled label set L':

$$L' = L(w_i, w_j) \cup L_r(w_j, w_i) \cup \langle SELF - LOOP \rangle \tag{5}$$

where $\langle SELF - LOOP \rangle$ represents the labels on the self-loop edges. Note that different from the previous researches [8] [9] [10] [17] ignore dependency edge information in syntactic dependency trees, we further integrate this type of informative information within the semantic dependency graph into our model to make better use of semantic dependency structure information. In other words, we not only process the adjacency matrix A_c representing the connectivity, but also transform the

dependency edge information into an edge matrix A_e, where $A_{e_{ij}}$ represents the corresponding $L'(w_i, w_j)$ (if $E'(w_i, w_j)$ exists).

Graph Encoder. The graph encoder component based on GAT applies an additional multi-head attention for integrating the rich information on edges, so that our EAGAT can perform information aggregation in an edge-aware way. We introduce GAT first.

a) Graph Attention Network: GAT [7] has the ability that weights the importance of neighbors due to the attention mechanism, thus producing effective representations. For simplicity, we introduce the core layer of GAT, i.e., graph attentional layer.

The input of the k-th graph attentional layer is an adjacent matrix A and a set of node (word) representations, denote $h^{k-1} = \{h_1^{k-1}, h_2^{k-1}, ..., h_i^{k-1}, ..., h_N^{k-1}\}$, where is N the number of nodes (words). And this layer's output is an updated node feature representation set, denote $h^k = \{h_1^k, h_2^k, ..., h_i^k, ..., h_N^k\}$.

During the calculation of the k-th graph attentional layer, the feature representation of node i will be updated through a multi-head attention, which performs the information aggregation process according to the neighbor set $Neighbor(i)$ of node i. Note that the $Neighbor(i)$ originates from the adjacent matrix A, and the process can be formulized as:

$$h_i^k = \|_{z=1}^{Z} \sigma\left(\sum_{j \in Neighbor(i)} \alpha_{ij}^{kz} W^{kz} h_j^{k-1}\right) \tag{6}$$

where $\|$ represents the concatenation operation, W^{kz} is a weight matrix of the z-th head at the k-th layer, Z is the number of attention heads, and σ represents a non-linear activation function, e.g., sigmoid. a_{ij}^{kz} is a normalized attention coefficient which measures h_i's dependence on h_j, and the a_{ij}^{kz} is computed as:

$$\alpha_{ij}^{kz} = \frac{\exp\left(f\left(h_i^{k-1}, h_j^{k-1}\right)\right)}{\sum_{p \in Neighbor(i)} \exp\left(f\left(h_i^{k-1}, h_p^{k-1}\right)\right)} \tag{7}$$

where f represents the LeakyReLU nonlinearity (with negative input slope $\alpha = 0.2$).

b) Edge-Aware Graph Attention Network: As we introduced the details of GAT, it takes an adjacent matrix of the graph as its input, which exploits the connectivity of the input graph well. However, the rich information on the semantic dependency edges has not yet been perceived. Inspired by the edge-agnostic phenomenon and the fact that different edge information is different in importance, we apply an additional multi-head attention to integrating edge information into the information aggregation process.

We maintain a randomly initialized edge embedding table according to the edge matrix A_e, which means that we transform $L'(w_i, w_j)$ into a real-valued vector l_{ij} through looking up the edge embedding table. Thus, we can add l_{ij} into the calculation of the graph attentional layer by an additional similar multi-head attention in GAT. We can get the edges aware attention weight through calculating as:

$$\overline{\alpha_{ij}} = \frac{\exp\left(f\left(h_i^{k-1}, h_j^{k-1}\right) + f\left(h_i^{k-1}, l_{ij}\right)\right)}{\sum_{p \in Neighbor(i)} \exp\left(f\left(h_i^{k-1}, h_p^{k-1}\right) + f\left(h_i^{k-1}, l_{ip}\right)\right)} \tag{8}$$

where f represents the aforementioned LeakyReLU nonlinearity. The $\overline{\alpha_{ij}}$ considers both node feature representations and edge feature representations, which is vital for the model implicitly perceiving edge information during aggregating information from neighbors. Moreover, in order to explicitly enhance the node feature representations by the edges' features, we explicitly add the l_{ij} into the information aggregation process for each word i as the following formula, where W_l^k is a weight matrix of this layer. Then, we take the h_i^k as the output of the k-th EAGAT layer, and the output of each word in the last EAGAT layer will be fed into the next classifier module. For convenience, we denote the final output feature representation as O, where $O = \{o_1, o_2, ..., o_i, ..., o_N\}$.

$$h_i^k = \|_{z=1}^{Z} \sigma\left(\sum_{j \in Neighbor(i)} \overline{\alpha_{ij}^{kz}} \left(W^{kz} h_j^{k-1} + W_l^k l_{ij}\right)\right) \quad (9)$$

3.3 Classifier module

We finally take a fully-connected network as the classifier, and the following is a softmax function used for computing the probability distribution $p(y_{et})$ as:

$$p(y_{et}) = softmax(W_{et} O + b_{et}) \quad (10)$$

where W_{et} is a parameter matrix which maps the O to the score for each event subtype and b_{et} is a bias item. After softmax function calculating, we choose the event subtype corresponding to the largest probability as our final predicted classification result. As general, we adopt cross entropy as the loss function for fine-tuning.

4. Experiments

4.1 Experiment settings

Dataset and Evaluation Metrics. We conduct experiments on ACE2005 dataset, which is the standard benchmark dataset for event decection. For comparison, we use the same data split as the previous work [2] [3] [4] [14] [21], where the same 40 newswire articles as test set, the same 30 articles as development set, and the rest 529 articles are used for training. Similar to previous [3] [4], we use the following criteria to judge the correctness of each predicted event trigger: A trigger is correct only if its span and event subtype match those of a golden trigger. Moreover, we use Precision (P), Recall (R), and F measure (F1) as evaluation metrics.

Hyper Parameter Setting. The hyperparameters are tuned on the development set. In the embedding layer, we set the dimension of word embeddings to 100, and that of the rest three types are to 50. In addition, the dimension of edge embeddings is set to 30. The hidden state size of BiLSTM and EAGAT are set to 250, and we use an one-layer BiLSTM and a two-layer EAGAT, respectively. We set the batch size to 32 and limit the max length of sentence to 50 (by padding shorter and cutting longer sentences). We adopt Adam optimizer with a learning rate of 1e-3. The dropout rate is set to 0.1 and the number of attention heads is set to 5. We preprocess the data by the Standford CoreNLP toolkit[2], and perform semantic dependency pasring by employing [20].

4.2 Overall performance

We compare our model with the state-of-the-art models, which can be classified into three classes: feature-based models, sequence-based models and GCN-based models.

Feature-based models. The feature-based models utilize elaborately human designed features for ED, including the following two: 1) CrossEntity [1]: utilizes cross-entity information to perform ED. 2) jointModel [2]: utilizes global features and local features.

Sequence-based models. The sequence-based models view the given sentence as a word-sequence, including the following six: 1) DMCNN [3]: designs a dynamic multipooling CNN model for ED. 2) JRNN [4]: applies BiRNN with memory matrixes (vectors) to improve ED. 3) dbRNN [21]: adds dependency bridges on BiLSTM to improve ED. 4) HBTNGMA [14]: employs hierarchical and bias tagging networks to detect multiple events in a given sentence collectively. 5) Bi-LSTM+GAN [15]: employs generative adversarial network to conduct self-regulation for ED. 6) LearnToSelectED [16]: applies hard attention with Gumbel-Softmax trick for ED.

GCN-based models. The GCN-based models employ Graph Convolutional Networks (GCNs) over the syntactic dependency tree of a given sentence to exploit syntactic information, including the following four: 1) GCN-ED [8]: employs GCN and utilizes an entity mention-guided pooling to perform ED. 2) JMEE [9]: applies GCN with syntactic shortcut arcs to enhance information flow. 3)

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

MOGANED [10]: proposes a multi-order GAT to model the multi-order syntactic dependency. 4) HGEED [17]: builds both sentence graph and document graph to improve ED.

Table 1. Overall performance on ACE2005 test set. Bold marks the highest score among all models. † means whe sequence-based models, ‡ means the GCN-based models.

Method	**Trigger Identification + Classification (%)**		
	P	R	F1
CrossEntity [1]	72.9	64.3	68.3
jointModel [2]	73.7	62.3	67.5
DMCNN† [3]	75.6	63.6	69.1
JRNN† [4]	66.0	73.0	69.3
dbRNN† [21]	74.1	69.8	71.9
HBTNGMA† [14]	77.9	69.1	73.3
Bi-LSTM+GAN† [15]	71.3	74.7	73.0
LearnToSelectED† [16]	75.4	**75.0**	75.2
GCN-ED‡ [8]	77.9	68.8	73.1
JMEE‡ [9]	76.3	71.3	73.7
MOGANED‡ [10]	79.5	72.3	75.7
HGEED‡ [17]	80.1	72.7	76.2
SemEAGAT‡	**80.7**	73.2	**76.8**

Table 1 presents the experimental results on ACE2005 test set. We can get the most important observation that our SemEAGAT outperforms all the compared models on P and F1. To be specific, SemEAGAT achieves the best performance on P and F1 score, with an absolute improvement of 0.6% to HGEED, proposed in 2021, is the newest and best model for comparison. We consider it mainly because HGEED is based on the co-occurrence relationship and similarity relationship when additionally utilizes document information. But in some cases these two relationships tend to predict a non-trigger to be the golden trigger (e.g., "*War*" in Fig. 1). By contrast, our SemEAGAT can overcome such misprediction problem by utilizing the predict-argument deep semantic dependency relationship. In addition, the improvement we gain also proves that the deep structure information within one sentence is worthy of further use in ED.

Furthermore, comparing with the GAT-based MOGANED, SemEAGAT surpasses it on both three metrics and improves by 1.1% on F1, which reflects the advantages that incorporate the semantic dependency graphs in an edge-aware way. Besides, when only the sentence-level information is used, both MOGANED and SemEAGAT perform a high performance on F1, which suggests the effectiveness of the employed GAT. And the result that both MOGANED and SemEAGAT have a particularly high P further proves that the attention mechanisms automatically weight different neighbors can help to predict event triggers more precisely. Moreover, comparing with dbRNN, HBTNGMA, GCN-ED, JMEE, MOGANED, and HGEED, those of syntactic-based neural network models respectively, our semantic-based obtains a consistent increase on P, which mainly benefit from the advantage of less irrelevant information and deeper semantic relationship in the semantic dependency structure.

In addition, among the compared models which utilize syntactic dependency structure, we notice that the GCN-based models, i.e., GCN-ED, JMEE, MOGANED, and HGEED, usually perform better

on F1 than the sequence-based model dbRNN, and should note that at the same time, our semantic structure based SemEAGAT performs the best F1. These observations clearly demonstrate the effectiveness of graph neural network methods on information exchange and the necessity of exploiting the graph structures within a sentence, and also suggest the effectiveness of the semantic dependency graph structure. Comparing with Bi-LSTM+GAN and LearnToSelectED which are relatively new in learning techniques, our SemEAGAT is exceeded by them on R and show an apparent improvement on P, this can be attributed to the efforts that our model focus on more precisely semantic representations while these two models for comparison pay more attention to recognize more trigger candidates. As for comparing the feature-based models with the other neural network methods, the results show the effectiveness of neural network models to automatically capture feature representations.

4.3 Analysis

Ablation Study There are two major components in our SemEAGAT model, one is the contextual encoder component in section 3.1, which aims to encode effective local contextual representation for each word, the other is the graph encoder component in section 3.2, which aims to encode the semantic dependency structure through GAT in an edge-aware way. To demonstrate the effect of the two important components, we conduct experiments with different variations toward them. Specifically, we set 1) SemEAGAT: it is our complete model shown for comparison, 2) -BiLSTM: it means we remove the first major component (i.e., contextual encoder), 3) with BiRNN: it indicates that we use RNN units instead of LSTM units, 4) -EA: it means that we remove the additional multi-head attention, 5) -EAGAT: it means that we remove the total graph information module.

Table 2. Ablation study on ACE2005 test set.

Method	**P**	**R**	**F1**
SemEAGAT	**80.7**	73.2	**76.8**
-BiLSTM	69.8	70.9	70.3
with BiRNN	73.0	**74.3**	73.6
-EA	76.4	72.7	74.5
-EAGAT	73.2	71.4	72.3

The experimental results are shown in Table 2. All of these four modified models perform worse than SemEAGAT, which suggests that both the major components play an important role in event detection. -BiLSTM achieves the worst performance, which indicates the great advantage of BiLSTM in capturing contextual representation, and the dropping of F1 in with BiRNN can further prove this advantage of BiLSTM. Compared to SemEAGAT, -EA drops by an absolute margin of 2.3% on F1, which demonstrates the rich semantic information on the semantic dependency edges and proves that the integration of edge information helps a lot and the edge-aware way in the information aggregation process of GAT is indeed effective. Moreover, there are two graph-based models (which are SemEAGAT and -EA) that perform a relative improvement when comparing with -EAGAT, verifying the effectiveness of incorporating semantic dependency structure information with GAT for ED, and also indicating the necessity of mining the graph structures in the sentence. Furthermore, compared to -EAGAT, -EA surpasses it mainly on P, which suggests that incorporates semantic dependency graph with GAT more likely gain on P, and this is consistent with our intuition that the incorporation may provide more accurate and effective information for ED, thus our SemEAGAT can overcome misprediction problems (e.g., "*War*" in Fig. 1) by utilizing the predict-argument deep semantic dependency relationship.

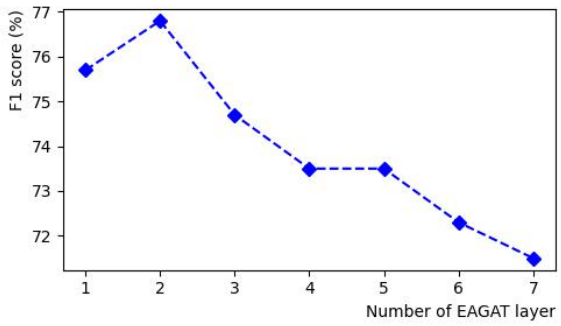

Figure 3. Effect of the EAGAT layer number on ACE2005 test set.

Effect of the EAGAT layers We further investigate the influences produced by different EAGAT layer numbers on the final performance. We set the number of EAGAT layers to at most 7. The experimental results are shown in Fig. 3. We can get an observation that at the beginning, as the number of EAGAT layers increases, the F1 score also shows an upward trend, which indicates a single layer of EAGAT is not sufficient for the information to flow fully, and increasing the number of EAGAT layers, i.e., stacking EAGAT Layers, can make information more fully exchanged and utilized between nodes, thereby improving the performance. However, as the number of layers continues to increase, we find that the model's performance turns to a downward trend. Therefore, according to the experimental results of parameter tuning, the optimal number of layers is set to 2. And such decline of performance can be interpreted as that with the continuous increase of layer numbers, the feature representations of nodes become more and more similar due to the information aggregation process is performed through neighbor nodes in the semantic dependency graphs. And this phenomenon is called over-smoothing [22], which means that the features of all nodes tend to be consistent.

5. Conclusion

In this paper, we propose a novel semantic dependency edges aware graph attention network based method (SemEAGAT) for improving event detection, which investigates semantic dependency graph in the sentence and incorporates the DM semantic dependency graph with the graph attention network and an additional multi-head attention for ED in an edge-aware way that utilizes both the connectivity and semantic dependency information on the semantic dependency edges. Experimental results on the widely used ACE 2005 dataset well demonstrate the effectiveness of the proposed method, achieves the better performance especially in Precision. In the future, we also plan to apply SemEAGAT to other information extraction tasks, such as event argument extraction, relation extraction etc. to enhance the extraction to be more exactly.

Acknowledgments

We would like to appreciate anonymous reviewers for their review comments. This research is supported by the National Key RD Program of China (2017YFC0405806, 2018YFC0407901).

References

[1] Yu Hong, Jianfeng Zhang, Bin Ma, Jianmin Yao, Guodong Zhou, and Qiaoming Zhu. 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, 2011.

[2] Qi Li, Heng Ji, and Liang Huang. 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, 2013.

[3] 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 (Volume 1: Long Papers)*, pages 167– 176, 2015.

[4] 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 300– 309, 2016.

[5] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *5th International Conference on Learning Representations, ICLR*, Toulon, France, April 2017.

[6] Linfeng Song, Yue Zhang, Zhiguo Wang, and Daniel Gildea. A graph-to-sequence model for amr-to-text generation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1616– 1626, 2018.

[7] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *International Conference on Learning Representations*, 2018.

[8] Thien Huu Nguyen and Ralph Grishman. Graph convolutional networks with argument-aware pooling for event detection. In *Thirty-second AAAI conference on artificial intelligence*, 2018.

[9] Xiao Liu, Zhunchen Luo, and He-Yan Huang. 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, 2018.

[10] Haoran Yan, Xiaolong Jin, Xiangbin Meng, Jiafeng Guo, and Xueqi Cheng. Event detection with multi-order graph convolution and aggregated attention. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (*EMNLP-*IJCNLP)*, pages 5766– 5770, 2019.

[11] Stephan Oepen, Marco Kuhlmann, Yusuke Miyao, Daniel Zeman, Silvie Cinková, Dan Flickinger, Jan Hajic, and Zdenka Uresova. Semeval 2015 task 18: Broad-coverage semantic dependency parsing. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pages 915– 926, 2015.

[12] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998– 6008, 2017.

[13] David Ahn. The stages of event extraction. In *Proceedings of the Workshop on Annotating and Reasoning about Time and Events*, pages 1– 8, 2006.

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

[15] Yu Hong, Wenxuan Zhou, Jingli Zhang, Guodong Zhou, and Qiaoming Zhu. 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, 2018.

[16] Nghia Trung Ngo, Tuan Ngo Nguyen, and Thien Huu Nguyen. Learning to select important context words for event detection. *Advances in Knowledge Discovery*

and Data Mining, 12085:756, 2020.

[17] Jianwei Lv, Zequn Zhang, Li Jin, Shuchao Li, Xiaoyu Li, Guangluan Xu, and Xian Sun. Hgeed: Hierarchical graph enhanced event detection. *Neurocomputing*, 453:141– 150, 2021.

[18] Tomas Mikolov, Kai Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. In *ICLR*, 2013.

[19] Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. Neural architectures for named entity recognition. In *HLT-NAACL*, 2016.

[20] Xinyu Wang, Jingxian Huang, and Kewei Tu. Second-order semantic dependency parsing with end-to-end neural networks. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4609– 4618, 2019.

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

[22] Jie Zhou, Ganqu Cui, Shengding Hu, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, and Maosong Sun. Graph neural networks: A review of methods and applications. *AI Open*, 1:57– 81, 2020.