Event Detection with Multi-Order Graph Convolution and Aggregated Attention

Haoran Yan[†‡]**, Xiaolong Jin**[†‡]**, Xiangbin Meng**[§]**, Jiafeng Guo**[†‡] **and Xueqi Cheng**[†‡]

[†]CAS Key Laboratory of Network Data Science and Technology,
Institute of Computing Technology, Chinese Academy of Sciences (CAS);
[‡]School of Computer and Control Engineering, The University of CAS
{yanhaoran17s, jinxiaolong, guojiafeng, cxq}@ict.ac.cn
[§]School of Computer Science, Beijing University of Posts and Telecommunications
mengxb@bupt.edu.cn

Abstract

Syntactic relations are broadly used in many NLP tasks. For event detection, syntactic relation representations based on dependency tree can better capture the interrelations between candidate trigger words and related entities than sentence representations. But, existing studies only use first-order syntactic relations (i.e., the arcs) in dependency trees to identify trigger words. For this reason, this paper proposes a new method for event detection, which uses a dependency tree based graph convolution network with aggregative attention to explicitly model and aggregate multi-order syntactic representations in sentences. Experimental comparison with state-of-the-art baselines shows the superiority of the proposed method.

1 Introduction

As an important information extraction task, event detection aims to find event mentions with specific event types from given texts. Each event mention is identified by a word or phrase called event trigger, which serve as the main word(s) to the corresponding event. For example, in the sentence presented in Figure 1, an event detection system should be able to recognize the trigger word "fired" corresponding to the event type "Attack".

Among existing methods for event detection, sequence based ones (e.g., Chen et al. (2015); Nguyen et al. (2016)) only use the given sentences, which suffer from the low efficiency problem in capturing long-range dependency; On the contrary, dependency tree based methods utilize the syntactic relations (i.e., arcs) in the dependency tree of a given sentence to more effectively capture the interrelation between each candidate trigger word and its related entities or other triggers. For example, Nguyen and Grishman (2018) and Liu et al. (2018) treat each dependency tree as a graph and adopt Graph Convolution Network (GCN) (Kipf and Welling, 2016) to extract event triggers.

The syntactic relations between trigger words and related entities may be of first order, embodied as the direct arcs in the dependency tree. They may also be of high-order (i.e., the paths more than one hop in the dependency tree). Particularly, according to our statistics on the benchmark ACE-2005 dataset, about 51% (4977/9793) of event related entities need more than one hop to get to the corresponding trigger words and related entities in their dependency trees. For example, in Figure 1, it needs at least 4 hops (i.e., "fired"–"evidence"–"blood"–"soldiers") to figure out that the word "fired" means "shot" instead of "dismissed".

However, the above dependency tree based methods explicitly use only first-order syntactic relations, although they may also implicitly capture high-order syntactic relations by stacking more GCN layers. However, as the number of GCN layers increases, the representations of neighboring words in the dependency tree will get more and more similar, since they all are calculated via those of their neighbors in the dependency tree. This is the so-called over-smoothing problem (Zhou et al., 2018), which damages the diversity of the representations of neighboring words.

To overcome this problem, this paper proposes a Multi-Order Graph Attention Network based method for Event Detection, called MOGANED. It utilizes both first-order syntactic graph and high-order syntactic graphs to explicitly model multi-order representations of candidate trigger words. In order to calculate multi-order representations for each word, we apply Graph Attention Network (GAT) proposed in (Veličković et al., 2017) to weight the importance of its neighboring words in the syntactic graphs of different orders. We then employ an attention aggregation mechanism to merge its multi-order representations. Finally,
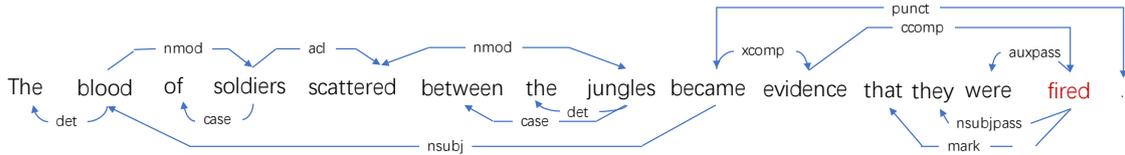
5765

Figure 1: An example sentence with the dependency parsing result.

through experimental comparison with state-of-the-art baselines we show the superiority of the proposed method in terms of both precision and F1-measure. It should be mentioned that, to the best of our knowledge, this is the first study applying GAT to event detection.

2 The Proposed Model

Following the existing studies, we regard event detection as a multi-class classification problem. Let $W = w_1, w_2, ..., w_n$ be a sentence of length n, where w_i is its i-th token. Since event triggers may contain multiple words, we adopt the "BIO" schema to make annotation. The number of labels is thus be $2L + 1$, where L is the number of predefined event types.

The proposed model contains three modules: (i) word encoding, which encodes input sentence to a sequence of vectors; (ii) multi-order graph attention network that performs graph attention convolution over multi-order syntactic graphs; (iii) attention aggregation, which aggregates multi-order representations of each word with different attention weights to predict its label.

2.1 Word Encoding

The word encoding module first transforms each input token w_i into a comprehensive embedding vector x_i by concatenating its word embedding $word_i$, entity type embedding et_i, POS-tagging embedding pos_i and position embedding ps_i (Nguyen et al., 2016; Nguyen and Grishman, 2018), where $word_i$ is obtained by looking up a pre-trained word embedding table on a large corpus and others are randomly initialized.

The input sentence W will then be transformed to a sequence of vectors $X = x_1, x_2, ..., x_n$. Since each word can only be updated by its neighbors in the dependency tree through dependency arcs, following the previous methods (Liu et al., 2018), we employ a Bidirectional Long-Short Term Memory network (BiLSTM) (Hochreiter and Schmidhuber, 1997) to encode X with its context as $P = p_1, p_2, ..., p_n$, which will be used as the input of

the multi-order GAT. Here,

$$p_i = [\overrightarrow{LSTM}(x_i)||\overleftarrow{LSTM}(x_i)], \quad (1)$$

where $||$ is the concatenation operation.

2.2 Multi-Order Graph Attention Network

Each dependency tree can be represented as a first-order syntactic graph in terms of an adjacency matrix. Let A be the adjacency matrix of the first-order syntactic graph, which is generated from the dependency tree of the sentence, W. A contains three sub matrixes A_{along}, A_{rev} and A_{loop} (Marcheggiani and Titov, 2017) of the same dimensions $n \times n$. $A_{along}(i, j) = 1$, if there is a dependency arc from w_i to w_j in the dependency tree, otherwise 0. As the reverse graph of A_{along}, $A_{rev} = A_{along}^T$. A_{loop} is an identity matrix.

The adjacency matrix of the k-th-order syntactic graph is $A_{subg}^k = (A_{subg})^k$, where $subg \in \{along, rev, loop\}$. A_{subg}^k records all k-hop paths of A_{subg}. Note that $A_{loop}^k = A_{loop}$, since it is an identity matrix. For simplicity, in what follows we use a^k, b^k, c^k to denote A_{along}^k, A_{rev}^k and A_{loop}^k, respectively.

The multi-order GAT module simply learns a list of representations over multi-order syntactic graphs by a few parallel GAT layers (Veličković et al., 2017), which weights the importance of neighbors of each word in each syntactic graph during convolution.

The representations h_i^k of the k-th-order syntactic graph A^k are calculated from the representations of the subgraphs of A^k:

$$h_i^k = f(p_i, a^k) \oplus f(p_i, b^k) \oplus f(p_i, c^k), \quad (2)$$

where $f(\cdot)$ is the graph attention convolution function and $\oplus$ is element-wise addition, and

$$f(p_i, a^k) = \sigma \sum_{j=1}^{n} (u_{ij} a_{ij}^k (W_{a,k} p_j + \epsilon_{a,k})), \quad (3)$$

where σ is exponential linear unit (ELU) (Clevert et al., 2015); $W_{a,k}$ and $\epsilon_{a,k}$ are the weight matrix and bias item for a^k; u_{ij} is the normalized weight of the neighbor w_j when updating w_i,
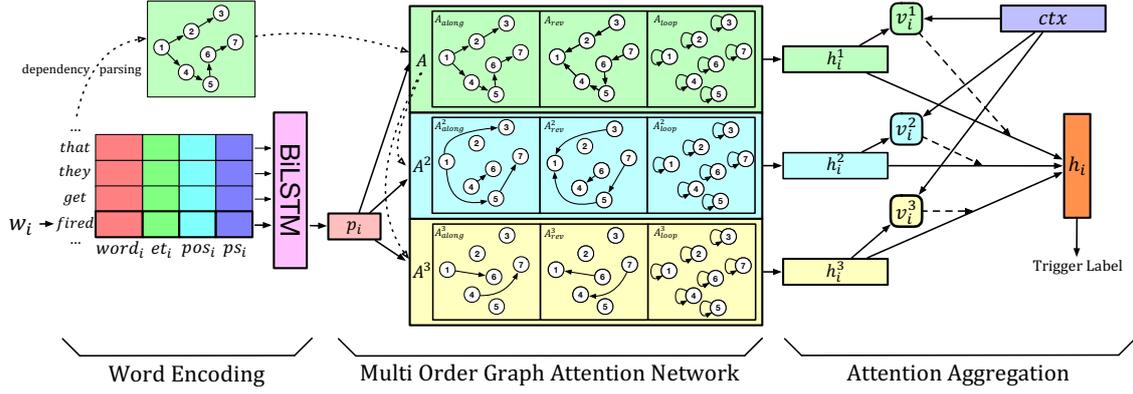
Figure 2: The illustrative architecture of the proposed MOGANED method, where K=3.

$$u_{ij} = softmax(e_{ij}) = \frac{exp(e_{ij})}{\sum_{j \in \mathcal{N}_i} exp(e_{ij})}, \quad (4)$$

where $e_{ij} = \gamma(W_{comb}[W_{att}p_i||W_{att}p_j])$. Here, $\mathcal{N}_i$ is the neighbor set of w_i in the subgraph, γ is LeakyReLU (with negative input slope $\alpha = 0.2$) (Maas et al., 2013); W_{comb} and W_{att} are weight matrices.

After graph attention convolution, each candidate trigger word w_i gets a list of multi-order representations $h_i^k, k \in [1, K]$, where K is the highest order used in this module.

2.3 Attention Aggregation

To aggregate the list of multi-order representations h_i^k of each word w_i, we employ a weighted attention aggregation mechanism from Hierarchical Attention Networks (Yang et al., 2016):

$$h_i = \sum_{k=1}^{K} v_i^k h_i^k, \quad (5)$$

where v_i^k is the normalized weight of the k-th order graph representation of the word w_i, which is calculated by

$$v_i^k = softmax(s_i^k) = \frac{exp((s_i^k)^T ctx)}{\sum_{j=1}^{K} exp((s_i^j)^T ctx)}, \quad (6)$$

where $s_i^j = tanh(W_{awa}h_i^j + \epsilon_{awa})$. Here, W_{awa} and ϵ_{awa} are the weight matrix and the bias term, respectively; ctx is a randomly initialized context vector, which captures the importance of graph representations of each order.

Finally, we use the aggregated representation h_i to predict the trigger label of word w_i as follows:

$$y_i^t = softmax(O_i^t) = \frac{exp(O_i^t)}{\sum_{q=1}^{2L+1} exp(O_i^q)}, \quad (7)$$

where y_i^q denotes the probability assigning label q to word w_i and $O_i = w_o h_i + \epsilon_o$. Here, w_o and ϵ_o are the weight matrix and the bias item, respectively.

2.4 Bias Loss Function

Since the number of "O" labels in the data is much larger than that of event labels, we use a bias loss function (Chen et al., 2018) to enhance the influence of event labels during training:

$$J(\theta) = -\sum_{i=1}^{N_s} \sum_{j=1}^{N_{i,w}} (\log p(y_j^t|s_i, \theta) \cdot I(O) + \\ \lambda \log p(y_j^t|s_i, \theta) \cdot (1 - I(O))), \quad (8)$$

where N_s is the number of sentences; $N_{i,w}$ is the number of words in s_i; $I(O)$ equals 1, if the label of the word is "O"; otherwise 0; λ is the weight parameter larger than 1.

3 Experiments

3.1 Experiment Settings

Dataset and Resources We compare our MOGANED model with baseline methods on ACE-2005 with the same data split, where 40 newswire documents are used as the test set, 30 newswire documents as the validation set and 529 remained documents as the training set.

For data preprocessing, we use the Stanford CoreNLP toolkit for sentence splitting, tokenizing, POS-tagging and dependency parsing. We adopt word embeddings trained over the New York Times corpus with the Skip-gram algorithm (Mikolov et al., 2013; Chen et al., 2018).

Hyper Parameter Setting The hyper parameters are tuned on the validation set. For all experiments

Method	P	R	F_1
Cross Event	68.7	68.9	68.8
DMCNN	75.6	63.6	69.1
JRNN	66.0	73.9	69.3
DEEB-RNN	72.3	**75.8**	74.0
dbRNN†	74.1	69.8	71.9
GCN-ED†	77.9	68.8	73.1
JMEE†	76.3	71.3	73.7
MOGANED†	**79.5**	72.3	**75.7**

Table 1: Overall performance of different methods on the test set with gold-standard entities. † indicates that the method uses dependency arcs.

below, we use 100 as the dimension of word embeddings and 50 as that of the rest embeddings. We set the hidden units of the BiLSTM network to 100. We set the highest order K to 3 and the dimension of graph representation to 150. We set batch size to 30 and utilize a fixed maximum sentence length $n = 50$ by padding short sentences and cutting longer ones. During training, we use the AdaDelta update rule (Zeiler, 2012) with a learning rate of 0.001. We set the dropout rate to 0.3 and the L2-norm to $1e-5$. We set the bias loss parameter λ to 5.

3.2 Overall Performance

We compare our model with the following state-of-the-art baselines: 1) Cross Event (Liao and Grishman, 2010), which uses document level information for event extraction; 2) DMCNN (Chen et al., 2015), which builds a dynamic multi-pooling CNN model; 3) JRNN (Nguyen et al., 2016), which uses a bidirectional RNN and human designed features; 4) DEEB-RNN (Zhao et al., 2018), which uses hierarchical supervised attention with document level information for event detection; 5) dbRNN (Sha et al., 2018), which adds dependency arcs over a Bi-LSTM network to improve event extraction; 6) GCN-ED (Nguyen and Grishman, 2018), which uses an argument pooling mechanism for event detection based on GCN; 7) JMEE (Liu et al., 2018), which uses GCN with highway network and self-attention.

Table 1 presents the performance comparison between different methods. We can see that MOGANED achieves 1.6% and 1.7% improvement on precision and F_1-measure, respectively, compared with the best baselines. MOGANED reaches a lower recall than two sequence based

Method	P	R	F_1
MOGANED-First	71.0	69.9	70.4
MOGANED-GCN	75.7	71.1	73.3
MOGANED-Mean	77.3	70.9	74.0
MOGANED	**79.5**	**72.3**	**75.7**

Table 2: Performance of modified architectures based on MOGANED.

methods, JRNN and DEEB-RNN. This may be caused by the propagated error from the dependency parsing tool. There are long sentences where trigger words and their corresponding entities are parsed into two independent dependency trees without connection. Therefore, the trigger words and their related entities are unable to interact in multi-order graph attention network, which leads to a lower recall. However, MOGANED still achieves the best performance in terms of precision, recall, F_1-measure among all dependency based methods, which suggests the effective of multi-order representations.

3.3 Ablation Study

This study aims to validate the impacts of multi-order representation, graph attention network and attention aggregation. For this purpose, we design three architectures based on MOGANED: 1) MOGANED-First: it only uses first-order syntactic graph (i.e., $K=1$); 2) MOGANED-GCN: it uses traditional GCN instead of GAT; 3) MOGANED-Mean: it adopts mean pooling as the attention aggregation mechanism of multi-order representations for each word.

The experimental result is shown in Table 2. All of these three modified models get lower performance than MOGANED. MOGANED-First achieves the worst performance, which suggests that high-order syntactic relations play an important role in event detection. MOGANED-GCN drops more on precision than recall, which illustrates that the attention learned from GAT helps MOGANED predict trigger words more precisely. The performance drop of MOGANED-Mean is the smallest among the three modified models. Although the average of multi-order representations achieves competitive performance for event detection, the proposed attention aggregation module still distinguishes the importance of syntactic representations of different order, which achieves 1.7% improvement on F_1-measure.

4 Conclusion

In this paper, we proposed the MOGANED model for modeling multi-order representations via GAT and employed an attention aggregation mechanism to better capture dependency contextual information for event detection. The experimental results well demonstrate its effectiveness and superiority.

References

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

Yubo Chen, Hang Yang, Kang Liu, Jun Zhao, and Yantao Jia. 2018. 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.

Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. 2015. Fast and accurate deep network learning by exponential linear units (elus). *arXiv preprint arXiv:1511.07289*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

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

Shasha Liao and Ralph Grishman. 2010. Using document level cross-event inference to improve event extraction. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 789–797. Association for Computational Linguistics.

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

Andrew L Maas, Awni Y Hannun, and Andrew Y Ng. 2013. Rectifier nonlinearities improve neural network acoustic models. In *Proc. icml*, volume 30, page 3.

Diego Marcheggiani and Ivan Titov. 2017. Encoding sentences with graph convolutional networks for semantic role labeling. *arXiv preprint arXiv:1703.04826*.

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

Thien Huu Nguyen, Kyunghyun Cho, and Ralph Grishman. 2016. Joint event extraction via recurrent neural networks. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 300–309.

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

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

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2017. Graph attention networks. *arXiv preprint arXiv:1710.10903*.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016. Hierarchical attention networks for document classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1480–1489.

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

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

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