Computers, Materials & Continua
DOI: 10.32604/cmc.2023.031052
Article

Tech Science Press

Combing Type-Aware Attention and Graph Convolutional Networks for Event Detection

Kun Ding[1], Lu Xu[2], Ming Liu[1], Xiaoxiong Zhang[1], Liu Liu[1], Daojian Zeng[2,*], Yuting Liu[1,3] and Chen Jin[4]

[1]The Sixty-Third Research Institute, National University of Defense Technology, Nanjing, 210007, China
[2]Hunan Normal University, Changsha, 410000, China
[3]School of Computer & Software, Nanjing University of Information Science and Technology, Nanjing, 210044, China
[4]School of Computer Science, University of Manchester, Manchester, M13 9PL, UK
*Corresponding Author: Daojian Zeng. Email: zengdj916@163.com
Received: 09 April 2022; Accepted: 11 May 2022

Abstract: Event detection (ED) is aimed at detecting event occurrences and categorizing them. This task has been previously solved via recognition and classification of event triggers (ETs), which are defined as the phrase or word most clearly expressing event occurrence. Thus, current approaches require both annotated triggers as well as event types in training data. Nevertheless, triggers are non-essential in ED, and it is time-wasting for annotators to identify the "*most clearly*" word from a sentence, particularly in longer sentences. To decrease manual effort, we evaluate event detection without triggers. We propose a novel framework that combines Type-aware Attention and Graph Convolutional Networks (TA-GCN) for event detection. Specifically, the task is identified as a multi-label classification problem. We first encode the input sentence using a novel type-aware neural network with attention mechanisms. Then, a Graph Convolutional Networks (GCN)-based multi-label classification model is exploited for event detection. Experimental results demonstrate the effectiveness.

Keywords: Event detection; information extraction; type-aware attention; graph convolutional networks

1 Introduction

ED is aimed at detecting the occurrence of predefined events and categorizing them in plain text. For instance, considering the sentence "*In Baghdad, a cameraman died when an American tank fired on the Palestine Hotel.*", an appropriate system for ED ought to recognize 2 events, *Die* and *Attack* (supposing that both *Attack* and Die are in the predefined event set). ED is widely used in social media analytics and Knowledge construction [1,2].

Previously, this task was solved via recognition and classification of ETs. A trigger is the phrase or word that *most clearly* describes an event occurrence according to the Automatic Context Extraction

(ACE) event evaluation program. As shown in the example in Fig. 1, "*died*" is a trigger word for *Die* event, while "*fired*" is a trigger word for *Attack* event. Following ACE definition, ED aims to detect the position of ETs in raw text and classify them into corresponding event types. Consequently, this task has been modeled in most of the current methods as word classification [3–11], which predicts if and what type of event is being triggered by every word in a sentence.. Thus, these methods require both annotated triggers as well as types of events for training.

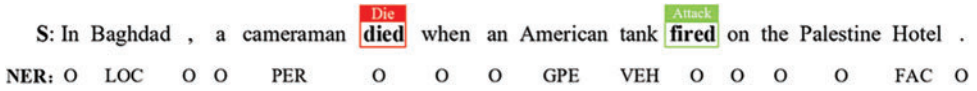

Figure 1: An event detection example annotated by ACE 2005. There are two events: *Die* and *Attack*

Nevertheless, ETs are dispensable to this task. Recall that ED is aimed at recognizing and categorizing events, thus triggers are intermediate outcomes of this task. We argue that it is unnecessary to identify the trigger words. In other words, we only need to know what events are included in a sentence.

Furthermore, identifying a word in a sentence which is "*most clearly*" is not an easy task for annotators, which restricts the applications of current ED methods. An event may be triggered by multiple types of words such as the verb, noun and pronoun. Each type of words may involve multiple forms as well; for example, the verb could be a past participle or in form of an adjective. According to the previous literature [3], human annotators can only approximately obtain an *F1* score of 73% on the ACE 2005 evaluation task. The low accuracy of training data hinders the performance of current approaches, which are under a supervised learning paradigm and rely on a large number of accurate training examples to ensure good performance.

Guided by these motivations, recent researchers explored detecting events without triggers which make the task more simplified. Consider example ***S*** in Fig. 1, its annotation is {*Die, Attack*}, the only annotated information for every sentence is the event type occurring in it. Contrasting, prior work required annotated triggers for every event, implying that the annotated information for ***S*** is {*Attack*:fired, *Die*:died}. After getting rid of the shackles of trigger words, annotate event type are more efficiently. Without ETs, it only needs to find out which events are included in the sentence, and not specifically focus on which word most clearly expresses the occurrence of events. Intuitively, the ED task can be modelled as a text classification problem. But, there are 2 problems:

Multi-label problem: every sentence may have a random total of events. In machine learning, this is referred to as the multi-label problem, which is more complex than single-label classification in that there is a strong label correlation. Take the sentence in Fig. 1 for example an attack event often results in death, While the death may be caused by an attack. As events normally co-occur in the physical world[1], another major challenge for ED without triggers is to model the label dependencies.

Trigger absence problem: With the help of the annotated trigger words, the machine learning model uses triggers as the supervision targets and can learn the pattern of extracting the trigger words. Then, as a strong feature, the trigger words help models to further determine event types. Without annotated trigger words, the core problem for multiple event classification is to learn the discriminative features.

Latest work has attempted to solve the *multi-label problem* by transforms multi-label classification into multiple binary classification problems [12]. This method predicts whether each event is included in a given sentence or not. A predefined type *t* event associated with a specified sentence s leads to

[1]More than 20% of the samples in the ACE 2005 dataset contain more than one event.

an instance that is labeled 0 or 1, depending on whether s receives the type t event. Thus, sentences conveying multiple events will show many positive pairs. Therefore, it theoretically guarantees that we can extract multiple events from one sentence. Benefited from the great success of learned features, the performance of the binary solution attains competitive performances, relative to state-of-the-arts that use annotated triggers. Unfortunately, this immature way to address the multi-label problem treats events in isolation. It cannot model the interaction among events by only exploiting this strategy.

We propose a novel framework, referred to as TA-GCN for event detection, which exploits type-aware attention neural network to learn distinguishing features and introduces GCN to capture the correlation among classifiers.

To address the *trigger absence problem*, we use a novel event type-aware neural networks with an attention mechanism to automatically learn distinguishing features. Very event type is typically triggered by specific words. But, in our task, annotated triggers are not available. For modeling this information, we suggest a simple, effective event type-aware model in this paper. Precisely, given a sentence, for the proposed model, input tokens are first transformed into embeddings after which it applies a Long Short-Term Memory (LSTM) layer to determine a context-dependent representation for every token. Then it calculates an attention vector, α, based on target event type, whereby the trigger word has a high score. Finally, sentence representation $\mathbf{S}_{att}$ is evaluated based on α. Through the type-aware attention strategy, the trigger information will be enhanced in the learned feature.

In this paper, we exploit Graph Convolutional Networks (GCN) to take event correlations into account. Instead of treating classifiers with randomly initialized parameters [12], we propose to consider the classifiers as the graph nodes and store the correlation among classifiers in the adjacency matrix. The adjacency matrix is randomly initialized and updated through the training data. The final classifiers are obtained via a GCN-based mapping function. As graph convolutional filters are shared across all class (i.e., event labels), gradients from classifiers affect the GCN based mapping function. Through this strategy, the label correlations will be modeled.

The event type-aware neural networks and GCN together constitute the entire ED model which enables end-to-end training. These two networks interact with each other through the sharing of classifier parameters and work in a mutually reinforcing way.

We have carried out extensive experiments using a benchmark dataset ACE 2005[2]. Our approach outperformed all the comparative baselines, and achieved higher performances relative to current methods that use annotated triggers and external data.

2 Related Work

Detection of events is a vital topic in NLP. Various methods have been shown to achieve this task. Almost all current methods on ACE event tasks follow a supervised paradigm. Recent research has been dominated by representational methods. A prominent feature of the neural network paradigm is the representation of mentions of candidate events by embeddings. Chen et al. [13] and Nguyen et al. [7] were the first work on this paradigm. Their models are Convolutional Neural Networks (CNN)-based. Nguyen et al. [14] suggested a method for joint event extraction that is Recurrent Neural Networks (RNN)-based for modeling the dependency of triggers as well as arguments. Liu et al. [15] proposed encoding argument information in event detection through supervised attention approaches. Nguyen et al. [16] and Sha et al. [17] suggested exploiting syntactic information to detect events.

[2]https://catalog.ldc.upenn.edu/LDC2006T06

The current methods require annotated triggers. Training data annotation is expensive, which limits the applications of these methods. This task is performed without the use of ETs in order to reduce manual effort.

Deep neural networks have been proven to be effective in several domains, such as natural language processing, computer vision, and audio analysis [18]. However, the existing successes have mainly established on data with an underlying grid-like structure or Euclidean space. Graphs offer a natural way to generalize the grid structure data and contain rich underlying values. Consequently, considerable researches are using deep learning approaches to analyze graph data [19–21]. Motivated by the success of CNN in many artificial intelligence applications, various methods attempt to re-define the operation of convolution for graphic data. These approaches are classified as GCN, which has gained great attention after the seminal work of [22]. GCN is assigned into two streams, spatial- and spectral-based methods. Spatial-based methods define convolutions based on the spatial associations of graph nodes. Micheli [23] first addressed node mutual dependency by the ideas of message passing from RecGNNs. Recently, many spatial-based researchers have proposed a variety of spatial-based GCN (e.g., [24–26]). Bruna et al. [22] first introduced convolution from the spectral domain and pointed out that the structure of the graph can be exploited using the graph Laplacian matrix. Next, extensions on spectral-based GCN have been increasing. Defferrard et al. [27] extended GCN with fast localized convolutions. Kipf et al. [28] used a localized first-order approximation of spectral graph convolutions. Levie et al. [29] suggested Cayley polynomials for computing spectral filters on graphs that concentrate in frequency bands of interest.

We leveraged GCN to establish event label correlations. Event labels are considered as graph nodes. The most related work to our approach is Chen et al. [30], which used GCN to propagate information among image labels based on the fixed correlation matrix. Differently, our work investigates to automatically learn the adjacency matrix apart from the fixed correlation Matrix.

3 Our Model

We present our model's framework. As mentioned in Section 1, we use a mixture of two neural network models and convert the task into a multi-label classification problem. Fig. 2 shows the model's structure. It consists of three main modules: type-aware attention neural networks, GCN and output classifier.

3.1 Type-aware Attention Neural Networks

The purpose of the type-aware attention neural networks is to learn feature vector representation of input sentence. As illustrated in Fig. 2, the part surrounded by the green dash line is this module, which is composed of the following neural components.

Event Type Embeddings As shown in Fig. 2, an event type is transformed into 2 embedding vectors: t_1 and t_2. The first (light green) is specific for type-aware attention, which aims at capturing the local information (hidden trigger word). The latter (light grey) is aimed at capturing global information (see Section 3.4 for details). The counts of dimensions in each event type embedding should be equivalent to the size of weighted average sentence representation, which is denoted by d_{evt}. The event type embeddings are parameters to be learned by the network. Both of them are randomly initialized.

The event type embeddings are shared by all the three modules in our model. In the output module, each event embedding is the parameter of the binary classifier as well. The GCN module captures the class inter-dependencies through the graph convolution operation on the event type embeddings.

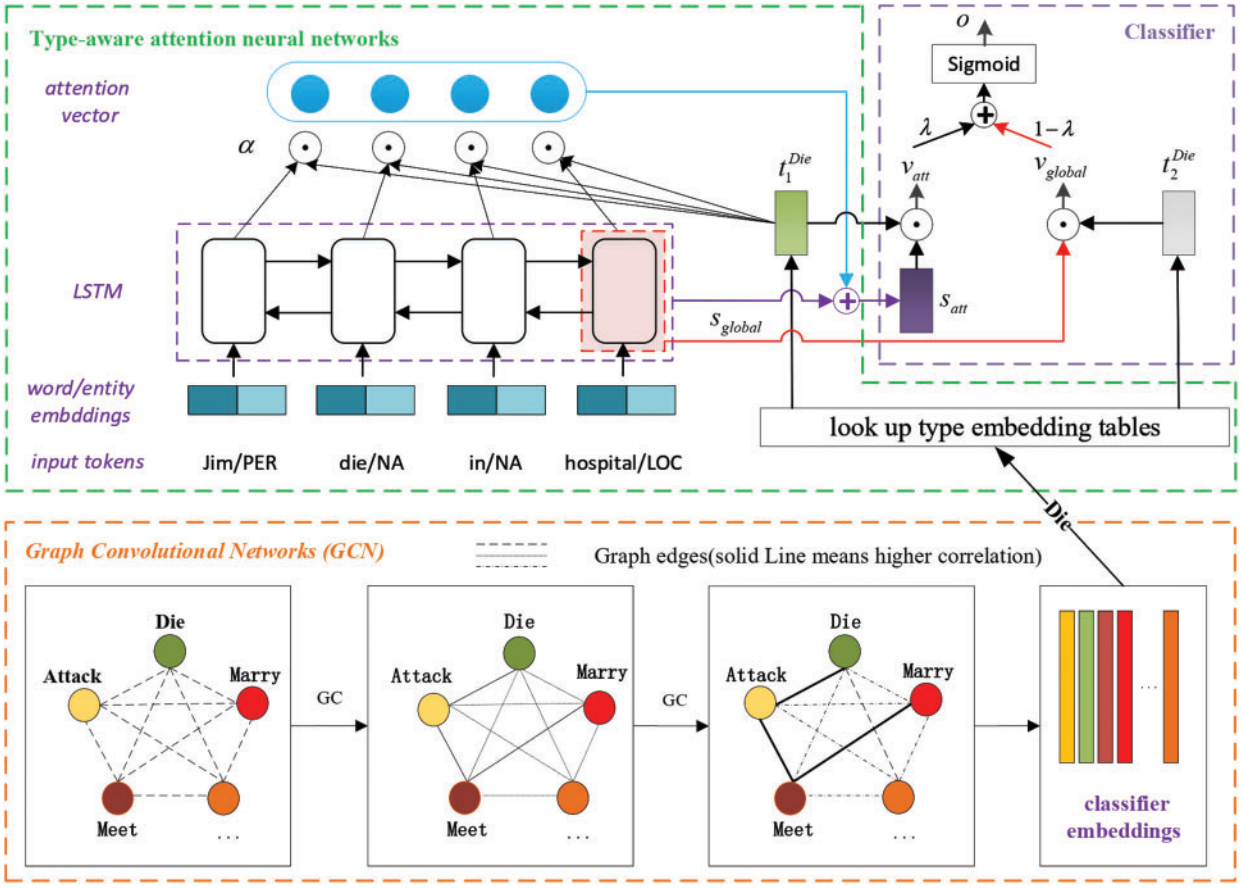

Figure 2: The framework of our proposed TA-GCN (better viewed in color). This model includes type-aware attention neural networks, GCN and output modules

LSTM Encoding Layer In this work, we used Bidirectional LSTM (BiLSTM) as a sentence encoder which reads input sequences in both left-to-right as well as reverse orders. Before feeding into the encoder, each word is first converted into a vector by looking up a pre-trained word embedding (e.g., word2vec). Then, bidirectional information for every word is combined by concatenating backward and forward output. The $i-$th word of the input sentence is encoded as:

$$\bar{h}_i = (\overrightarrow{h}_i + \overleftarrow{h}_i)/2 \tag{1}$$

Thus, given a sentence $s = \{s_1, s_2, ..., s_n\}$ as a sequence of tokens, then, the LSTM encoding layer is responsible for mapping every token to continuous embedding representations as $H = \{\bar{h}_1, \bar{h}_2, ..., \bar{h}_n\}$.

3.2 Event Type-Aware Attention

There is normally a specific set of words that trigger each event type, commonly referred to as event trigger words. For instance, the ***Die*** events are majorly triggered by "passed away", "die", "gone", etc. To accomplish this task, event trigger words are of vital importance. But since because annotations are not present in our task, this evidence is hidden. For hidden trigger modeling, we introduced attention mechanisms.

As shown in Fig. 2, attention vector α is evaluated based on target event type embedding t_1 and the encoding outputs of tokens $\bar{h}$ yielded by LSTM. Precisely, attention score for $k-$th token in a sentence is determined as:

$$\alpha^k = \frac{\exp\left(\bar{\mathbf{h}}_k \cdot \mathbf{t}_1^T\right)}{\sum_i \exp\left(\bar{\mathbf{h}}_i \cdot \mathbf{t}_1^T\right)} \tag{2}$$

The attention weight α indicates the relative contribution of each token to the sentence representation. We comprehensively consider event type and the LSTM encoding output to calculate α. Essentially, the event type-aware attention mechanism is looking for the features most relevant to the given event type. As mentioned above, the trigger words are the most salient features of the task. In this model, it is expected that the trigger words of the target event type obtain higher contibution scores relative to other words. The following equation is used to compute S_{att} with respect to the above sentence:

$$\mathbf{s}_{\text{att}} = \alpha^T \mathbf{H} \tag{3}$$

where $\alpha = [\alpha_1, \ldots, \alpha_n]$ is the attention vector, $\mathbf{H} = \left[\bar{\mathbf{h}}_1, \bar{\mathbf{h}}_2, \ldots, \bar{\mathbf{h}}_n\right]$ is the matrix, $\bar{\mathbf{h}}_k$ is LSTM's output for $k-$th token, and $\mathbf{S}_{att}$ represents the given sentence.

3.3 Graph Convolutional Networks

A graph mining technique known as GCN can be used to derive structural features from graphs. GCN was used to capture associations of multiple binary classifiers. As illustrated in Fig. 2, the part surrounded by the orange dash line is this module.

Event Label Graph GCN refers to an extension of the convolutional neural network that can be used to encode graphs. Basically, it performs convolution filtration on the graph and propagates information between nodes so that node representations are updated. Ordinarily, it is an ordered pair $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, whereby $\mathcal{V}$ denotes a set of vertices while $\mathcal{E}$ denotes set of edges. Mathematically, a graph can be represented with n nodes by adjacency matrix $\mathbf{A} \in \mathbb{R}^{n \times n}$, whereby $\mathbf{A}_{ij} = 1$ if there is an edge between nodes i and j , otherwise 0. If the nodes $\mathcal{V}$ are denoted as d dimensional vectors $\in \mathbb{R}^{n \times d}$, then, the GCN layer on a graph can be written as a non-linear function $f(\mathbf{V}, \mathbf{A})$. Give the stacking of several GCN layers and after exploitation of the convolutional operation suggested in [28], the GCN can be denoted as:

$$\mathbf{V}^{l+1} = f(\mathbf{V}^l, \mathbf{A}) = \delta(\mathbf{A}\mathbf{V}^l\mathbf{W}^l) \tag{4}$$

whereby $\delta(\cdot)$ is the activation function that is selected as LeakyReLU in this study, the superscript l denotes the layer number. $\mathbf{W}^l \in \mathbb{R}^{d \times d}$ denotes learnable parameters of the convolutional filter.

Recognizing multiple events of a sentence is a fundamental yet practical in event detection without triggers, since real-world sentences always contain rich and diverse event information. As mentioned in Section 3, effectively capturing event label correlations and evaluating these correlations to enhance performance of binary classifiers is very important to address the multi-label problem. We used GCN for modeling event label correlations. Specifically, as introduced in Section 3.1, each event label is transformed into a vector which is further considered as the graph node. The correlations among the events are stored in the edges of the graph.

From Eq. (4), we can observe that the nodes of the graph in the $l-$th GCN layer are aggregated by neighbors to form the nodes of the $(l+1)-$th layer. For instance, if label 5 has two adjacent labels 3 and 4, Eq. (4) can written as:

$$V_5^{l+1} = \delta\left(a_{35}v_3^l\mathbf{W}^l + a_{45}V_4^l\mathbf{W}^l + a_{55}V_5^l\mathbf{W}^l\right) \tag{5}$$

where a_{ij} denotes the element at column j and row i of adjacent matrix A, V_i^l is the label representation corresponding to $i-$th node of $l-$th layer. Thus, GCN uses common convolution weights to combine all adjacent label information, which is then processed via a single activation function to produce updated node features. Through this approach, adjacent labels in the graph affect each other, and correlations among labels are established after several layers of convolution operations.

Adjacency Matrix The adjacency matrix has a vital role in the model. It models the dependence between labels. The construction of the adjacency matrix is the key issue of the model. We use two different methods to initialize the adjacency matrix.

1. Correlation Matrix: the correlation matrix is constructed similarly to [30] by counting the co-occurrence of labels in the training corpus. If two events occur in the same sentence, they will be considered as the co-occurrence of their corresponding labels. To address the label imbalance problem, we normalize the correlation Matrix as follows:

   $$\mathbf{M}_{\text{Nom}} = \mathbf{M}/\mathbf{F} \tag{6}$$

   where $\mathbf{M}$ is the co-occurrence matrix and $\mathbf{F}$ is the frequency vector of individual label. We simply add a self-loop to each node by adding the identity matrix to the adjacency matrix. We fix the matrix during the training process, thus it completely relies on the prior co-occurrence information in the training data as the label dependence.
2. Random Initialization: We randomly initialize the weight of the adjacency matrix. The weight is considered as the model parameter and updated during the training process. A sentence may contain multiple events. When applying the propagation rule, the error will be propagated back to the corresponding label, which will further affect the update of the adjacency matrix. Thus, label dependence is learned from training instances.

These two adjacency matrices construction methods have their own characteristics. On the one hand, the correlation matrix is sparse, in the sense that most of the possible edges between pairs of vertices do not exist. It is relatively simple and has fewer parameters. On the other hand, the random initialization matrix is not a sparse matrix. It may carry not only positive label correlation, but also other factors. For example, the model may learn that the two labels are mutually exclusive.

3.4 Classifier Layer

As shown in Fig. 2, final output O is connected to 2 components: v_{att} and v_{global}. v_{att} is evaluated by dot product of $\mathbf{S}_{\text{att}}$ and $\mathbf{t}_1$, which is aimed at capturing local features (particularly, features regarding hidden trigger words). The last LSTM layer output, $\bar{\mathbf{h}}_n$, encodes global information of the whole sentence, thus $v_{\text{global}} = \bar{\mathbf{h}}_n \cdot \mathbf{t}_2^T$ captures the global features of a sentence. Lastly, O is the weighted sum of v_{att} and v_{global}:

$$o = \sigma\left(\lambda \cdot v_{att} + (1-\lambda) \cdot v_{global}\right) \tag{7}$$

where σ is the Sigmoid function, $\lambda \in [0, 1]$ is the hyper-parameter for trade-off between v_{att} and v_{global}.

4 Results and Discussions

4.1 Dataset and Evaluation Indices

The experiments were conducted using the ACE 2005 dataset. Based on previous studies [3,10,12], we randomly selected 30 articles from dissimilar genres as the development set, and performed a blind

test on a distinct set of 40 ACE 2005 newswire documents. The remaining 529 articles were included in the training set.

Our study was to detect events minus triggers. Thus, we removed trigger annotations from corpus. Based on annotations to the ACE 2005 corpus, Stanford CoreNLP Toolkit was used for splitting documents into sentences and assigning each sentence a set of labels. If a sentence lacks any event, it is assigned a special label, NA; if it has several events of the same type (<3% in ACE corpus), only one label is kept for each type.

Precision (*P*), F1-measure (*F1*) and recall (*R*) are used to evaluate the results.

4.2 Overall Performances

We illustrate the findings of the suggested approach (see Tab. 1). The results list in the first group from the baseline systems we implement based on the encoding sentence strategy. Three models are used for comparison: CNN, LST, $LSTM_{\text{avg}}$.

Table 1: Experimental results on ACE 2005 corpus. + means triggers are required, $*$ means training with external data

Methods	P (%)	R (%)	F1 (%)
CNN	76.6	52.9	62.6
$LSTM_{avg}$	68.1	49.2	57.1
$LSTM_{last}$	69.8	52.2	59.7
Nguyen's CNN+	71.8	66.4	69.0
PSL+	75.3	64.4	69.4
DMCNN+	75.6	63.6	69.1
DS-DMCNN+∗	75.7	66.0	70.5
TA-GCN	**77.2**	**68.1**	**72.4**

Methods in the second group are state-of-the-art ED systems on ACE 2005 dataset.

- *CNN* uses a CNN model for encoding sentences.
- $LSTM_{last}$ employs LSTM model, and uses the hidden state of the last token to represent a particular sentence.
- $LSTM_{avg}$ also uses the LSTM model, but uses the mean of all hidden states as a representation of a particular sentence.
- Nguyen's CNN: the CNN model proposed by [7].
- DMCNN: the dynamic multi-pooling CNN model proposed by [13].
- PSL: the soft probabilistic soft logic model proposed by [8].
- DS-DMCNN: the DMCNN model augmented with automatically labeled data, proposed by [10].

TA-GCN is our proposed approach which combing type-aware attention and graph convolutional networks.

From the Tab. 1, we make the following observations:

- Both precision and recall of *TA-GCN* are higher than all baseline systems, showing the effectiveness of the proposed attention mechanism and GCN.

– All state-of-the-art ED systems require annotated triggers. Without trigger annotations, our approach achieves higher performances as well, even better than *DS-DMCNN* which requires both annotated triggers and external data.
– The results obtained by $LSTM_{\text{avg}}$ and $LSTM_{\text{last}}$ are both worse than *CNN*. Compared with LSTM, CNN captures more local features. This result indicates that the local features of the task are very important. It guides us to further study the effect of the type-aware attention mechanism for this task.

4.3 Effects of Type-Aware Attention

Based on the results of the overall performances, CNN has achieved better results in the first group. We have confirmed the effectiveness of local information to some extent. We propose a type-aware attention mechanism mainly to capture local information (trigger words). To this end, we designed two sets of experiments to prove the effectiveness of this strategy.

First, we perform an ablation study to understand the impact of the type-aware attention mechanism. We compare the effects of *CNN* and $LSTM_{\text{att}}$ in Tab. 2. The results show that although the type-aware attention based method is inferior to CNN in precision, the recall and F1 value have a significant improvement. Since we have proved that *CNN* is better than $LSTM_{\text{avg}}$ and $LSTM_{\text{last}}$ in Section 4.2, this experiment shows that the proposed type-aware attention method has a certain effect.

Table 2: Results of systems using type-aware attention mechanism

Methods	P (%)	R (%)	F1 (%)
CNN	76.6	52.9	62.6
$LSTM_{\text{att}}$	68.3	**64.5**	**66.3**

Second, we evaluate the quality of the learned features in a visual way. We explore the effectiveness of type-aware attention mechanism from the perspective of feature separability. Traditional machine learning models convert each sentence into a fixed feature vector. In contrast, each sentence is transformed into multiple feature vectors based on the given event type label through the type-aware attention mechanism. To measure the feature separability, the sentences with only one event are simply dropped for clarity. For each sentence, we use the golden event label to get the vectors by the attentive weighted summation. We use t-distributed Stochastic Neighbor Embedding (t-SNE) with two components and principal component analysis (PCA) initialization to visualize the sentence vectors that contain multiple events in Fig. 3. We can observe that events that often co-occur in a sentence are also closer in the figure, such as Marry *vs.* Divorce, End-Position *vs.* Start-Position. Although there are few events mixed together, the learned features have good separation between classes and good clustering within classes for majority events. Considering that for the same sentence, we get different vectors for different golden event labels. This experiment further confirms that the type-aware attention mechanism is very effective.

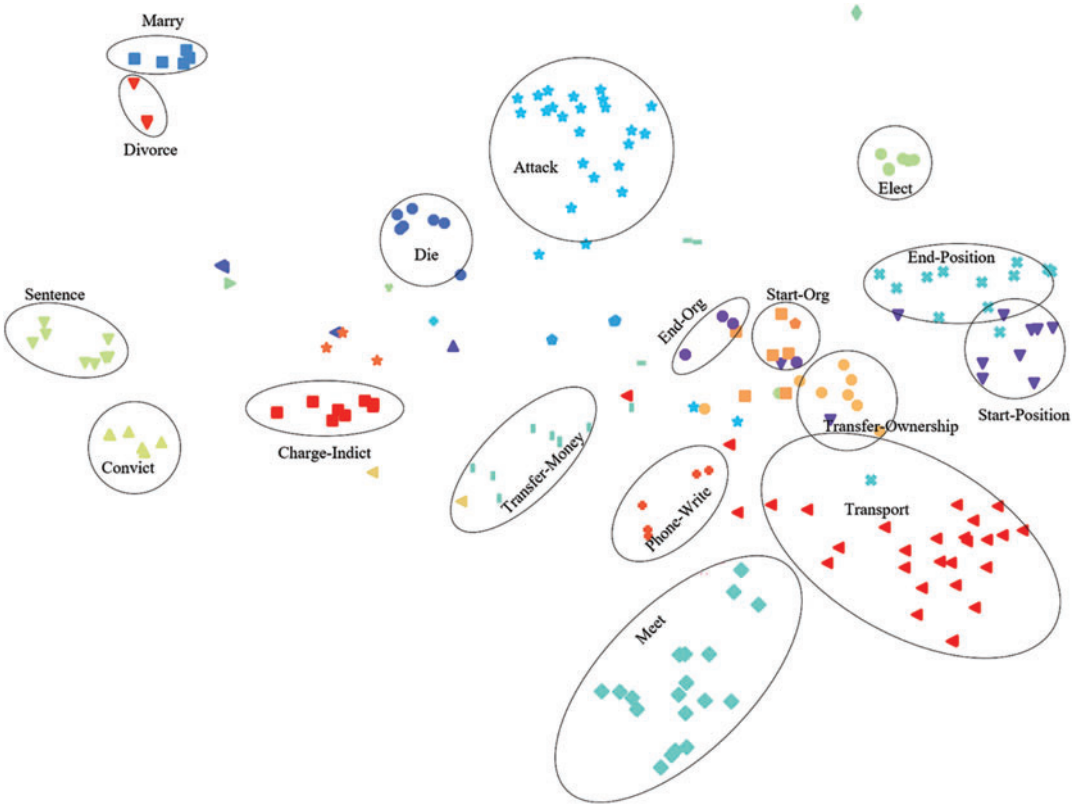

Figure 3: t-SNE on the sentence vectors of the test dataset (better viewed in color)

4.4 Effects of GCN

GCN is exploited to model the correlation among the binary classifiers. In this section, we perform ablation studies and mainly evaluate the effectiveness of GCN. We compare the experimental results of the model without/with GCN and using different adjacency matrices in Tab. 3. We can observe that no matter which adjacency matrix is used, the effect of *TA-GCN* exceeds that of $LSTM_{\text{att}}$. This result can indicate that GCN is very effective for this task. It achieves the purpose of capturing the correlation among classifiers. Another interesting observation is that, *TA-GCN**Random* gets better results than *TA-GCN**Correlation*. We suppose that the improvement is benefited from that *TA-GCN**Correlation* can not only learn the relation between event labels from the training data, but also can model the noise in the data.

Table 3: Results of systems without/with GCN and different adjacency matrix, where $*$*Correlation* and $*$*Random* indicate to respectively use the correlation matrix and random initialization as the adjacency matrix

Methods	P (%)	R (%)	F1 (%)
$LSTM_{att}$	68.3	64.5	66.3

(Continued)

Table 3: Continued

Methods	P (%)	R (%)	F1 (%)
TA-GCN\Correlation	76.4	65.1	70.3
TA-GCN\Random	**77.2**	**68.1**	**72.4**

4.5 Case Study

Our model illustrates several examples of the attention vectors α as shown in Fig. 4. In case one, "died" is the most important keyword for the Die event. As a result of a large attention score, our model was able to capture this characteristic. According to our model, in case two, the "fired" denotes the Attack event, which was given an attention score of high. Die and Attack events are both triggered by the words "died" and "fired" respectively. Thus, even though annotated triggers are not available, our model exploited trigger information for this task. In addition, our method could also model dependencies among events, which is useful for this task [1,5]. For instance, Attack events repeatedly co-occur with Die events. In Cases 1 and 2 (Fig. 4), our method models such information by paying attention to the words "died" and "fired". A negative sample is case 3, which lacks key clues. Approximately equal attention scores were assigned to each token by our model.

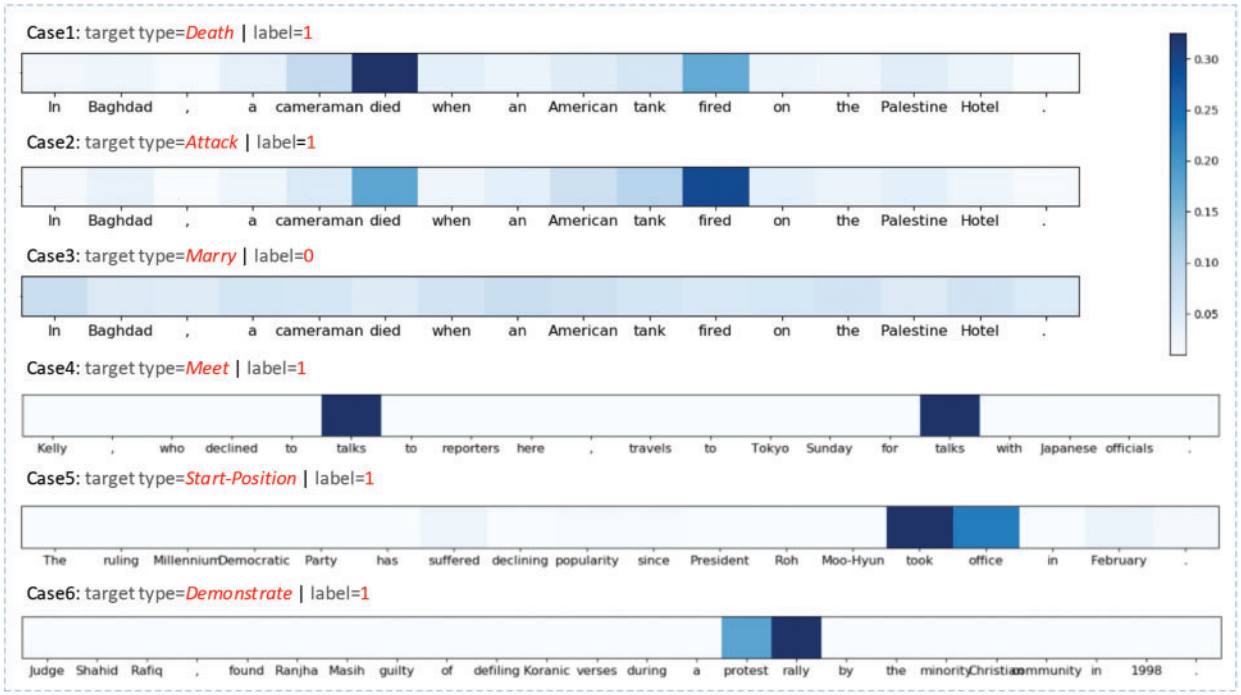

Figure 4: A visual representation of the attention weight vector α learned by our model

In the data preprocessing stage, we only keep one label when a sentence has multiple events of the same type. Such instances are illustrated in Case 4 (Fig. 4). We can observe that our method can attentively find two trigger words (both are "talks") for two "Meet" events in one sentence. In Case 5, "took" and "office" together trigger the Start-Position event. This example shows that our

method has the ability to handle multi-word triggers. In Case 6, the ground truth trigger word for Demonstrate event is "rally" in ACE 2005. However, our model finds that "protest" is the trigger word as well. From the ACE 2005 annotation guidelines, a Demonstrate event occurs when a large number of people aggregate in a public area to demand or protest an official action. These two words should henceforth be combined to trigger this event. As discussed in Section 1, the accuracy of human annotated trigger words are relatively low and pose obstacles for event extraction. This case further reflects the superiority of event detection without triggers.

5 Conclusions and Future Work

We propose a new method for ED without triggers by combining type-aware attention and graph convolutional networks. Our contribution is mainly in two aspects. We propose type-aware attention networks to encode the input sentence. Through the type-aware attention strategy, the trigger information has been implicitly captured to address the trigger absence problem. Besides, we consider event detection without triggers as a multi-label problem. The GCN is exploited to model the event correlations, rather than treating the events in isolation. The type-aware attention networks and GCN work in a mutual reinforcing way. We increase the ED without triggers at an F1 score of 72.4%, which achieves higher performances relative to state-of-the-arts that use both annotated triggers and external data. The experimental results show that we not only accurately get the trigger word through type-aware attention, but also make up for the defects of manually annotation.

Acknowledgement: This work was supported by Youth Cultivation Project of Research Institute of Languages and Cultures, Hunan Normal University (Grant No. 2020QNP05), the Hunan Provincial Natural Science Foundation of China (Grant No. 2020JJ4624), the National Social Science Fund of China (Grant No. 20&ZD047), the Scientific Research Fund of Hunan Provincial Education Department (Grant No.19A020), the National University of Defense Technology Research Project ZK20-46 and the Young Elite Scientists Sponsorship Program 2021-JCJQ-QT-050.

Funding Statement: This work was supported by the Hunan Provincial Natural Science Foundation of China (Grant No. 2020JJ4624), the National Social Science Fund of China (Grant No. 20&ZD047), the Scientific Research Fund of Hunan Provincial Education Department (Grant No.19A020), the National University of Defense Technology Research Project ZK20-46 and the Young Elite Scientists Sponsorship Program 2021-JCJQ-QT-050.

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

[1] S. Khatoon, M. A. Alshamari, A. Asif, M. M. Hasan, S. Abdou *et al.,* "Development of social media analytics system for emergency event detection and crisis management," *Computers, Materials & Continua*, vol. 68, no. 3, pp. 3079–3100, 2021.

[2] G. Anitha and S. B. Priya, "Vision based real time monitoring system for elderly fall event detection using deep learning," *Computer Systems Science and Engineering*, vol. 42, no. 1, pp. 87–103, 2022.

[3] H. Ji and R. Grishman, "Refining event extraction through cross-document inference," in *Proc. ACL-08: HLT*, Columbus, OH, USA, pp. 254–262, 2008.

[4] S. Liao and R. Grishman, "Using document level cross-event inference to improve event extraction," in *Proc. ACL-2010*, Uppsala, Sweden, pp. 789–797, 2010.

[5] Y. Hong, J. Zhang, B. Ma, J. Yao, G. Zhou *et al.,* "Using cross-entity inference to improve event extraction," in *Proc. ACL-HLT-11*, pp. 1127–1136, 2011.
[6] Q. Li, H. Ji and L. Huang, "Joint event extraction via structured prediction with global features," in *Proc. ACL-13*, Sofia, Bulgaria, pp. 73–82, 2013.
[7] H. T. Nguyen and R. Grishman, "Event detection and domain adaptation with convolutional neural networks," in *Proc. ACL-IJCNLP-15*, Beijing, China, pp. 365–371, 2015.
[8] S. Liu, K. Liu, S. He and J. Zhao, "A probabilistic soft logic based approach to exploiting latent and global information in event classification," in *Proc. AAAI-16*, Phoenix, AZ, USA, pp. 2993–2999, 2016.
[9] S. Liu, Y. Chen, S. He, K. Liu and J. Zhao, "Leveraging framenet to improve automatic event detection," in *Proc. in Proc. ACL-16*, Berlin, Germany, pp. 2134–2143, 2016.
[10] Y. Chen, S. Liu, X. Zhang, K. Liu and J. Zhao, "Automatically labeled data generation for large scale event extraction," in *Proc. in Proc. ACL-17*, Vancouver, Canada, pp. 409–419, 2017.
[11] N. Ding, Z. Li, Z. Liu, H. Zheng and Z. Lin, "Event detection with trigger-aware lattice neural network," in *Proc. EMNLP-IJCNLP*, Hong Kong, China, pp. 347–356, 2019.
[12] S. Liu, Y. Li, F. Zhang, T. Yang and X. Zhou, "Event detection without triggers," in *Proc. NAACL-HLT-19*, Minneapolis, Minnesota, pp. 735–744, 2019.
[13] Y. Chen, L. Xu, K. Liu, D. Zeng, and J. Zhao. "Event extraction via dynamic multi-pooling convolutional neural networks," in *Proc. ACL-IJCNLP-15*, Beijing, China, pp. 167–176, 2015.
[14] H. T. Nguyen and R. Grishman, "Modeling skip-grams for event detection with convolutional neural networks," in *Proc. EMNLP*, Austin, TX, USA, pp. 886–891, 2016.
[15] S. Liu, Y. Chen, K. Liu and J. Zhao, "Exploiting argument information to improve event detection via supervised attention mechanisms," in *Proc. ACL-17*, Vancouver, Canada, pp. 1789–1798, 2017.
[16] T. Nguyen and R. Grishman, "Graph convolutional networks with argument-aware pooling for event detection," in *Proc. AAAI-18*, New Orleans, LA, USA, pp. 5900–5907, 2018.
[17] L. Sha, F. Qian, B. Chang and Z. Sui, "Jointly extracting event triggers and arguments by dependency-bridge rnn and tensor-based argument interaction," in *Proc. AAAI-18*, New Orleans, LA, USA, 2018.
[18] D. Zeng, J. Tian, R. Peng, J. Dai, H. Gao *et al.,* "Joint event extraction based on global event-type guidance and attention enhancement," *Computers, Materials & Continua*, vol. 68, no. 3, pp. 4161–4173, 2021.
[19] D. Zhang, J. Hu, F. Li, X. Ding, A. K. Sangaiah *et al.,* "Small object detection via precise regionbased fully convolutional networks," *Computers, Materials and Continua*, vol. 69, no. 2, pp. 1503–1517, 2021.
[20] J. Wang, Y. Wu, S. He, P. K. Sharma, X. Yu *et al.,* "Lightweight single image super-resolution convolution neural network in portable device," *KSII Transactions on Internet and Information Systems (TIIS)*, vol. 15, no. 11, pp. 4065–4083, 2021.
[21] W. Wang, H. Liu, J. Li, H. Nie and X. Wang, "Using CFW-net deep learning models for X-ray images to detect COVID-19 patients," *International Journal of Computational Intelligence Systems*, vol. 14, no. 1, pp. 199–207, 2021.
[22] J. Bruna, W. Zaremba, A. Szlam and Y. Lecun, "Spectral networks and locally connected networks on graphs," in *Proc. ICLR-14*, Banff, Canada, 2014.
[23] A. Micheli, "Neural network for graphs: A contextual constructive approach," *IEEE Transactions on Neural Networks*, vol. 20, no. 3, pp. 498–511, 2009.
[24] M. Niepert, M. Ahmed and K. Kutzkov, "Learning convolutional neural networks for graphs," in *Proc. PMLR-16*, New York, NY, USA, pp. 2014–2023, 2016.
[25] J. Atwood and D. Towsley, "Diffusion-convolutional neural networks," in *Proc. NIPS-16*, Barcelona, Spain, pp. 1993–2001, 2016.
[26] A. Jothi and P. L. K. Priyadarsini, "Optimal path planning for intelligent UAVs using graph convolution networks," *Intelligent Automation & Soft Computing*, vol. 31, no. 3, pp. 1577–1591, 2022.
[27] M. Defferrard, X. Bresson and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," in *Proc. NIPS-16,* Barcelona, Spain, pp. 3844–3852, 2016.
[28] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *Proc. ICLR-17*, Toulon, France, 2017.

[29] R. Levie, F. Monti, X. Bresson and M. M. Bronstein, "Cayleynets: Graph convolutional neural networks with complex rational spectral filters," *IEEE Transactions on Signal Processing*, vol. 67, no. 1, pp. 97–109, 2019.

[30] Z. M. Chen, X. S. Wei, P. Wang and Y. Guo, "Multi-label image recognition with graph convolutional networks," in *Proc. CVPR-19*, Honolulu, HI, USA, pp. 5177–5186, 2019.