Jointly Multiple Events Extraction via Attention-based Graph Information Aggregation

Xiao Liu[†] and **Zhunchen Luo**[‡] and **Heyan Huang**[†*]
[†]School of Computer Science and Technology, Beijing Institute of Technology
100081 Beijing, China
{xiaoliu,hhy63}@bit.edu.cn
[‡]Information Research Center of Military Science, PLA Academy of Military Science
100142 Beijing, China
zhunchenluo@gmail.com

Abstract

Event extraction is of practical utility in natural language processing. In the real world, it is a common phenomenon that multiple events existing in the same sentence, where extracting them are more difficult than extracting a single event. Previous works on modeling the associations between events by sequential modeling methods suffer a lot from the low efficiency in capturing very long-range dependencies. In this paper, we propose a novel **J**ointly **M**ultiple **E**vents **E**xtraction (JMEE) framework to jointly extract multiple event triggers and arguments by introducing syntactic shortcut arcs to enhance information flow and attention-based graph convolution networks to model graph information. The experiment results demonstrate that our proposed framework achieves competitive results compared with state-of-the-art methods.

1 Introduction

Extracting events from natural language text is an essential yet challenging task for natural language understanding. When given a document, event extraction systems need to recognize event triggers with their specific types and their corresponding arguments with the roles. Technically speaking, as defined by the ACE 2005 dataset[1], a benchmark for event extraction (Grishman et al., 2005), the event extraction task can be divided into two subtasks, i.e., event detection (identifying and classifying event triggers) and argument extraction (identifying arguments of event triggers and labeling their roles).

In event extraction, it is a common phenomenon that multiple events exist in the same sentence. Extracting the correct multiple events from those sentences is much more difficult than in the one-event-one-sentence cases because those various types of events are often associated with each other. For example, in the sentence "He ***left*** the company, and planned to ***go*** home directly.", the trigger word ***left*** may trigger a *Transport* (a person left a place) event or an *End-Position* (a person retired from a company) event. However, if we take the following event triggered by ***go*** into consideration, we are more confident to judge it as a *Transport* event rather than an *End-Position* event. This phenomenon is quite common in our real world, as *Injure* and *Die* events are more likely to co-occur with *Attack* events than others, whereas *Marry* and *Born* events are less likely to co-occur with *Attack* events. As we investigated in ACE 2005 dataset, there are around 26.2% (1042/3978) sentences belong to this category.

Significant efforts have been dedicated to solving this problem. Most of them exploiting various features (Liu et al., 2016b; Yang and Mitchell, 2016; Li et al., 2013; Keith et al., 2017; Liu et al., 2016a; Li et al., 2015), introducing memory vectors and matrices (Nguyen et al., 2016), introducing more transition arcs (Sha et al., 2018), keeping more contextual information (Chen et al., 2015) into sentence-level sequential modeling methods like RNNs and CRFs. Some also seek features in document-level methods (Liao and Grishman, 2010; Ji and Grishman, 2008). However, sentence-level sequential modeling methods suffer a lot from the low efficiency in capturing very long-range dependencies while the feature-based methods require extensive human engineering, which also largely affects model performance. Besides, these methods do not adequately model the associations between events.

An intuitive way to alleviate this phenomenon is to introduce shortcut arcs represented by linguistic resources like dependency parsing trees to

[*]Corresponding author.

[1]https://catalog.ldc.upenn.edu/ldc2006t06

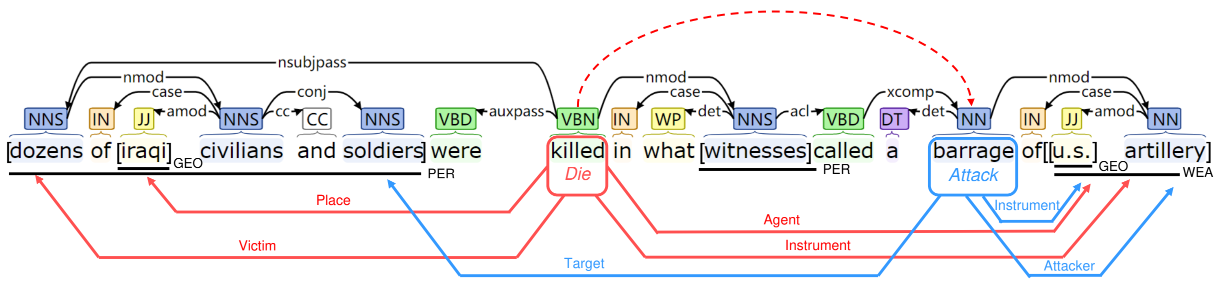

Figure 1: An example of dependency parsing result produced by Stanford CoreNLP. There are two events in the sentence: a *Die* event triggered by the word *killed* with four arguments in red and an *Attack* event triggered by the word *barrage* with three arguments in blue. The red dotted arc is the shortcut path consisting of three directed arcs from trigger *killed* to another trigger *barrage*.

drain the information flow from a point to its target through fewer transitions. Comparing to sequential order, modeling with these arcs often successfully reduce the needed hops from one event trigger to another in the same sentences. In Figure 1, for example, there are two events: a *Die* event triggered by the word *killed* with four arguments in red and an *Attack* event triggered by the word *barrage* with three arguments in blue. We need six hops from *killed* to *barrage* according to sequential order, but only three hops according to the arcs in dependency parsing tree (along the *nmod*-arc from *killed* to *witnesses*, along the *acl*-arc from *witnesses* to *called*, and along the *xcomp*-arc from *called* to *barrage*). These three arcs consist of a shortcut path[2], draining the dependency syntactic information flow from *killed* to *barrage* with fewer hops[3].

In this paper, we propose a novel **J**ointly **M**ultiple **E**vents **E**xtraction (JMEE) framework by introducing syntactic shortcut arcs to enhance information flow and attention-based graphic convolution networks to model the graph information. To implement modeling with the shortcut arcs, we adopt the graph convolutional networks (GCNs) (Kipf and Welling, 2016; Marcheggiani and Titov, 2017; Nguyen and Grishman, 2018) to learn syntactic contextual representations of each node by the representative vectors of its immediate neighbors in the graph. And then we utilize the syntactic contextual representations to extract triggers and arguments jointly by a self-attention mechanism to aggregate information especially keeping the associations between multiple events.

[2]In a shortcut path which consists of existing arcs, some arcs may reverse their directions.

[3]The length of the longest path in a tree is always no more than the sequential length consisting of the same number of nodes, which means even in the worst cases, the shortcut path will not perform worse than sequential modeling.

We extensively evaluate the proposed JMEE framework with the widely-used ACE 2005 dataset to demonstrate its benefits in the experiments especially in capturing the associations between events. To summary, our contribution in this work is as follows:

- We propose a novel joint event extraction framework JMEE based on syntactic structures which enhance information flow and alleviate the phenomenon where multiple events are in the same sentence.
- We propose a self-attention mechanism to aggregate information especially keeping the associations between multiple events and prove it is useful in event extraction.
- We achieve the state-of-the-art performance on the widely used datasets for event extraction using the proposed model with GCNs and self-attention mechanism.

2 Approach

Generally, event extraction can be cast as a multi-class classification problem deciding whether each word in the sentence forms a part of event trigger candidate and whether each entity in the sentence plays a particular role in the event triggered by the candidate triggers. There are two main approaches to event extraction: (i) the joint approach that extracts event triggers and arguments simultaneously as a structured prediction problem, and (ii) the pipelined approach that first performs trigger prediction and then identifies arguments in separate stages. We follow the joint approach that can effectively avoid the propagated errors in the pipeline.

Additionally, we extract events in sentence-level mainly for three reasons. Firstly, in our in-

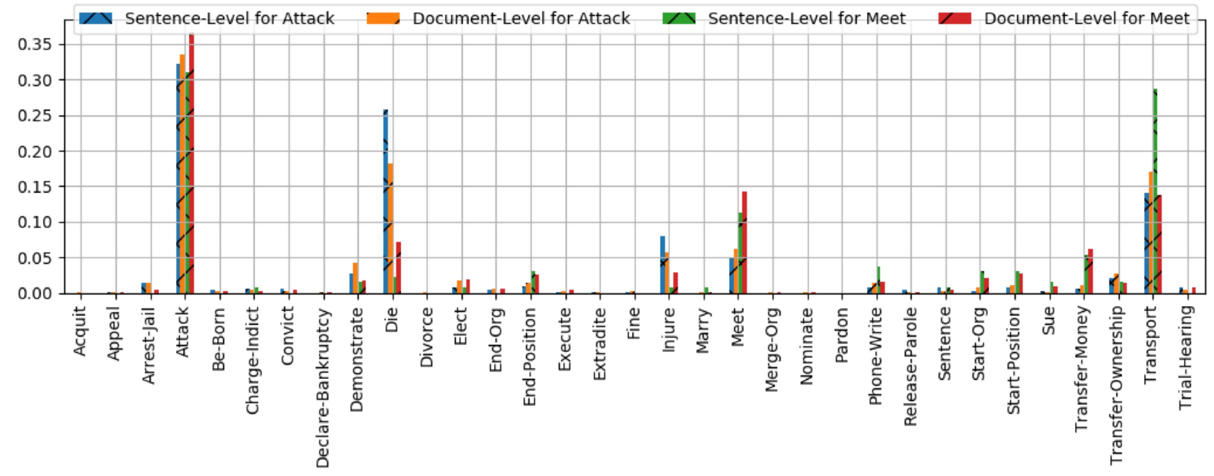

Figure 2: The conditional probability distributions of all the 33 event types in sentences and documents where an *Attack* event or a *Meet* event appears. The label on the top shows which histogram the color stands.

vestigation, we find that the document-level co-occurrence distributions of 33 types of events in the ACE 2005 dataset are relatively similar to the sentence-level co-occurrence distributions. In Figure 2, for example, the blue bars and the orange bars indicate the conditional probability distribution of all the 33 event types in sentences and documents, respectively, where an *Attack* event appears. While the green bars and the red bars indicate the sentence-level and document-level conditional probability distributions respectively. As we can see from this figure, the top three types of *Attack* event in co-occurrence relationships are *Die*, *Transport* and *Injure*, while those of *Meet* event are *Attack*, *Transport*, and *Transfer-Money*. Although different types of events have different co-occurrence relationships, the conditional probability distributions in two levels of the same event type are relatively similar[4]. Secondly, there are many off-the-shelf sentence-level linguistic resources in the NLP community which can offer analytical information about the shortcut paths of some structures, like dependency parsing trees, AMR parsing graphs, and semantic role labeling structures. Last but not least, we also find that events within the same sentences have more explicit relationships with each other than events in different sentences of a document, which means the associations between two events is more accessible to capture.

Let $W = w_1, w_2, ..., w_n$ be a sentence of length n where w_i is the i-th token. Similarly, let $E = e_1, e_2, ..., e_k$ be the entity mentions in the sentence where k is the number of the entity mentions. We apply the BIO annotation schema to assign trigger label t_i to each token w_i, as there are triggers that consist of multiple tokens. If we can get trigger candidates of certain types from trigger labels in BIO annotation schema, we then need to predict the roles (if any) that each entity mention e_j plays in such events.

Our JMEE framework consists of the following four modules: (i) word representation module that represents the sentence with vectors, (ii) syntactic graph convolution network module that performs convolution operations by introducing shortcut arcs from syntactic structures, (iii) self-attention trigger classification module that captures the associations between multiple events in a sentence, and (iv) argument classification that predicts the roles each entity mention e_j plays in the event candidates of specific types, as shown in Figure 3.

2.1 Word Representation

In the word representation module, each token w_i in the sentence is transformed to a real-valued vector x_i by looking up in embedding matrices and concatenating the following vectors:

- The word embedding vector of w_i: This is obtained by looking up a pre-trained word embedding matrix Glove (Pennington et al., 2014).

- The POS-tagging label embedding vector of w_i: This is generated by looking up the randomly initialized POS-tagging label embedding table.

[4]We only focus on the top-K co-occurrence relationships because the rest are too sparse for statistic analysis.

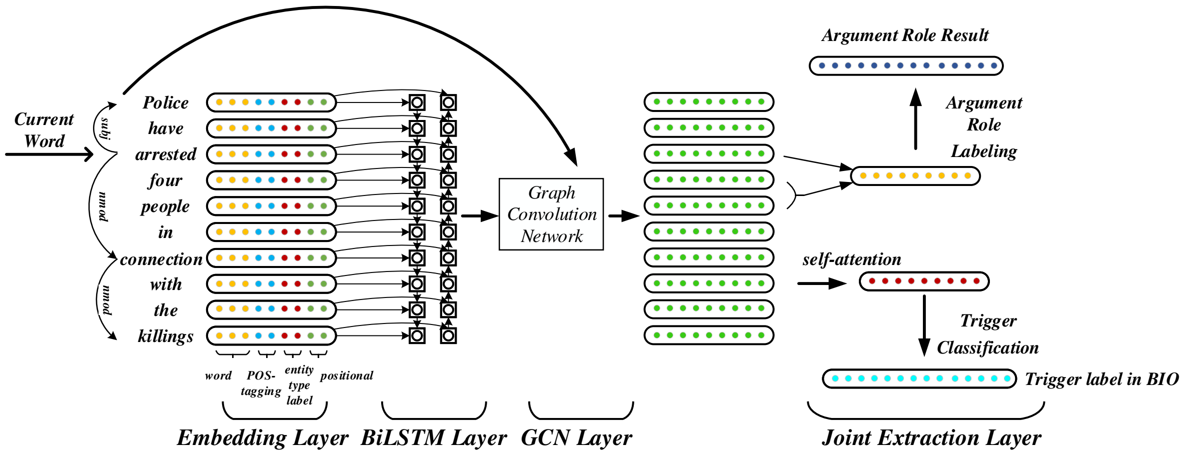

Figure 3: The architecture of our jointly multiple events extraction framework.

- The positional embedding vector of w_i: If w_c is the current word, we encode the relative distance $i - c$ from w_i to w_c as a real-valued vector by looking up the randomly initialized position embedding table (Nguyen et al., 2016; Liu et al., 2017; Nguyen and Grishman, 2018).

- The entity type label embedding vector of w_i: Similarly to the POS-tagging label embedding vector of w_i, we annotate the entity mentions in a sentence using BIO annotation schema and transform the entity type labels to real-valued vectors by looking up the embedding table. It should be noticed that we use the whole entity extent in ACE 2005 dataset which contains overlapping entity mentions and we sum all the possible entity type label embedding vectors for each token.

The transformation from the token w_i to the vector x_i essentially converts the input sentence W into a sequence of real-valued vectors $X = (x_1, x_2, ..., x_n)$, which will be feed into later modules to learn more effective representations for event extraction.

2.2 Syntactic Graph Convolution Network

Considering an undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ as the syntactic parsing tree for sentence W, where $\mathcal{V} = v_1, v_2, ..., v_n(|\mathcal{V}| = n)$ and $\mathcal{E}$ are sets of nodes and edges, respectively. In $\mathcal{V}$, each v_i is the node representing token w_i in W. Each edge $(v_i, v_j) \in \mathcal{E}$ is a directed syntactic arc from token w_i to token w_j, with the type label $K(w_i, w_j)$. Additionally, to allow information to flow against the direction, we also add reversed edge (v_j, v_i) with the type label $K'(w_i, w_j)$. Following Kipf and Welling (2016), we also add all the self-loops, i.e., (v_i, v_i) for any $v_i \in \mathcal{V}$. For example, in the dependency parsing tree shown in Figure 1, there are four arcs in the subgraph with only two nodes *"killed"* and *"witnesses"*: the dependency arc with the type label $K(\text{"killed"}, \text{"witnesses"}) = nmod$, the revresed dependency arc with the additional type label $K(\text{"witnesses"}, \text{"killed"}) = nmod'$, and the two self-loops of *"killed"* and *"witnesses"* with type label $K(\text{"killed"}, \text{"killed"}) = K(\text{"witnesses"}, \text{"witnesses"}) = loop$.

Therefore, in the k-th layer of syntactic graph convolution network module, we can calculate the graph convolution vector $h_v^{(k+1)}$ for node $v \in \mathcal{V}$ by:

$$h_v^{(k+1)} = f\Big(\sum_{u \in \mathcal{N}(v)} \big(W_{K(u,v)}^{(k)} h_u^{(k)} + b_{K(u,v)}^{(k)}\big)\Big) \quad (1)$$

where $K(u, v)$ indicates the type label of the edge (u, v); $W_{K(u,v)}^{(k)}$ and $b_{K(u,v)}^{(k)}$ are the weight matrix and the bias for the certain type label $K(u, v)$, respectively; $\mathcal{N}(v)$ is the set of neighbors of v including v (because of the self-loop); f is the activation function. Moreover, we use the output of the word representation module x_i to initialize the node representation $h_{v_i}^0$ of the first layer of GCNs.

After applying the above two changes, the number of predefined directed arc type label (let us say, N) will be doubled (to $2N + 1$). It means we will have $2N + 1$ sets of parameter pairs $W_k^{(k)}$ and $b_k^{(k)}$ for a single layer of GCN. In this work, we use Stanford Parser (Klein and Manning, 2003) to generate the arcs in dependency parsing trees for sentences as the shortcut arcs. The current representation contains approximately 50 different grammatical relations, which is too high for the parameter number of a single layer of GCN and not compatible with the existing training data scale. To reduce the parameter numbers, following Marcheggiani and Titov (2017), we modify the definition of type label $K(w_i, w_j)$ to:

$$K(w_i, w_j) = \begin{cases} along, & (v_i, v_j) \in \mathcal{E} \\ rev, & i! = j\&(v_j, v_i) \in \mathcal{E} \\ loop, & i == j \end{cases} \tag{2}$$

where the new $K(w_i, w_j)$ only have three type labels.

As not all types of edges are equally informative for the downstream task, moreover, there are also noises in the generated syntactic parsing structures; we apply gates on the edges to weight their individual importances. Inspired by Dauphin et al. (2017); Marcheggiani and Titov (2017), we calculate a weight $g_{u,v}^{(k)}$ for each edge (u, v) indicating the importance for event extraction by:

$$g_{u,v}^{(k)} = \sigma(h_u^{(k)} V_{K(u,v)}^{(k)} + d_{K(u,v)}^{(k)}) \tag{3}$$

where σ is the logistic sigmoid function, $V_{K(u,v)}^{(k)}$ and $d_{K(u,v)}^{(k)}$ are the weight matrix and the bias of the gate. With this additional gating mechanism, the final syntactic GCN computation is formulated as

$$h_v^{(k+1)} = f(\sum_{u \in \mathcal{N}(v)} g_{u,v}^{(k)} (W_{K(u,v)}^{(k)} h_u^{(k)} + b_{K(u,v)}^{(k)})) \tag{4}$$

As stacking k layers of GCNs can model information in k hops, and sometimes the length of shortcut path between two triggers is less than k, to avoid information over-propagating, we adapt highway units (Srivastava et al., 2015), which allow unimpeded information flowing across stacking GCN layers. Typically, highway layers conduct nonlinear transformation as:

$$t = \sigma(W_T h_v^k + b_T) \tag{5}$$

$$\overline{h}_v^{(k+1)} = h_v^{(k+1)} + t \odot g(W_H h_v^k + b_H) + (1-t) \odot h_v^k \tag{6}$$

where σ is the sigmoid function; $\odot$ is the element-wise product operation; g is a nonlinear activation function; t is called transform gate and $(1 - t)$ is called carry gate. Therefore, the input of the k-th GCN layers should be $\overline{h}^{(k)}$ instead of $h^{(k)}$.

The GCNs are designed to capture the dependencies between shortcut arcs, while the layer number of GCNs limits the ability to capture local graph information. However, in this cases, we find that leveraging local sequential context will help to expand the information flow without increasing the layer number of GCNs, which means LSTMs and GCNs maybe complementary. Therefore, instead of feeding the word representation $X = (x_1, x_2, ..., x_n)$ into the first GCN layer, we follow Marcheggiani and Titov (2017), apply Bidirectional LSTM (Bi-LSTM) (Hochreiter and Schmidhuber, 1997) to encode the the word representation X as:

$$\overrightarrow{p}_t = \overrightarrow{LSTM}(\overrightarrow{p}_{t-1}, x_t) \tag{7}$$

$$\overleftarrow{p}_t = \overleftarrow{LSTM}(\overleftarrow{p}_{t-1}, x_t) \tag{8}$$

and the input of t-th token to GCNs is $\overline{x}_t = [\overrightarrow{p}_t, \overleftarrow{p}_t]$, where $[,]$ is the concatenation operation. The Bi-LSTM adaptively accumulates and abstracts the context for each token in the sentence.

2.3 Self-Attention Trigger Classification

When taking each token as the current word, we get the representation D from all tokens calculated by GCNs. Traditional event extraction systems often use max-pooling or its amelioration to aggregate information to each position. However, the max-pooling aggregation mechanisms tend to produce similar results after GCN modules in our framework. For example, if we get the aggregated vector Ag_i at each position i by this max-pooling mechanism $Ag_i = max\_pooling_{j=1}^{n}(H_j)$ with the GCNs output $\{H_j | j = 1, ..., n\}$ in which n is the sentence length, and the vector Ag_i is all the same at each position. Besides, predicting a trigger label for a token should take other possible trigger candidates into consideration. To capture the associations between triggers in a sentence, we design a self-attention mechanism to aggregate information especially keeping the associations between multiple events.

Given the current token w_i, the self-attention score vector and the context vector at position i are calculated as:

$$score = norm(exp(W_2 f(W_1 D + b_1) + b_2)) \tag{9}$$

$$C_i = [\sum_{j=1,j!=i}^{n} score_j * D_j, D_i] \quad (10)$$

where $norm$ means the normalization operation. Then we feed the context vector C_i into a fully-connected network to predict the trigger label in BIO annotation schema as:

$$\overline{C}_i = f(W_c C_i + b_c) \quad (11)$$

$$y_{t_i} = softmax(W_t \overline{C}_i + b_t) \quad (12)$$

where f is a non-linear activation and y_{t_i} is the final output of the i-th trigger label.

2.4 Argument Classification

When we have extracted an entire trigger candidate, which is meeting an *O* label after an *I-Type* label or a *B-Type* label, we use the aggregated context vector $\overline{C}$ to perform argument classification on the entity list in the sentence.

For each entity-trigger pair, as both the entity and the trigger candidate are likely to be a subsequence of tokens, we aggregate the context vectors of subsequences to trigger candidate vector T_i and entity vector E_j by average pooling along the sequence length dimension. Then we concatenate them together and feed into a fully-connected network to predict the argument role as:

$$y_{a_{ij}} = softmax(W_a[T_i, E_j] + b_a) \quad (13)$$

where $y_{a_{ij}}$ is the final output of which role the j-th entity plays in the event triggered by the i-th trigger candidate.

When training our framework, if the trigger candidate that we focus on is not a correct trigger, we set all the golden argument labels concerning the trigger candidate to *OTHER* (not any roles). With this setting, the labels of the trigger candidate will be further adjusted to reach a reasonable probability distribution.

2.5 Biased Loss Function

In order to train the networks, we minimize the joint negative log-likelihood loss function. Due to the data sparsity in the ACE 2005 dataset, we adapt our joint negative log-likelihood loss function by adding a bias item as:

$$\begin{aligned} J(\theta) = -\sum_{p=1}^{N}(\sum_{i=1}^{n_p} I(y_{t_i}) log(p(y_{t_i}|\theta)) \\ + \beta \sum_{i=1}^{t_p}\sum_{j=1}^{e_p} log(p(y_{a_{ij}}|\theta))) \end{aligned} \quad (14)$$

where N is the number of sentences in training corpus; n_p, t_p and e_p are the number of tokens, extracted trigger candidates and entities of the p-th sentence; $I(y_{t_i})$ is an indicating function, if y_{t_i} is not *O*, it outputs a fixed positive floating number α bigger than one, otherwise one; β is also a floating number as a hyper-parameter like α.

3 Experiments

3.1 Experiment Settings

Dataset, Resources and Evaluation Metric

We evaluate our JMEE framework on the ACE 2005 dataset. The ACE 2005 dataset annotate 33 event subtypes and 36 role classes, along with the NONE class and BIO annotation schema, we will classify each token into 67 categories in event detection and 37 categories in argument extraction. To comply with previous work, we use the same data split as the previous work (Ji and Grishman, 2008; Liao and Grishman, 2010; Li et al., 2013; Chen et al., 2015; Liu et al., 2016b; Yang and Mitchell, 2016; Nguyen et al., 2016; Sha et al., 2018). This data split includes 40 newswire articles (881 sentences) for the test set, 30 other documents (1087 sentences) for the development set and 529 remaining documents (21,090 sentences) for the training set.

We deploy the Stanford CoreNLP toolkit[5] to preprocess the data, including tokenizing, sentence splitting, pos-tagging and generating dependency parsing trees.

Also, we follow the criteria of the previous work (Ji and Grishman, 2008; Liao and Grishman, 2010; Li et al., 2013; Chen et al., 2015; Liu et al., 2016b; Yang and Mitchell, 2016; Nguyen et al., 2016; Sha et al., 2018) to judge the correctness of the predicted event mentions.

Hyperparameter Setting

For all the experiments below, in the word representation module, we use 300 dimensions for the embeddings and 50 dimensions for the rest

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

Method	Trigger Identification (%)			Trigger Classification (%)			Argument Identification (%)			Argument Role (%)		
	P	R	F_1	P	R	F_1	P	R	F_1	P	R	F_1
Cross-Event		N/A		68.7	68.9	68.8	50.9	49.7	50.3	45.1	44.1	44.6
JointBeam	76.9	65.0	70.4	73.7	62.3	67.5	69.8	47.9	56.8	64.7	44.4	52.7
DMCNN	**80.4**	67.7	73.5	75.6	63.6	69.1	68.8	51.9	59.1	62.2	46.9	53.5
PSL		N/A		75.3	64.4	69.4		N/A			N/A	
JRNN	68.5	**75.7**	71.9	66.0	**73.0**	69.3	61.4	64.2	62.8	54.2	56.7	55.4
dbRNN		N/A		74.1	69.8	71.9	71.3	64.5	67.7	66.2	52.8	58.7
JMEE	80.2	72.1	**75.9**	**76.3**	71.3	**73.7**	**71.4**	**65.6**	**68.4**	**66.8**	**54.9**	**60.3**

Table 1: Overall performance comparing to the state-of-the-art methods with golden-standard entities.

three embeddings including pos-tagging embedding, positional embedding and entity type embedding. In the syntactic GCN module, we use a three-layer GCN, a one-layer Bi-LSTM with 220 hidden units, self-attention with 300 hidden units and 200 hidden units for the rest transformation. We also set dropout rate to 0.5 and L2-norm to 1e-8. The batch size in our experiments is 32, and we utilize a maximum length $n = 50$ of sentences in the experiments by padding shorter sentences and cutting off longer ones. These hyperparameters are either randomly searched or chosen by experiences when tuning in the development set.

We use ReLU (Glorot et al., 2011) as our nonlinear activate function. We apply the stochastic gradient descent algorithm with mini-batches and the AdaDelta update rule (Zeiler, 2012). The gradients are computed using back-propagation. During training, besides the weight matrices, we also fine-tune all the embedding tables.

3.2 Overall Performance

We compare our performance with the following state-of-the-art methods:

1 **Cross-Event** is proposed by Liao and Grishman (2010), which uses document level information to improve the performance of event extraction;

2 **JointBeam** is the method proposed by Li et al. (2013), which extracts events based on structure prediction by manually designed features;

3 **DMCNN** is proposed by Chen et al. (2015), which uses dynamic multi-pooling to keep multiple events' information;

4 **PSL** is proposed by Liu et al. (2016b), which uses a probabilistic reasoning model to classify events by using latent and global information to encode the associations between events;

5 **JRNN** is proposed by Nguyen et al. (2016), which uses a bidirectional RNN and manually designed features to jointly extract event triggers and arguments.

6 **dbRNN** is proposed by Sha et al. (2018), which adds dependency bridges over Bi-LSTM for event extraction.

Table 1 shows the overall performance comparing to the above state-of-the-art methods with golden-standard entities. From the table, we can see that our JMEE framework achieves the best F_1 scores for both trigger classification and argument-related subtasks among all the compared methods. There is a significant gain with the trigger classification and argument role labeling performances, which is 2% higher over the best-reported models. These results demonstrate the effectivenesses of our method to incorporate with the graph convolution and syntactic shortcut arcs.

3.3 Effect on Extracting Multiple Events

To evaluate the effect of our framework for alleviating the multiple events phenomenon, we divide the test data into two parts (**1/1** and **1/N**) following Nguyen et al. (2016); Chen et al. (2015) and perform evaluations separately. **1/1** means that one sentence only has one trigger or one argument plays a role in one sentence; otherwise, **1/N** is used.

Table 2 illustrates the performance (F_1 scores) of **JRNN** (Nguyen et al., 2016), **DMCNN** (Chen et al., 2015), the two baseline model **Embedding+T** and **CNN** in Chen et al. (2015) and our framework in trigger classification subtask and argument role labeling subatsk. **Embedding+T** uses word embedding vectors and the traditional sentence-level features in Li et al. (2013), while

Stage	Model	1/1	1/N	all
Trigger	Embedding+T	68.1	25.5	59.8
	CNN	72.5	43.1	66.3
	DMCNN	74.3	50.9	69.1
	JRNN	**75.6**	64.8	69.3
	JMEE	75.2	**72.7**	**73.7**
Argument	Embedding+T	37.4	15.5	32.6
	CNN	51.6	36.6	48.9
	DMCNN	54.6	48.7	53.5
	JRNN	50.0	55.2	55.4
	JMEE	**59.3**	**57.6**	**60.3**

Table 2: System Performance on Single Event Sentences **(1/1)** and Multiple Event Sentences **(1/N)**

CNN is similar to **DMCNN**, except that it applies the standard max-pooling mechanism instead of the dynamic multi-pooling mechanism. We can see that our framework significantly outperforms all the other methods, especially in trigger classification subtask. In the **1/N** data split of triggers, our framework is 7.9% better than the **JRNN**, which demonstrates that our method of leveraging syntactic shortcut arcs and self-attention aggregation mechanism is helpful in alleviating the multiple events phenomenon.

3.4 Analysis of Self-Attention Mechanism

We use a sentence "police have arrested four people in connection with the killing" as an example to illustrate the captures features in our self-attention aggregation mechanism by transforming the attention scores to a row-wise heap map in Figure 4. There are two events in the sentence: an *Arrest-Jail* event triggered by *arrested* and a *Die* event triggered by *killings*. Additionally, the entity *police* plays an *Agent* role and the entity *four people* plays a *Person* role in the *Arrest-Jail* event.

As we can see from the Figure 4, in the row of *arrested*, there are relatively strong connections with *arrested* (self), *four people* (its argument) and *killings* (other event). And in the row of *killings*, there are also relatively strong connections with *killings* (self) and *arrested* (other event). Besides, the words *police*, *four* and *in* also have high scores with *killings*, which may mean be on account of the context information propagation though syntactic shortcut arcs.

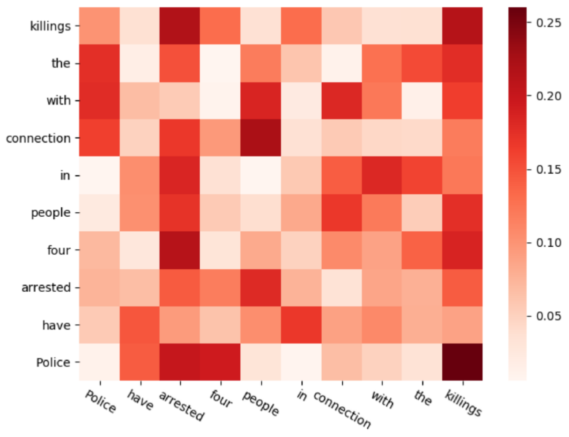

Figure 4: Visualization of the attention scores of a sentence containing two events: an *Arrest-Jail* event triggered by *arrested* and a *Die* event triggered by *killings*. Each row is the group of scores derived by the self-attention aggregation mechanism. Darker red means higher score and stronger interaction.

4 Related Work

There are several existing approaches exploiting the associations between events in the event extraction task. Some of them alleviate this phenomenon by exploiting various sentence-level features, such as ranking dependencies (McClosky et al., 2011), combinational features of triggers and arguments (Li et al., 2013), probabilistic soft logic information (Liu et al., 2016b,a), trigger-specific features and relational features (Yang and Mitchell, 2016; Keith et al., 2017). Others also seek features in document-level methods (Liao and Grishman, 2010; Ji and Grishman, 2008; Hong et al., 2011; Reichart and Barzilay, 2012; Lu and Roth, 2012). The feature-based methods require extensive human engineering, which also essentially affects model performances, and learn them from the unbalanced training data, however, it is difficult for sparse events.

There are also a group of deep learning methods using RNNs (Nguyen et al., 2016; Sha et al., 2018; Liu et al., 2018) and CNNs (Chen et al., 2015; Feng et al., 2016; Nguyen and Grishman, 2016) capturing the associations between events. However, sentence-level sequential modeling methods suffer a lot from the low efficiency in capturing very long-range dependencies. Besides, these methods do not fully model the associations between events.

5 Conclusion and Future Work

This paper presents a novel deep neural jointly multiple events extraction (JMEE) framework for the task of event extraction, especially for alleviating the multiple-event phenomenon. In our framework, we introduce syntactic shortcut arcs to enhance information flow and adapt the graph convolution network to capture the enhanced representation. Then a self-attention aggregation mechanism is applied to aggregate the associations between events. Besides, we jointly extract event triggers and arguments by optimizing a biased loss function due to the imbalances in the dataset. The experiment results demonstrate the effectiveness of our proposed framework. In the future, we plan to exploit the information of one argument which plays different roles in various events to do better in event extraction task.

Acknowledgments

We would like to thank Yansong Feng, Ying Zeng, Xiaochi Wei, Qian Liu and Changsen Yuan for their insightful comments and suggestions. We also very appreciate the comments from anonymous reviewers which will help further improve our work. This work is supported by National Natural Science Foundation of China (No. 61751201 and No. 61602490) and National Key R&D Plan (No. 2017YFB0803302).

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 of the Asian Federation of Natural Language Processing*, pages 167–176.

Yann N. Dauphin, Angela Fan, Michael Auli, and David Grangier. 2017. Language modeling with gated convolutional networks. In *Proceedings of the 34th International Conference on Machine Learning*, pages 933–941.

Xiaocheng Feng, Lifu Huang, Duyu Tang, Heng Ji, Bing Qin, and Ting Liu. 2016. A language-independent neural network for event detection. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 66–71.

Xavier Glorot, Antoine Bordes, and Yoshua Bengio. 2011. Deep sparse rectifier neural networks. In *Proceedings of the 14th International Conference on Artificial Intelligence and Statistics*, pages 315–323.

Ralph Grishman, David Westbrook, and Adam Meyers. 2005. Nyu's english ace 2005 system description. *Journal on Satisfiability*, 51(11):1927–1938.

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

Yu Hong, Jianfeng Zhang, Bin Ma, Jian-Min Yao, Guodong Zhou, and Qiaoming Zhu. 2011. Using cross-entity inference to improve event extraction. In *roceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 1127–1136.

Heng Ji and Ralph Grishman. 2008. Refining event extraction through cross-document inference. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics*, pages 254–262.

Katherine A. Keith, Abram Handler, Michael Pinkham, Cara Magliozzi, Joshua McDuffie, and Brendan O'Connor. 2017. Identifying civilians killed by police with distantly supervised entity-event extraction. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1547–1557.

Thomas N. Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *CoRR*, abs/1609.02907.

Dan Klein and Christopher D. Manning. 2003. Accurate unlexicalized parsing. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 423–430.

Qi Li, Heng Ji, and Liang Huang. 2013. Joint event extraction via structured prediction with global features. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, pages 73–82.

Xiang Li, Thien Huu Nguyen, Kai Cao, and Ralph Grishman. 2015. Improving event detection with abstract meaning representation. In *Proceedings of the 1st Workshop on Computing News Storylines*, pages 11–15.

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.

Jian Liu, Yubo Chen, Kang Liu, and Jun Zhao. 2018. Event detection via gated multilingual attention mechanism. In *Proceedings of the 32nd AAAI Conference on Artificial Intelligence*, pages 4865–4872.

Shulin Liu, Yubo Chen, Shizhu He, Kang Liu, and Jun Zhao. 2016a. Leveraging framenet to improve automatic event detection. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 2134–2143.

Shulin Liu, Yubo Chen, Kang Liu, and Jun Zhao. 2017. Exploiting argument information to improve event detection via supervised attention mechanisms. In

Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, pages 1789–1798.

Shulin Liu, Kang Liu, Shizhu He, and Jun Zhao. 2016b. A probabilistic soft logic based approach to exploiting latent and global information in event classification. In *Proceedings of the 30th AAAI Conference on Artificial Intelligence*, pages 2993–2999.

Wei Lu and Dan Roth. 2012. Automatic event extraction with structured preference modeling. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics*, pages 835–844.

Diego Marcheggiani and Ivan Titov. 2017. Encoding sentences with graph convolutional networks for semantic role labeling. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1506–1515.

David McClosky, Mihai Surdeanu, and Christopher D. Manning. 2011. Event extraction as dependency parsing. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 1626–1635.

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. 2016. Modeling skip-grams for event detection with convolutional neural networks. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 886–891.

Thien Huu Nguyen and Ralph Grishman. 2018. Graph convolutional networks with argument-aware pooling for event detection. In *Proceedings of the 32nd AAAI Conference on Artificial Intelligence*, pages 5900–5907.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 1532–1543.

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

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 *Proceedings of the 32nd AAAI Conference on Artificial Intelligence*, pages 5916–5923.

Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. 2015. Training very deep networks. In *Proceedings of the 28th Annual Conference on Neural Information Processing Systems*, pages 2377–2385.

Bishan Yang and Tom M. Mitchell. 2016. Joint extraction of events and entities within a document context. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 289–299.

Matthew D. Zeiler. 2012. ADADELTA: an adaptive learning rate method. *CoRR*, abs/1212.5701.