Improving Cascade Decoding with Syntax-aware Aggregator and Contrastive Learning for Event Extraction

Zeyu Sheng , Yuanyuan Liang , Yunshi Lan*
School of Data Science & Engineering, East China Normal University, Shanghai, China
{51205903051,leonyuany}@stu.ecnu.edu.cn, yslan@dase.ecnu.edu.cn

Abstract

Cascade decoding framework has shown superior performance on event extraction tasks. However, it treats a sentence as a sequence and neglects the potential benefits of the syntactic structure of sentences. In this paper, we improve cascade decoding with a novel module and a self-supervised task. Specifically, we propose a syntax-aware aggregator module to model the syntax of a sentence based on cascade decoding framework such that it captures event dependencies as well as syntactic information. Moreover, we design a type discrimination task to learn better syntactic representations of different event types, which could further boost the performance of event extraction. Experimental results on two widely used event extraction datasets demonstrate that our method could improve the original cascade decoding framework by up to 2.2% percentage points of F1 score and outperform a number of competitive baseline methods.

1 Introduction

As an important yet challenging task in natural language processing, event extraction has attracted much attention for decades (Chen et al., 2015; Nguyen and Grishman, 2018; Zheng et al., 2019; Lai et al., 2021; Wang et al., 2021; Li et al., 2022; Ma et al., 2022; Zhou et al., 2022). This task aims at predicting event types, triggers and arguments from a given sentence. We display three examples in Figure 1. Given an example sentence (a) "*In 2018, Chuangwei Tech acquired equity of Qianhong Electronics for 1.5 billion* ...", an event extraction system is able to recognize the trigger "*acquired*", that corresponds to the event type "*invest*", and the argument "*Chuangwei Tech*", that plays the subject role of "*sub*" in the event.

A great number of methods have been developed for event extraction. Early methods formulate the event extraction as a sequence labeling task, where each token is considered as a candidate for labeling. They perform trigger extraction and argument extraction with joint learning (Li et al., 2013; Nguyen et al., 2016; Nguyen and Nguyen, 2019), which easily causes the label conflict issue. Considering the precedence relationship between the components in an event, pipeline methods are explored to perform trigger and argument extraction in separate stages (Chen et al., 2015; Du and Cardie, 2020; Liu et al., 2020; Ma et al., 2022). But the error is accumulated along with the pipeline. Recently, a cascade decoding framework (Xu et al., 2020; Sheng et al., 2021) is proposed to extract events with a cascade tagging strategy, which could not only handle the label conflict issue, but also avoid error propagation.

In above methods, a sentence is treated as a sequence, and methods suffer from the low efficiency problem in capturing long-range dependency. We take sentence (a) in Figure 1 as an example. The argument "*1.5 billion*" is far from the trigger "*acquired*" based on the sequential order while they are closely connected via the dependency arc. Therefore, it is necessary to take advantage of the syntactic structure to capture the relations between triggers and arguments. Some researches managed to include syntactic information of sentences in event extraction. Chen et al.(Chen et al., 2015) first employed dependency trees to conduct event extraction. Nguyen et al.(Nguyen and Grishman, 2018) and Yan et al.(Yan et al., 2019) treated each dependency tree as a graph and adopted Graph Convolution Network (GCN) (Kipf and Welling, 2017) to represent the sentence. More recent studies strengthened the graph representation via gate mechanism to filter out noisy syntactic information (Lai et al., 2020) or empowered the graph

*Corresponding author

Proceedings of the 22nd China National Conference on Computational Linguistics, pages 748-760, Harbin, China, August 3 - 5, 2023.

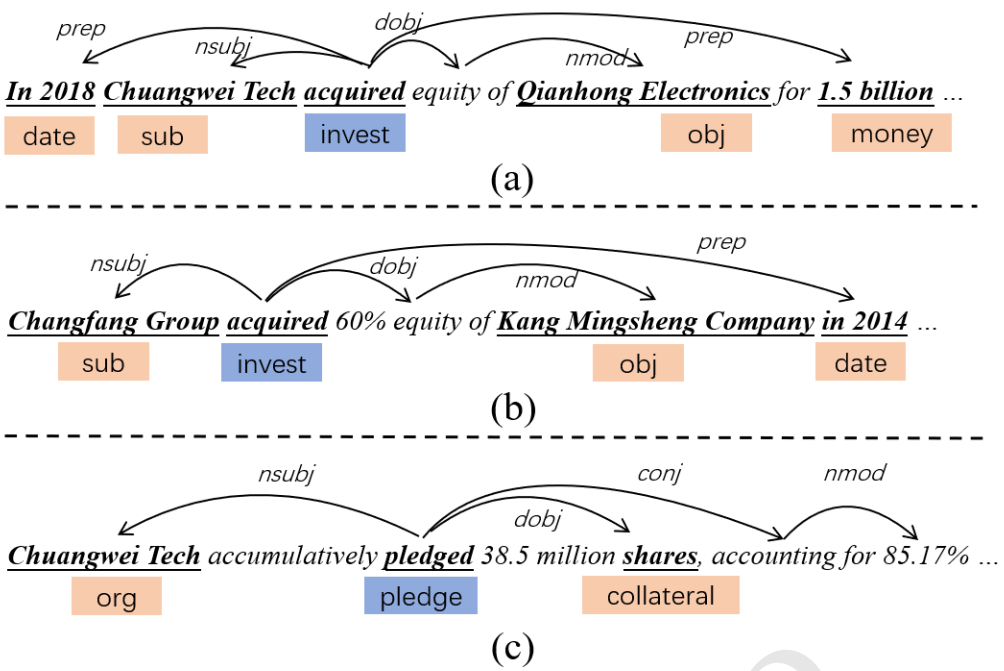

Figure 1: Three examples of event extraction. We annotate the event types with blue boxes under the triggers, and label the argument roles with orange boxes under the arguments.

encoder with more advanced Transformer (Ahmad et al., 2021). These methods could effectively solve the long-range dependency issue. However, they either follow the joint learning paradigm or pipeline paradigm thus still encounter the issue of label conflict or error propagation. In this paper, we develop our approach modeling syntactic information for event extraction based on cascade decoding framework. To achieve this, two main challenges should be addressed.

First, cascade decoding represents event types, triggers as well as arguments in the format of a triple. It sequentially predicts components in triples as subtasks and learns the implicit dependencies of the subtasks. It is not trivial to design a syntax encoder which is customized for the cascade decoders. In this paper, we propose a novel *Syntax-enhanced Aggregator* which could not only integrate the information from the precedent subtask with the current subtask but also model the syntactic structure of sentences. Moreover, this module could fuse the heterogeneous features together. In detail, our aggregator processes both subtask dependency and syntactic information via two channels. The final representation will be fused based on the alignment between tokens of a sentence and components in a dependency tree. Such aggregators are deployed in cascade decoders.

Second, existing methods involving syntactic structure rarely consider the interaction among event types. As examples (a) and (b) shown in Figure 1, the sentences of the same event type usually have similar syntactic structure despite different involved entities. In contrast, the sentences of the different event types usually have different syntactic structure despite similar involved entities, as examples (a) and (c). We design *Contrastive Learning* of syntactic representation to capture the interactions between sentences. Specifically, we define a type discrimination task to distinguish whether two sentences belong to the same event type based on their syntactic representations. This is jointly trained with event extraction task.

We conduct experiments on two event extraction datasets, FewFC (Zhou et al., 2021) and DuEE (Li et al., 2020b). The experiments show that compared with original cascade framework, our method can clearly perform better on both datasets. Our method also outperforms competitive baseline methods that represent the state-of-the-art on event extraction tasks. To reveal the working mechanism of our method, we also conduct ablation study and visualization that shed light on where the improvement comes from.

We summarize the contributions of this paper as follows: (1) We propose a novel syntax-enhanced aggregator to model the syntactic structure of sentences, which is a good fit for the cascade decoding framework. This aggregator is able to model syntax and fuse with dependencies of events. (2) To

Proceedings of the 22nd China National Conference on Computational Linguistics, pages 748-760, Harbin, China, August 3 - 5, 2023.

further benefit from the syntax modeling, we design a type discrimination task to refine the syntactic representation via contrastive learning. (3) We empirically show the effectiveness of our method on two datasets. Our proposed method outperforms the baseline methods with remarkable margins based on F1 score of all measurement metrics.

2 Background

2.1 Problem Formulation

The task of event extraction aims at identifying event triggers with their types and event arguments with their roles. Specifically, a pre-defined event schema contains an event type set $\mathcal{C}$ and an argument role set $\mathcal{R}$. Given a sentence $x = \{w_1, w_2, ..., w_n\}$, the goal is to predict all events in gold set $\mathcal{E}_x$ of the sentence x, where the components of $\mathcal{E}_x$ are in the format of triples (c, t, a_r). Here, $c \in \mathcal{C}$ is an event type, t is a trigger word in sentence x, and a_r is an argument word corresponding to the role $r \in \mathcal{R}$. A dataset $\mathcal{D}$ consists of a set of $(x, \mathcal{E}_x)$.

2.2 A Cascade Decoding Framework

To solve the task, we follow the existing cascade decoding approach, CasEE method (Sheng et al., 2021), which is proposed to predict the events by maximizing the following joint likelihood:

$$\begin{aligned}&\prod_{(x,\mathcal{E}_x)\in\mathcal{D}}[\prod_{(c,t,a_r)\in\mathcal{E}_x} P((c,t,a_r)|x)]\\ =&\prod_{(x,\mathcal{E}_x)\in\mathcal{D}}[\prod_{c\in\mathcal{C}} P(c|x)\prod_{t\in\mathcal{T}_x} P(t|x,c)\prod_{a_r\in\mathcal{A}_{x,r}} P(a_r|x,c,t)],\end{aligned} \tag{1}$$

where $\mathcal{T}_x$ and $\mathcal{A}_{x,r}$ denote trigger and argument sets of x, respectively.

The joint likelihood explicates the dependencies among the type, trigger, and argument. The order of cascade decoding indicates that the framework first learns a *Type Decoder* $P(c|x)$ to identify the event types in the sentence. Then, it extracts the trigger words from the sentence via a *Trigger Decoder* $P(t|x, c)$ which corresponds to the detected type. After that, an *Argument Decoder* $P(a_r|x, c, t)$ is developed to extract role-specific arguments.

In the cascade decoding approach, the decoders are built upon a sharing BERT encoder:

$$\{\mathbf{h}_1, \mathbf{h}_2, ..., \mathbf{h}_n\} = \text{BERT}(x), \tag{2}$$

where $\mathbf{H} = \{\mathbf{h}_1, \mathbf{h}_2, ..., \mathbf{h}_n\}$ is the hidden representation of x for downstream decoding. Next, an attention layer followed by a simple feed-forward neural network is leveraged as the type decoder to predict the event type. We denote it as:

$$P(c|x) = \text{TypeDecoder}(\mathbf{H}). \tag{3}$$

After that, the predicted type embedding $\mathbf{c}$ is concatenated with each token representation. This will be further processed via a conditional layer normalization (CLN) (Lee et al., 2021) layer and a self-attention layer to form the hidden representation $\mathbf{H}^c$. A pointer network takes charge of predicting the position of start and end indexes based on $\mathbf{H}^c$. We denote the above trigger extraction procedure as follows:

$$\begin{aligned}\mathbf{H}^c &= \text{Aggregator}(\mathbf{H}, \mathbf{c}),\\ P(t|x,c) &= \text{Pointer}(\mathbf{H}^c).\end{aligned} \tag{4}$$

For argument decoder, the trigger information is concatenated with $\mathbf{H}^c$ and processed with a CLN to form the hidden representation $\mathbf{H}^{ct}$. Given $\mathbf{H}^{ct}$, the start and end indexes of role-specific arguments are then predicted as follows:

$$\begin{aligned}\mathbf{H}^{ct} &= \text{Aggregator}(\mathbf{H}^c, \mathbf{t}),\\ P(a_r|x,c,t) &= \text{Pointer}(\mathbf{H}^{ct}).\end{aligned} \tag{5}$$

More details could be found in the original paper (Sheng et al., 2021).

Proceedings of the 22nd China National Conference on Computational Linguistics, pages 748-760, Harbin, China, August 3 - 5, 2023.

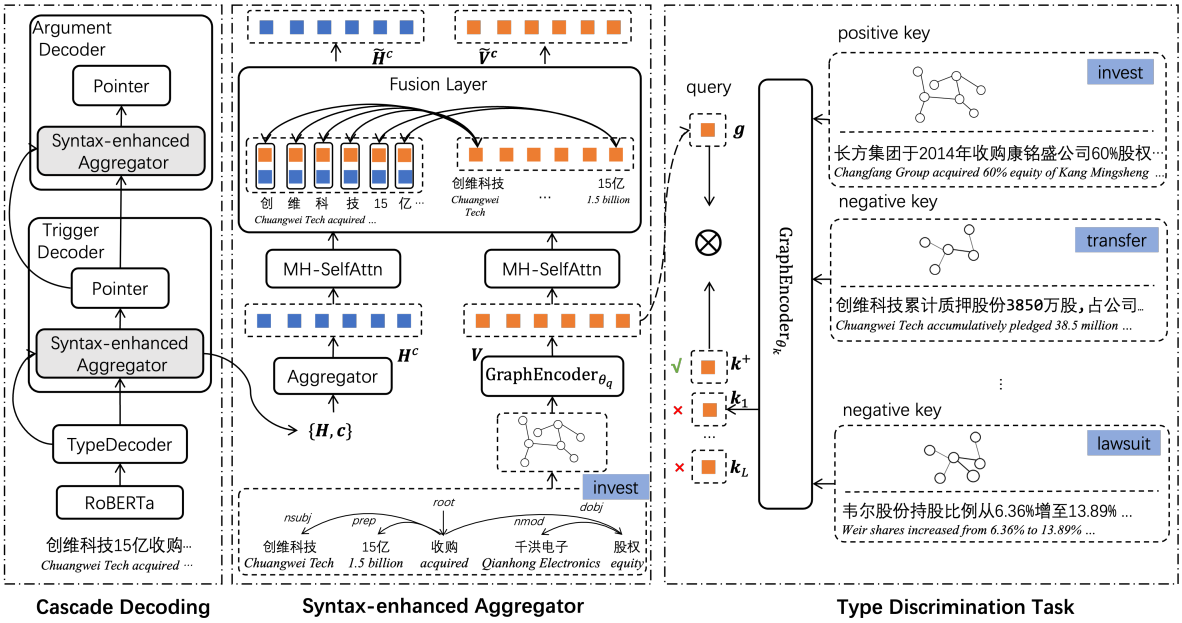

Figure 2: The overall architecture of our approach. The network modules are annotated with solid boxes and data is annotated with imaginary boxes. The left part is the cascade decoding framework. We modify the original aggregators to syntax-enhanced aggregators. The middle part shows the details of proposed syntax-enhanced aggregator in trigger decoder, where dependency and syntactic information are carried via two channels and eventually fuse together in fusion layer. The right part shows the details of type discrimination task, where syntactic representations belonging to the same event type are learned to be closer. Please note the imaginary line from the aggregator to the discriminator is meant to show the input of the discriminator rather than forward pass of the architecture.

3 Our Approach

The cascade decoding framework that we described in Section 2.2 decodes different components of events in a cascading manner, the inputs of which are hidden representations of tokens featured with subtask dependencies. Our approach follows the framework, but we improve it by introducing a module to fuse the syntactic information over the decoding process and a self-supervised task to further refine the syntactic representation. Specifically, we propose *Syntax-enhanced Aggregators* to take place of the original aggregators. The proposed aggregator elaborately models the syntactic structure of the sentence and fuses syntax with the original hidden representation, as we will explain in Section 3.1. To better capture the interactions among event types, we design a *Type Discrimination Task* to distinguish whether the representations belonging to the same type are syntactically close or not, which will be presented in Section 3.2. Eventually, event detection and type discrimination generate their training objectives and we join them together, as we will describe in Section 3.3. The overall architecture of our approach is displayed in Figure 2.

3.1 Syntax-enhanced Aggregator

Recall that we could prepare the hidden representations enriched with dependency information $\mathbf{H}^c$ and $\mathbf{H}^{ct}$ through the aggregators in trigger decoder and argument decoder, respectively. Now we describe, in our syntax-enhanced aggregator, how we obtain the syntactic representations and fuse these heterogeneous features to form new representations $\tilde{\mathbf{H}}^c$ and $\tilde{\mathbf{H}}^{ct}$. For simplicity, we take $\mathbf{H}^c$ in trigger decoder as the example. The similar procedure is conducted for $\mathbf{H}^{ct}$ in argument decoder.

We first extract the dependency tree of the sentence via existing parsing tools. To avoid one way message transition from the root to leaf nodes, we add reversed edges and distinguish them with different edge labels in the dependency tree. This results in a syntactic graph $\mathcal{G}(v, e)$, where v is the entity in a

dependency tree and e is the grammatical link between these entities. The representation of entities are updated along with the graph structure. Let us use $\mathbf{V} = \{\mathbf{v}_1, ...\mathbf{v}_m\}$ to denote the representations of m entities in $\mathcal{G}$. Each entity is initially represented via the average embeddings of their tokens.

To model the syntactic structure of sentences, we adopt the commonly used Relational Graph Convolutional Network (R-GCN) (Schlichtkrull et al., 2018) as our graph encoder to capture the message transition of the syntactic graph:

$$\{\mathbf{v}_1, ...\mathbf{v}_m\} = \text{GraphEncoder}(\{\mathbf{v}_1, ...\mathbf{v}_m)\}. \tag{6}$$

In this way, the updated entity representation is featured with sentence syntax. Next, we aggregate them with the original hidden representations $\mathbf{H}^c = \{\mathbf{h}_1^c, ..., \mathbf{h}_n^c\}$, which are arranged in token level, such that we can fuse these two types of information together.

We first utilize two individual multi-head self-attentions (MH-SelfAttns) to process both $\mathbf{H}^c$ and $\mathbf{V}$, respectively. Inspired by the Knowledgeable Encoder proposed in prior work (Zhang et al., 2019), where the language representation is enhanced with knowledge graphs, we align an entity with its corresponding tokens or characters if it is formed by multiple tokens or characters. As shown in Figure 2, the entity "创维科技 *(Chuangwei Tech)*" is aligned with "创", "维", "科", and "技". Thus there are explicit links between this entity and the four characters. We define the fusion layer as follows:

$$\mathbf{z}_j = \sigma(\mathbf{U}_1\mathbf{h}_j^c + \sum_{v_i \in \mathcal{A}lign(w_j)} \mathbf{W}_1\mathbf{v}_i + \mathbf{b}_1) \tag{7}$$

$$\tilde{\mathbf{h}}_j^c = \sigma(\mathbf{U}_2\mathbf{z}_j + \mathbf{b}_{21}) \tag{8}$$

$$\tilde{\mathbf{v}}_i^c = \sigma(\sum_{w_j \in \mathcal{A}lign(v_i)} \mathbf{W}_2\mathbf{z}_j + \mathbf{b}_{22}), \tag{9}$$

where σ is non-linear activation function GELU (Hendrycks and Gimpel, 2016) and $\mathcal{A}lign$ indicates the alignment between tokens and entities. The inputs are hidden representation $\mathbf{H}^c$ and entity representation $\mathbf{V}$. $\mathbf{U}$, $\mathbf{W}$ and $\mathbf{b}$ with subscripts are parameters to learn. $\mathbf{z}_j$ indicates fused hidden representation of j-th token. As a result, $\tilde{\mathbf{H}}^c = \{\tilde{\mathbf{h}}_1^c, \tilde{\mathbf{h}}_2^c, ..., \tilde{\mathbf{h}}_n^c\}$ is the token representation with fusion of syntax information. It will be leveraged as the input of pointer network in Equation (4) for trigger extraction. $\tilde{\mathbf{V}}^c = \{\tilde{\mathbf{v}}_1^c, \tilde{\mathbf{v}}_2^c, ..., \tilde{\mathbf{v}}_m^c\}$ is the entity representation enriched with subtask dependencies. It will be utilized in downstream decoding.

When it comes to the argument decoder, $\tilde{\mathbf{V}}^c$ is used as the input entity representation to be continuously processed via the graph encoders and eventually fuse with the hidden representation $\mathbf{H}^{ct}$ to generate $\tilde{\mathbf{H}}^{ct}$. This will be fed into pointer network in Equation (5) for argument extraction. Compared with the original aggregator, besides capturing dependency information, our syntax-enhanced aggregators encode syntactic structure and fuse both subtask dependencies and syntactic information to generate a more expressive representation for decoding.

3.2 Type Discrimination Task

Type discrimination task aims at predicting whether two sentences are syntactically close or not. The intuition behind is that sentences describing the same event type usually have similar syntactic structure. To this end, we adopt the idea of contrastive learning and push the syntactic representations of positive pairs closer than negative pairs. The syntactic representations learned from type discrimination task can further boost the performance of cascade decoding.

We conduct dependency parsing for all sentences and obtain a collection of syntactic graphs denoting as $\mathcal{U}$, each $\mathcal{G} \in \mathcal{U}$ deriving from a sentence is labeled with their event type. Then, we train the representations of a pair of syntactic graphs that share the same event type to be closer in the space. We adopt Momentum Contrast (MoCo) (He et al., 2020) for self-supervised representation learning, which formulates contrastive learning as a dictionary look-up task and is effective in maintaining a large-scale dynamic dictionary.

Proceedings of the 22nd China National Conference on Computational Linguistics, pages 748-760, Harbin, China, August 3 - 5, 2023.

Specifically, given a syntactic graph $\mathcal{G}$ as a query, we represent it by the average of all entities encoded via the graph encoder of Equation (6) and obtain $\mathbf{g} = \frac{1}{m}\sum_{i=1}^{m}\mathbf{v}_i$ to indicate the status of the syntactic graph. Meanwhile, we sample a set of syntactic graphs from $\mathcal{U}$ as keys of a dictionary and encode these key graphs via another graph encoder to obtain their representations. For clear presentation, we denote the query graph encoder and key graph encoder as $\text{GraphEncoder}_{\theta_q}$ and $\text{GraphEncoder}_{\theta_k}$, respectively. In the dictionary, the positive key (denoted as $\mathbf{k}^+$) is the only graph having the same type as the query. The others are negative keys $\{\mathbf{k}_1, \mathbf{k}_2, ..., \mathbf{k}_L\}$, as depicted in Figure 2. We define the loss function of the type discrimination task as follows:

$$\mathcal{L}_{TD} = -\sum_{\mathcal{G}\in\mathcal{U}} log\frac{\exp(\mathbf{g}^\intercal\mathbf{k}^+/\tau)}{\sum_{i=0}^{L}\exp(\mathbf{g}^\intercal\mathbf{k}_i/\tau)}, \tag{10}$$

where τ is a temperature hyper-parameter. For each query, we construct one positive key and L negative keys.

Similar as MoCo, during training, the keys in the dictionary are progressively updated. For each new query graph $\mathcal{G}$, the old key graphs in the dictionary are removed and new key graphs are collected. Moreover, the parameters of the encoder of keys are driven by momentum update as follows:

$$\text{GraphEncoder}_{\theta_k} \leftarrow \gamma\text{GraphEncoder}_{\theta_k} + (1-\gamma)\text{GraphEncoder}_{\theta_q}, \tag{11}$$

where γ is the momentum coefficient. This results in a smooth evolution of $\text{GraphEncoder}_{\theta_k}$ as we can control the evolving progress.

3.3 Training Objective

During our training procedure, event extraction and type discrimination tasks are performed simultaneously. For each sampled data, a sentence and its corresponding syntactic graph are both collected for event extraction training. A dictionary of key graphs for a query graph is also prepared for contrastive learning.

The overall training objectives of our improved cascade decoding framework is shown as follows:

$$\mathcal{L} = \lambda\mathcal{L}_{EE} + (1-\lambda)\mathcal{L}_{TD}, \tag{12}$$

where $\mathcal{L}_{EE}$ is the negative logarithm of the joint likelihood of event extraction task in Equation (1), and λ is a hyper-parameter. All the parameters except for $\text{GraphEncoder}_{\theta_k}$ are updated by back-propagation.

4 Experiments

In this section, we conduct experiments to evaluate the proposed method. We first introduce our experiment settings including datasets and evaluation metrics, comparable methods, and implementation details in Section 4.1, Section 4.2, and Section 4.3. Next, we discuss our main results in Section 4.4. We show further analysis in Section 4.5

4.1 Datasets and Evaluation Metrics

We conduct experiments on two commonly used event extraction datasets:

- **FewFC** (Zhou et al., 2021)[1] is a public Chinese dataset for event extraction in the financial domain. It contains 10 event types and 19 role types. There are 12,890 sentences in the dataset. Following previous setting (Sheng et al., 2021), we split the dataset with the ratio 8 : 1 : 1 to form training, development, and test sets.
- **DuEE** (Li et al., 2020b)[2] is a relatively large Chinese event extraction dataset, which contains 19,640 sentences in total. The data is collected by crowdsourcing and contains 65 event types associated with 121 role types in real-world scenarios. We follow its default split setting to construct the data sets.

[1] https://github.com/TimeBurningFish/FewFC
[2] http://ai.baidu.com/broad/download

Proceedings of the 22nd China National Conference on Computational Linguistics, pages 748-760, Harbin, China, August 3 - 5, 2023.

Methods	TI			TC			AI			AC		
	P(%)	R(%)	F1(%)	P(%)	R(%)	F1(%)	P(%)	R(%)	F1(%)	P(%)	R(%)	F1(%)
DMCNN⋆	82.0	79.4	80.7	69.4	68.2	68.8	70.2	66.3	68.2	66.8	65.7	66.2
GCN-ED⋆	84.4	83.7	84.0	71.7	68.9	70.3	69.1	69.6	69.4	71.2	65.7	68.3
GatedGCN⋆	88.9	85.0	86.9	76.2	73.4	74.8	72.3	70.1	71.2	71.4	68.8	70.1
BERT-CRF	88.4	84.1	86.2	74.2	70.5	72.3	69.4	68.1	68.7	70.8	68.2	69.5
MQAEE	88.7	86.2	87.4	77.2	76.4	76.8	**72.7**	69.7	71.2	70.2	66.5	68.3
CasEE	89.1	87.8	88.4	77.8	78.6	78.2	71.6	73.2	72.4	71.2	72.4	71.8
Ours⋆	**90.1**	**88.9**	**89.5**	**78.1**	**79.4**	**78.7**	71.9	**77.0**	**74.4**	**71.5**	**75.7**	**73.5**

Table 1: Event extraction results on test set of FewFc dataset. P(%), R(%) and F1(%) denote percentages of precision, recall and F1 score, respectively. The methods annotated with "⋆" are those enriched with syntactic features.

We utilize the standard evaluation metrics (Chen et al., 2015; Du and Cardie, 2020) to evaluate performance of trigger detection and argument detection: (1) Trigger Identification (**TI**): If a predicted trigger word matches the gold word, this trigger is identified correctly. (2) Trigger Classification (**TC**): If a trigger is correctly identified and assigned to the correct type, it is correctly classified. (3) Argument Identification (**AI**): If an event type is correctly recognized and the predicted argument word matches the gold word, it is correctly identified. (4) Argument Classification (**AC**): If an argument is correctly identified and the predicted role matches the gold role type, it is correctly classified. We measure Precision, Recall and F1 score based on the above four metrics.

4.2 Comparable Methods

We choose a range of advanced approaches for event extraction as our baselines:

- **DMCNN** (Chen et al., 2015) is a pipeline with dynamic multi-pooling convolutional neural network and enriched encoded syntactic features. It is the early attempt adopting syntactic information into event extraction.
- **GCN-ED** (Nguyen and Grishman, 2018) develops a GCN based on dependency trees to perform event detection, where each word is treated as a trigger candidate and joint learning is performed to label words with event types.
- **GatedGCN** (Lai et al., 2020) is GCN-based model for event detection which uses a gating mechanism to filter noisy information. It also follows a joint learning paradigm.
- **BERT+CRF** (Du and Cardie, 2020) is a sequence labeling model with advanced pre-trained language model BERT for encoding sentences and conditional random field (CRF) for tagging labels.
- **MQAEE** (Li et al., 2020a) is a pipeline method that formulates the extraction task as a multi-turn question answering without any syntactic information involved.
- **CasEE** (Sheng et al., 2021) is the representative cascade decoding approach for event extraction, which simply treats a sentence as a sequence.

We either utilize official source codes or follow their descriptions to re-implement the baseline methods.

4.3 Implementation Details

For implementation, we use Chinese BERT Model (Devlin et al., 2018) in Transformers library[3] as our basic textual encoder to convert words into vector representations. For other parameters, we randomly initialize them. To obtain syntactic graphs, we extract the syntactic dependency of sentences via StanfordNLP parsing tool[4] and convert dependency trees into graphs via DGL[5] library. In our syntax-

[3] https://huggingface.co/
[4] https://nlp.stanford.edu/software/lex-parser.shtml
[5] https://www.dgl.ai/

Proceedings of the 22nd China National Conference on Computational Linguistics, pages 748-760, Harbin, China, August 3 - 5, 2023.

Methods	TI			TC			AI			AC		
	P(%)	R(%)	F1(%)	P(%)	R(%)	F1(%)	P(%)	R(%)	F1(%)	P(%)	R(%)	F1(%)
DMCNN⋆	78.4	80.2	79.3	79.4	76.3	77.8	69.2	67.4	68.3	67.2	65.6	66.4
GCN-ED⋆	82.4	76.2	79.2	81.6	76.2	78.8	71.3	69.5	70.4	70.9	64.5	67.5
GatedGCN⋆	**88.6**	83.0	85.7	82.4	80.5	81.4	**73.8**	71.6	72.7	**72.5**	68.4	70.4
BERT-CRF	87.2	77.6	82.1	80.4	77.4	78.8	70.6	68.1	69.3	70.5	66.7	68.5
MQAEE	87.9	82.1	84.9	80.9	79.4	80.1	73.2	71.7	72.4	71.0	69.7	70.3
CasEE	85.5	88.2	86.8	83.6	83.9	83.7	70.3	75.4	72.8	68.6	75.7	72.0
Ours⋆	87.7	**89.0**	**88.3**	**83.7**	**86.8**	**85.2**	72.8	**76.9**	**74.8**	71.2	**77.4**	**74.2**

Table 2: Event extraction results on test set of DuEE dataset. P(%), R(%) and F1(%) denote percentages of precision, recall and F1 score, respectively. The methods annotated with "⋆" are those enriched with syntactic features.

enhanced aggregator, we use 8 heads for MH-SelfAttns layers and 2 stacked R-GCN layers to form a GraphEncoder. For hyper-parameters, we search via grid search through pre-defined spaces and decide the best configuration based on the best F1 score on the development set. The dimension of hidden representations in graph encoders or aggregators are all set to 768. We use an Adam optimizer (Kingma and Ba, 2015) to train all trainable parameters. The initial learning rate is set to $1e-5$ for BERT parameters and $1e-4$ for the other parameters. A warmup proportion for learning rate is set to 10%. The training batch is set to 16 and the maximum training epoch is 30. The size of dictionary L is set to 1000 for contrastive learning. We set $\tau = 0.07$, $\lambda = 0.5$ and $\gamma = 0.8$. To avoid overfitting, we apply dropout layers in syntax-enhanced aggregators with a dropout ratio as 0.3.

4.4 Main Results

The performance of all methods on FewFC and DeEE datasets is displayed in Table 1 and Table 2, respectively. Based on the two tables, we have the following observations:

(1) For both datasets, our method surpasses all baseline methods with a remarkable margin and obtains new state-of-the-art results on F1 score of all measurement metrics. This shows that our method incorporating syntactic information with cascade decoding framework indeed brings the largest benefit for event extraction task. Compared with CasEE, our method shows gains on TI as well as AI measurement. This may because that leveraging syntactic relation of sentences captures long-range dependency and enables the model to retrieve more accurate trigger and arguments. Also, the gains on TC and AC may comes from contrastive learning, which helps the model label events by discriminating the different syntactic structure of event types.

(2) In the perspective of framework, compared with the joint learning and pipeline paradigms, cascade decoding could achieve better performance. CasEE outperforms BERT-CRF as well as MQAEE with marginal improvement on both datasets. As discussed in Section 1, cascade decoding framework could avoid label conflicts and error propagation effectively (Sheng et al., 2021), which reveals the necessity of developing methods based on cascade decoding framework.

(3) For methods featured with syntactic information, different methods show different effects. Specifically, DMCNN and GCN-ED are methods involving syntactic information, their performance on both datasets are not ideal, this may because that these two methods are developed upon un-contextual word embeddings thus cannot fully capture the deep semantics of sentences. GatedGCN takes advantage of BERT encoder and encodes syntactic information via GCN model and it could outperform the BERT-CRF method. Our method is also built upon BERT encoder and featured with syntax-enhanced aggregator and type discrimination task, which is effective in solving the label conflict and modeling syntactic information of sentences.

4.5 Further Analysis

Ablation study. To explore details of our proposed method, we show the result of ablation study in Table 3. As we can see, both syntax-enhanced aggregators and contrastive learning contribute to the entire system. After we omit the contrastive learning, the performance decreases. This indicates that

Proceedings of the 22nd China National Conference on Computational Linguistics, pages 748-760, Harbin, China, August 3 - 5, 2023.

	TI(%)	TC(%)	AI(%)	AC(%)
Our Model	**89.5**	**78.7**	**74.4**	**73.5**
w/o Contrastive learning	88.7	78.3	73.6	72.4
w/o Fusion Layer	89.0	78.4	73.6	72.8
w/o SA in Trigger Decoder	88.5	78.1	72.4	71.9
w/o SA in Argument Decoder	89.3	78.6	73.0	72.1

Table 3: Results of ablation study on FewFC dataset. We display the percentages of F1 score on all measurement metrics. **SA** denotes Syntax-enhanced Aggregator.

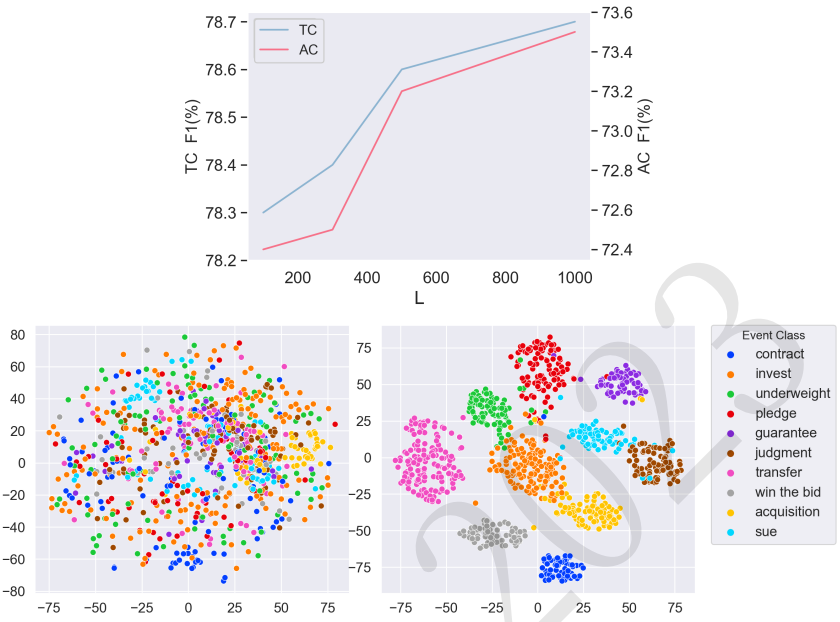

Figure 3: (a) shows the performance change of TC and AC on FewFC with increasing L value in contrastive learning. (b) shows the t-SNE plots of representations of query graphs of FewFC without and with contrastive learning.

capturing the syntactic structure of sentences is key for detecting the event types. Similarly, After we omit the fusion layer in syntax-enhanced aggregator and simply add the hidden representation of syntactic graph to $\mathbf{H}^c$, the performance drops. This indicates that the way to combine syntactic feature and subtask dependencies is critical. We remove the syntax-enhanced aggregators in trigger and argument decoders in turn. The performance decrease indicates that the proposed syntax-enhanced aggregators contribute to both trigger extraction and argument extraction.

Effect of L. In order to show the effect of L value in contrastive learning, we train our method on FewFC dataset with varying dictionary sizes and draw curves in Figure 3 (a). The figure shows that with the increase of L value in contrastive learning, the performance of trigger classification and argument classification increases. This is because seeing more interactions of different event types could help the model learn more distinct syntactic features.

Representation visualization. In Figure 3 (b), we display the learned query representations in FewFC dataset by mapping them into two dimensional space via t-distributed stochastic neighbor embedding (t-SNE) (Hinton and Roweis, 2002). The data points with different colors indicate query graphs of different categories of event types. As we can observe, the query representations of different event types without contrastive learning mix together and exhibit random distribution. In contrast, after including type discrimination task with contrative learning, the same event types clustered. This verifies that contrastive learning leads to a better syntactic representation for each sentence.

Proceedings of the 22nd China National Conference on Computational Linguistics, pages 748-760, Harbin, China, August 3 - 5, 2023.

5 Related Work

5.1 Frameworks of Event Extraction

The frameworks of event extraction can be roughly categorized into three groups. Joint learning framework solves event extraction in a sequence labeling manner (Li et al., 2013; Nguyen et al., 2016; Sha et al., 2018a; Liu et al., 2018; Nguyen and Nguyen, 2019; Shen et al., 2020; Huang et al., 2020). They treat each token as the candidate of a trigger or an argument and tag it with types. However, joint learning has the disadvantage of solving sentences where one token could have more than one event types. Pipeline framework performs trigger extraction and argument extraction in separate stages (Yang et al., 2019; Wadden et al., 2019; Li et al., 2020a; Du and Cardie, 2020; Liu et al., 2020; Chen et al., 2020; Ma et al., 2022; Zhou et al., 2022). This framework could avoid the label conflict issue but it ignores the potential label dependencies in modeling and suffers from error propagation. The cascade decoding framework formulates triples to represent event types, triggers and arguments (Xu et al., 2020; Sheng et al., 2021; Yang et al., 2021). It jointly performs predictions for event triggers as well as arguments based on shared feature representations and learns the implicit dependencies of the triples. It could avoid label conflict and error propagation. Empirical results show it is an effective solution for event extraction. The cascade decoding framework is also effective in jointly extracting relations and entities from text (Zheng et al., 2017; Wei et al., 2020).

5.2 Syntax Modeling for Event Extraction

There are a number of studies that incorporate the syntactic structure of sentences into event extraction tasks. The early work (Chen et al., 2015) collected syntactic features from the dependency tree and fed them into a dynamic multi-pooling convolutional neural network for extracting events. Li et al.(Li et al., 2018) also utilized dependency-based embeddings to represent words semantically and syntactically and proposed a PMCNN for biomedical event extraction. Some studies tried to enhance the basic network with syntactic dependency, Sha et al.(Sha et al., 2018b) proposed a novel dependency bridge recurrent neural network and Zhang et al. (Zhang et al., 2018) transformed dependency trees into target-dependent trees. The follow-up studies (Nguyen and Grishman, 2018; Liu et al., 2018; Yan et al., 2019) employed graph convolutional network to encode the dependency tree and utilized it for predicting event types. More advanced neural networks are leveraged to model syntax in event extraction tasks. The gate mechanism and Transformer (Lai et al., 2020; Ahmad et al., 2021; Xie et al., 2021) have shown to be effective in encoding the graph information of dependency tree. (Li et al., 2021) utilized the relationships of event arguments based on a reinforcement learning and incremental learning. (Lu et al., 2021) designed a sequence-to-structure framework to uniformly models different subtasks of event extraction. However, some of them focus on detecting event types with syntax modeling which can be treated as a joint learning framework of event extraction, the others follow a pipeline framework of event extraction to enhance syntactic information. To fully make use of the cascade decoding framework, we propose our method based on the cascade decoding architecture, which captures the subtask dependencies and syntactic structure simultaneously.

6 Conclusions

In this paper, we improved cascade decoding with syntax-aware aggregator and contrastive learning for event extraction. We demonstrated the effectiveness of our proposed method on two datasets. The results showed that our method outperforms all baseline methods based on F1 score. Considering that many scenes have relatively high requirements for real-time performance, we will explore to optimize the computational complexity of the model and improving the universality of the model in the future.

Acknowledgements

This work was supported in part by ECNU Research Fund on Cultural Inheritance and Innovation (Grant No. 2022ECNU—WHCCYJ-31) and Shanghai Pujiang Talent Program (Project No. 22PJ1403000). We sincerely thank the anonymous reviewers for their valuable comments and feedback.

Proceedings of the 22nd China National Conference on Computational Linguistics, pages 748-760, Harbin, China, August 3 - 5, 2023.

References

Wasi Uddin Ahmad, Nanyun Peng, and Kai-Wei Chang. 2021. Gate: graph attention transformer encoder for cross-lingual relation and event extraction. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 12462–12470.

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)*, pages 167–176.

Yunmo Chen, Tongfei Chen, Seth Ebner, Aaron Steven White, and Benjamin Van Durme. 2020. Reading the manual: Event extraction as definition comprehension. In *Proceedings of the Fourth Workshop on Structured Prediction for NLP*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Xinya Du and Claire Cardie. 2020. Event extraction by answering (almost) natural questions. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 671–683.

Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. 2020. Momentum contrast for unsupervised visual representation learning. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9726–9735.

Dan Hendrycks and Kevin Gimpel. 2016. Bridging nonlinearities and stochastic regularizers with gaussian error linear units. *arXiv preprint arXiv:1606.08415*.

Geoffrey E Hinton and Sam Roweis. 2002. Stochastic neighbor embedding. In S. Becker, S. Thrun, and K. Obermayer, editors, *Advances in Neural Information Processing Systems*, volume 15.

Peixin Huang, Xiang Zhao, Ryuichi Takanobu, Zhen Tan, and Weidong Xiao. 2020. Joint event extraction with hierarchical policy network. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 2653–2664.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *ICLR*.

Thomas N. Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations (ICLR)*.

Viet Dac Lai, Tuan Ngo Nguyen, and Thien Huu Nguyen. 2020. Event detection: Gate diversity and syntactic importance scores for graph convolution neural networks. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 5405–5411.

Viet Lai, Minh Van Nguyen, Heidi Kaufman, and Thien Huu Nguyen. 2021. Event extraction from historical texts: A new dataset for black rebellions. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 2390–2400.

Dongkyu Lee, Zhiliang Tian, Lanqing Xue, and Nevin L. Zhang. 2021. Enhancing content preservation in text style transfer using reverse attention and conditional layer normalization. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 93–102.

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 (Volume 1: Long Papers)*, pages 73–82.

Lishuang Li, Yang Liu, and Meiyue Qin. 2018. Extracting biomedical events with parallel multi-pooling convolutional neural networks. *IEEE/ACM transactions on computational biology and bioinformatics*, 17(2):599–607.

Fayuan Li, Weihua Peng, Yuguang Chen, Quan Wang, Lu Pan, Yajuan Lyu, and Yong Zhu. 2020a. Event extraction as multi-turn question answering. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 829–838.

Xinyu Li, Fayuan Li, Lu Pan, Yuguang Chen, Weihua Peng, Quan Wang, Yajuan Lyu, and Yong Zhu. 2020b. Duee: a large-scale dataset for chinese event extraction in real-world scenarios. In *CCF International Conference on Natural Language Processing and Chinese Computing*, pages 534–545.

Proceedings of the 22nd China National Conference on Computational Linguistics, pages 748-760, Harbin, China, August 3 - 5, 2023.

Qian Li, Hao Peng, Jianxin Li, Jia Wu, Yuanxing Ning, Lihong Wang, S Yu Philip, and Zheng Wang. 2021. Reinforcement learning-based dialogue guided event extraction to exploit argument relations. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 30:520–533.

Qian Li, Jianxin Li, Jiawei Sheng, Shiyao Cui, Jia Wu, Yiming Hei, Hao Peng, Shu Guo, Lihong Wang, Amin Beheshti, et al. 2022. A survey on deep learning event extraction: Approaches and applications. *IEEE Transactions on Neural Networks and Learning Systems*.

Xiao Liu, Zhunchen Luo, and Heyan Huang. 2018. Jointly multiple events extraction via attention-based graph information aggregation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1247–1256.

Jian Liu, Yubo Chen, Kang Liu, Wei Bi, and Xiaojiang Liu. 2020. Event extraction as machine reading comprehension. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1641–1651.

Yaojie Lu, Hongyu Lin, Jin Xu, Xianpei Han, Jialong Tang, Annan Li, Le Sun, Meng Liao, and Shaoyi Chen. 2021. Text2event: Controllable sequence-to-structure generation for end-to-end event extraction. *arXiv preprint arXiv:2106.09232*.

Yubo Ma, Zehao Wang, Yixin Cao, Mukai Li, Meiqi Chen, Kun Wang, and Jing Shao. 2022. Prompt for extraction? PAIE: Prompting argument interaction for event argument extraction. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*.

Thien Huu Nguyen and Ralph Grishman. 2018. Graph convolutional networks with argument-aware pooling for event detection. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence and Thirtieth Innovative Applications of Artificial Intelligence Conference and Eighth AAAI Symposium on Educational Advances in Artificial Intelligence*, pages 5900–5907.

Trung Minh Nguyen and Thien Huu Nguyen. 2019. One for all: Neural joint modeling of entities and events. *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 6851–6858.

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.

Michael Schlichtkrull, Thomas N Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling. 2018. Modeling relational data with graph convolutional networks. In *European semantic web conference*, pages 593–607. Springer.

Lei Sha, Feng Qian, Baobao Chang, and Zhifang Sui. 2018a. Jointly extracting event triggers and arguments by dependency-bridge rnn and tensor-based argument interaction. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence and Thirtieth Innovative Applications of Artificial Intelligence Conference and Eighth AAAI Symposium on Educational Advances in Artificial Intelligence*.

Lei Sha, Feng Qian, Baobao Chang, and Zhifang Sui. 2018b. Jointly extracting event triggers and arguments by dependency-bridge rnn and tensor-based argument interaction. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32.

Shirong Shen, Guilin Qi, Zhen Li, Sheng Bi, and Lusheng Wang. 2020. Hierarchical Chinese legal event extraction via pedal attention mechanism. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 100–113, December.

Jiawei Sheng, Shu Guo, Bowen Yu, Qian Li, Yiming Hei, Lihong Wang, Tingwen Liu, and Hongbo Xu. 2021. CasEE: A joint learning framework with cascade decoding for overlapping event extraction. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 164–174.

David Wadden, Ulme Wennberg, Yi Luan, and Hannaneh Hajishirzi. 2019. Entity, relation, and event extraction with contextualized span representations. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5784–5789, November.

Ziqi Wang, Xiaozhi Wang, Xu Han, Yankai Lin, Lei Hou, Zhiyuan Liu, Peng Li, Juanzi Li, and Jie Zhou. 2021. CLEVE: Contrastive Pre-training for Event Extraction. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 6283–6297.

Zhepei Wei, Jianlin Su, Yue Wang, Yuan Tian, and Yi Chang. 2020. A novel cascade binary tagging framework for relational triple extraction. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1476–1488.

Jianye Xie, Haotong Sun, Junsheng Zhou, Weiguang Qu, and Xinyu Dai. 2021. Event detection as graph parsing. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, page 1630–1640.

Nuo Xu, Haihua Xie, and Dongyan Zhao. 2020. A novel joint framework for multiple Chinese events extraction. In *Proceedings of the 19th Chinese National Conference on Computational Linguistics*, pages 950–961.

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

Sen Yang, Dawei Feng, Linbo Qiao, Zhigang Kan, and Dongsheng Li. 2019. Exploring pre-trained language models for event extraction and generation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5284–5294.

Hang Yang, Dianbo Sui, Yubo Chen, Kang Liu, Jun Zhao, and Taifeng Wang. 2021. Document-level event extraction via parallel prediction networks. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 6298–6308.

Wenbo Zhang, Xiao Ding, and Ting Liu. 2018. Learning target-dependent sentence representations for chinese event detection. In *China Conference on Information Retrieval*, pages 251–262. Springer.

Zhengyan Zhang, Xu Han, Zhiyuan Liu, Xin Jiang, Maosong Sun, and Qun Liu. 2019. ERNIE: Enhanced language representation with informative entities. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1441–1451.

Suncong Zheng, Feng Wang, Hongyun Bao, Yuexing Hao, Peng Zhou, and Bo Xu. 2017. Joint extraction of entities and relations based on a novel tagging scheme. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1227–1236.

Shun Zheng, Wei Cao, Wei Xu, and Jiang Bian. 2019. Doc2EDAG: An end-to-end document-level framework for Chinese financial event extraction. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 337–346.

Yang Zhou, Yubo Chen, Jun Zhao, Yin Wu, Jiexin Xu, and Jinlong Li. 2021. What the role is vs. what plays the role: Semi-supervised event argument extraction via dual question answering. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 14638–14646.

Jie Zhou, Qi Zhang, Qin Chen, Qi Zhang, Liang He, and Xuanjing Huang. 2022. A multi-format transfer learning model for event argument extraction via variational information bottleneck. In *Proceedings of the 29th International Conference on Computational Linguistics*.

Proceedings of the 22nd China National Conference on Computational Linguistics, pages 748-760, Harbin, China, August 3 - 5, 2023.