CasEE: A Joint Learning Framework with Cascade Decoding for Overlapping Event Extraction

Jiawei Sheng[1,2], **Shu Guo**[3], **Bowen Yu**[1,2], **Qian Li**[4], **Yiming Hei**[5],
Lihong Wang[3], **Tingwen Liu**[1,2] and **Hongbo Xu**[1,2]
[1]Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China
[2]School of Cyber Security, University of Chinese Academy of Sciences, Beijing, China
[3]National Computer Network Emergency Response Technical Team/Coordination Center of China
[4]School of Computer Science and Engineering, Beihang University, Beijing, China
[5]School of Cyber Science and Technology, Beihang University, Beijing, China
`shengjiawei@iie.ac.cn, guoshu@cert.org.cn, wlh@isc.org.cn`

Abstract

Event extraction (EE) is a crucial information extraction task that aims to extract event information in texts. Most existing methods assume that events appear in sentences without overlaps, which are not applicable to the complicated overlapping event extraction. This work systematically studies the realistic event overlapping problem, where a word may serve as triggers with several types or arguments with different roles. To tackle the above problem, we propose a novel joint learning framework with cascade decoding for overlapping event extraction, termed as CasEE. Particularly, CasEE sequentially performs type detection, trigger extraction and argument extraction, where the overlapped targets are extracted separately conditioned on the specific former prediction. All the subtasks are jointly learned in a framework to capture dependencies among the subtasks. The evaluation on a public event extraction benchmark FewFC demonstrates that CasEE[1] achieves significant improvements on overlapping event extraction over previous competitive methods.

1 Introduction

Event Extraction (EE) is an important yet challenging task in natural language understanding. Given a sentence, an event extraction system ought to identify event types, triggers and arguments appearing in the sentence. As an example, Figure 1(b) presents an event mention of type `Share Reduction`, triggered by "reduced". There are several arguments, such as "Fuda Industry" playing the `subject` role in the event.

However, events often appear in sentences complicatedly, where the triggers and arguments may have overlaps in a sentence. This paper focuses

[1]The source code is available at `https://github.com/JiaweiSheng/CasEE`.

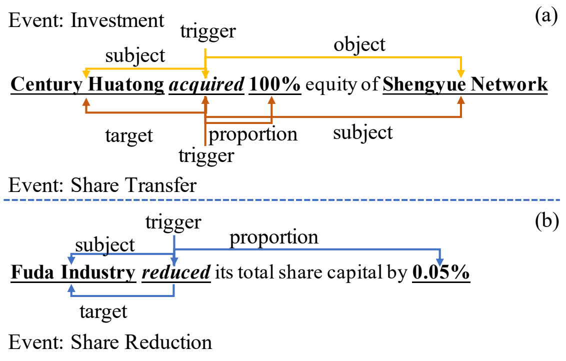

Figure 1: Examples of event overlapping problem: (a) Events with overlapped triggers and arguments; (b) An event with an overlapped argument in several roles.

on a challenging and realistic problem in EE: *overlapping event extraction*. Generally, we categorize all the overlapping cases into three patterns: 1) A word may serve as triggers with different event types across several events. Figure 1(a) shows the token "acquired" triggers an `Investment` event and a `Share Transfer` event at the same time. 2) A word may serve as arguments with different roles across several events. Figure 1(a) shows "Shengyue Network" plays an `object` role in the `Investment` event and a `subject` role in the `Share Transfer` event. 3) A word may serve as arguments playing different roles in one event. Figure 1(b) shows that "Fuda Industry" plays a `subject` role and a `target` role in an event. For simplicity, we call pattern 1) as *overlapped trigger problem*, and both pattern 2) and 3) as *overlapped argument problem* in the following sections. There are about 13.5% / 21.7% sentences having overlapped trigger/argument problems in the Chinese financial event extraction dataset, FewFC (Zhou et al., 2021).

Most existing EE studies assume that events appear in sentences without overlaps, which are not applicable to the complicated overlapping scenarios. Typically, current EE studies can be roughly categorized into two groups: 1) Traditional joint methods (Nguyen et al., 2016; Liu et al., 2018; Nguyen and Nguyen, 2019), which simultaneously extract triggers and arguments by a unified decoder labeling the sentence only once. However, they fail in extracting overlapped targets due to label conflicts, where a token may have several typed labels but only one label can be assigned. 2) Pipeline methods (Chen et al., 2015; Yang et al., 2019; Du and Cardie, 2020b), which sequentially extract triggers and arguments in separate stages. Yang et al. (2019) attempts to tackle the overlapped argument problem in the pipeline manner, but overlooks the overlapped trigger problem. Nevertheless, the pipeline methods neglect the feature-level dependencies between the trigger and arguments, and suffer from error propagation. In our knowledge, existing researches in EE neglect overlapping problems or only focus on one overlapping problem. Few researches simultaneously solve all the three mentioned overlapping patterns.

To address the above issues, we propose CasEE, a joint learning framework with Cascade decoding for overlapping Event Extraction. Specifically, CasEE realizes event extraction with a shared textual encoder and three decoders for type detection, trigger extraction and argument extraction. To extract overlapped targets across events, CasEE sequentially decodes the three subtasks, conducting trigger extraction and argument extraction according to the former predictions. Such a cascade decoding strategy extracts event elements according to the different conditions, so that the overlapped targets can be extracted in separate phases. A condition fusion function is designed to explicitly model the dependencies between adjacent subtasks. All the subtask decoders are jointly learned to further build connections among subtasks, which refines the shared textual encoder with feature-level interactions among downstream subtasks.

The contributions of this paper are three-fold:

(1) We systematically investigate the overlapping problems in EE, and categorize them into three patterns. To the best of our knowledge, this paper is among the first to simultaneously tackle all the three overlapping patterns.

(2) We propose CasEE, a novel joint learning framework with cascade decoding, to simultaneously solve all the three overlapping patterns.

(3) We conduct experiments on a public Chinese financial event extraction benchmark, FewFC. Experimental results reveal that CasEE achieves significant improvements on overlapping event extraction over existing competitive methods.

2 Related Work

Current EE research can be roughly categorized into two groups: 1) Traditional joint methods (Li et al., 2013; Nguyen et al., 2016; Nguyen and Nguyen, 2019; Liu et al., 2018; Sha et al., 2018) that perform trigger extraction and argument extraction simultaneously. They solve the task in a sequence labeling manner, and extract triggers and arguments by tagging the sentence only once. However, these methods fail in solving overlapping event extraction since the overlapping tokens would cause label conflicts when forced to have more than one label. 2) Pipeline methods (Chen et al., 2015; Yang et al., 2019; Wadden et al., 2019; Li et al., 2020; Du and Cardie, 2020b; Liu et al., 2020; Chen et al., 2020) that perform trigger extraction and argument extraction in separate stages. Though pipeline methods have the potential capacity to solve overlapping EE, they usually lack explicit dependencies between triggers and arguments, and suffer from error propagation. Among the researches, Yang et al. (2019) and Xu et al. (2020) solve the overlapped argument problem, but overlook the overlapped trigger problem, thus can not discern correct triggers for argument extraction. All the above methods can not simultaneously solve all the overlapping patterns in event extraction.

The overlapping problem has also been explored in other information extraction tasks outside event extraction. Luo and Zhao (2020) tackles nested named entity recognization with bipartite flat-graph networks. Zeng et al. (2018) tackles overlapped relational triple extraction by applying a sequence-to-sequence paradigm with a copy mechanism. Wei et al. (2020) and Yu et al. (2020) extract overlapped relational triples with a novel cascade tagging strategy, which inspire us to solve overlapping event extraction in the cascade decoding paradigm. Wang et al. (2020) further discusses the propagation error in cascade decoding. All the above researches are proposed for other tasks, which can not be directly transferred for overlapping event extraction due to the complicated event extraction definition.

3 Our Approach

Given an input sentence, the goal of EE is to identify triggers with their event types and arguments

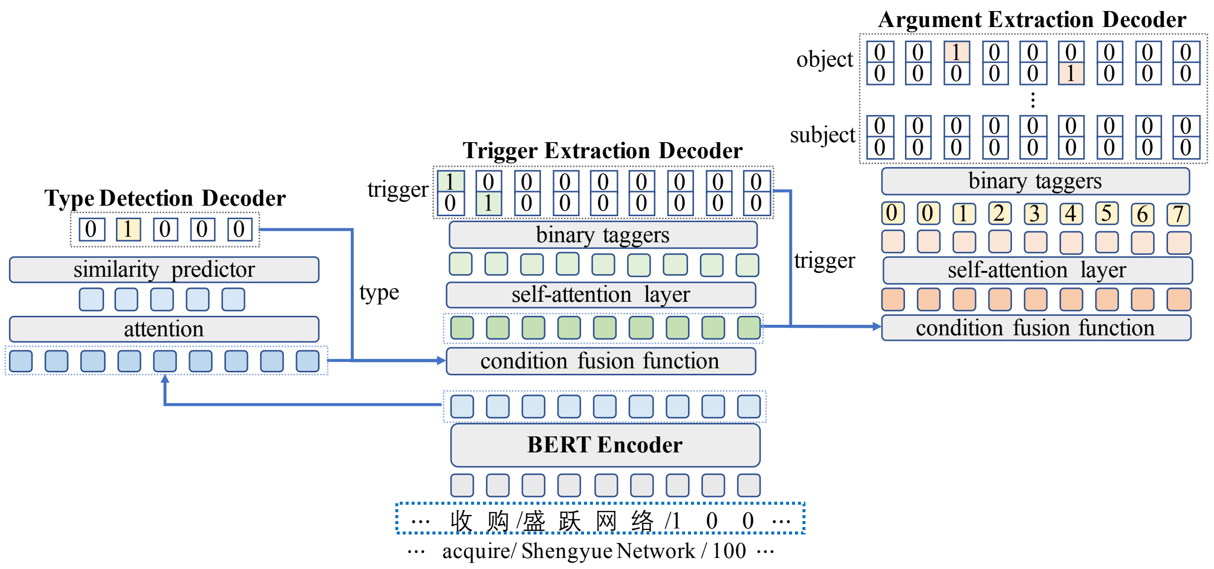
Argument Extraction Decoder
object
subject
binary taggers
self-attention layer
condition fusion function
Trigger Extraction Decoder
trigger
binary taggers
self-attention layer
condition fusion function
Type Detection Decoder
similarity predictor
attention
type
trigger
BERT Encoder
··· 收 购 /盛 跃 网 络 /1 0 0 ···
··· acquire/ Shengyue Network / 100 ···

Figure 2: The overview of our proposed approach, CasEE, which contains a shared BERT encoder, a type detection decoder, a trigger extraction decoder and an argument extraction decoder.

with their corresponding roles, where triggers and arguments may overlap on some tokens. To tackle this problem, we propose a training objective at the event level. Formally, according to the pre-defined event schema, we have an event type set $\mathcal{C}$ and an argument role set $\mathcal{R}$. The overall goal is to predict all events in gold set $\mathcal{E}_x$ of the sentence x. We aim to maximize the joint likelihood of training data $\mathcal{D}$:

$$\prod_{x\in\mathcal{D}}\Big[\prod_{(c,t,a_r)\in\mathcal{E}_x} p((c,t,a_r))|x)\Big]$$
$$=\prod_{x\in\mathcal{D}}\Big[\prod_{c\in\mathcal{C}_x} p(c|x)\prod_{t\in\mathcal{T}_{x,c}} p(t|x,c)\prod_{a_r\in\mathcal{A}_{x,c,t}} p(a_r|x,c,t)\Big] \quad (1)$$

where $\mathcal{C}_x$ denotes the set of types occurring in x, $\mathcal{T}_{x,c}$ denotes the trigger set of type c, and $\mathcal{A}_{x,c,t}$ denotes the argument set of type c and trigger t. Note that each c is a type in $\mathcal{C}$, each t is a trigger word, and each $a_r \in \mathcal{A}_x$ is an argument word corresponding to its own role $r \in \mathcal{R}$. Eq. (1) exploits the fact of dependencies among the type, trigger and argument. Actually, it motivates us to learn a type detection decoder $p(c|x)$ to detect event types occurring in the sentence, a trigger extraction decoder $p(t|x,c)$ to extract triggers of type c, and an argument extraction decoder $p(a_r|x,c,t)$ to extract role-specific arguments with type c and trigger t.

Such a task decomposition solves all the event overlapping patterns claimed in the Introduction. Specifically, we first detect event types occurring in the sentence. When extracting triggers, we only predict the triggers with a specific type, thus the triggers overlapped across several events will be predicted in separate phases. Similarly, when extracting arguments, we predict the arguments with a specific type and trigger, thus the arguments overlapped across several events will also be predicted in separate phases. Since we adopt role-specific taggers in argument extraction, the overlapped arguments having several roles in an event can be predicted separately with specific taggers. All the predictions in type detection, trigger extraction and argument extraction form the final prediction.

Figure 2 demonstrates the details of CasEE. CasEE adopts a shared BERT encoder to capture textual features, and three decoders for type detection, trigger extraction and argument extraction. Since all subtasks are jointly learned in contrast to previous pipeline methods (Yang et al., 2019; Li et al., 2020), CasEE could capture feature-level dependencies among subtasks. For prediction, CasEE sequentially predicts event types, triggers and arguments in the cascade decoding process.

3.1 BERT Encoder

To capture the feature-level dependencies among subtasks, we share the textual representations of each sentence. As BERT has shown performance improvements across multiple NLP tasks, we adopt BERT (Devlin et al., 2019) as our textual encoder. BERT is a bi-directional language representation model based on transformer architecture (Vaswani et al., 2017), which generates textual representations conditioned on token context and remains rich textual information. Formally, the sentence with N tokens is denoted as $x = w_1, w_2, ..., w_N$. We input the tokens into BERT, and then obtain the hidden states $\mathbf{H} = \mathbf{h}_1, \mathbf{h}_2, ..., \mathbf{h}_N$ as the token representations for the following downstream subtasks.

3.2 Type Detection Decoder

Since we tackle the overlapped trigger problem by extracting triggers conditioned on the type predictions, we devise a type detection decoder to predict event types. Inspired by event detection without triggers (Liu et al., 2019), we adopt attention mechanism to detect event types, capturing the most relative context for each possible type. Specifically, we randomly initialize embedding matrix $\mathbf{C} \in \mathbb{R}^{|\mathcal{C}| \times d}$ as the type embeddings. We define a similarity function δ to measure the relevance between the candidate type $\mathbf{c} \in \mathbf{C}$ and each token representation $\mathbf{h}_i$. To fully capture the similarity information in different aspects, we achieve δ with an expressive learnable function. According to the relevance scores, we obtain the sentence representation $\mathbf{s}_c$ adaptive to the type. The details are as follows:

$$\delta(\mathbf{c}, \mathbf{h}_i) = \mathbf{v}^\intercal \tanh(\mathbf{W}[\mathbf{c}; \mathbf{h}_i; |\mathbf{c} - \mathbf{h}_i|; \mathbf{c} \odot \mathbf{h}_i])$$
$$\mathbf{s}_c = \sum_{i=1}^{N} \frac{\exp(\delta(\mathbf{c}, \mathbf{h}_i))}{\sum_{j=1}^{N} \exp(\delta(\mathbf{c}, \mathbf{h}_j))} \mathbf{h}_i \tag{2}$$

where $\mathbf{W} \in \mathbb{R}^{4d \times 4d}$ and $\mathbf{v} \in \mathbb{R}^{4d \times 1}$ are learnable parameters, $|\cdot|$ is an absolute value operator, $\odot$ is the element-wise production, and $[\cdot;\cdot]$ denotes the concatenation of representations.

Finally, we predict event types by measuring the similarity of the adaptive sentence representation $\mathbf{s}_c$ and the type embedding $\mathbf{c}$ with the same similarity function δ. Then, the predicted probability of each event type c occurring in the sentence is:

$$\hat{c} = p(c|x) = \sigma(\delta(\mathbf{c}, \mathbf{s}_c)) \tag{3}$$

where σ denotes sigmoid function. We select the event type with $\hat{c} > \xi_1$ as results, where $\xi_1 \in [0, 1]$ is a scalar threshold. All predicted types in sentence x form event type set $\mathcal{C}_x$. The decoder learnable parameter $\theta_{td} \triangleq \{\mathbf{W}, \mathbf{v}, \mathbf{C}\}$.

3.3 Trigger Extraction Decoder

To discern overlapped triggers with several types, we extract triggers conditioned on a specific type $c \in \mathcal{C}_x$. This decoder contains a condition fusion function, a self-attention layer, and a pair of binary taggers for triggers.

To model the conditional dependency between type detection and trigger extraction, we devise a condition fusion function ϕ to integrate condition information into textual representation. Specifically, we obtain the conditional token representation $\mathbf{g}_i^c$ by integrating the type embedding $\mathbf{c}$ into the token representation $\mathbf{h}_i$ as:

$$\mathbf{g}_i^c = \phi(\mathbf{c}, \mathbf{h}_i) \tag{4}$$

Actually, ϕ can be achieved by concatenation, addition operator or gate mechanism. To fully generate conditional representations in the statistical aspect, we introduce an effective and general mechanism, conditional layer normalization (CLN) (Su, 2019; Yu et al., 2021), to achieve ϕ. CLN is mostly based on the well-known layer normalization (Ba et al., 2016), but can dynamically generate gain γ and bias β based on the condition information. Given a condition embedding $\mathbf{c}$ and a token representation $\mathbf{h}_i$, CLN is formulated as:

$$\begin{aligned} \mathrm{CLN}(\mathbf{c}, \mathbf{h}_i) &= \gamma_c \odot (\frac{\mathbf{h}_i - \mu}{\sigma}) + \beta_c, \\ \gamma_c &= \mathbf{W}_\gamma \mathbf{c} + \mathbf{b}_\gamma, \beta_c = \mathbf{W}_\beta \mathbf{c} + \mathbf{b}_\beta \end{aligned} \tag{5}$$

where $\mu \in \mathbb{R}$ and $\sigma \in \mathbb{R}$ are the mean and standard variance taken across the elements of $\mathbf{h}_i$, and $\gamma_c \in \mathbb{R}^d$ and $\beta_c \in \mathbb{R}^d$ are the conditional gain and bias, respectively. In this way, the given condition representation is encoded into the gain and bias, and then integrated into contextual representations.

To further refine representations for trigger extraction, we adopt a self-attention layer over the conditional token representations. Formally, the refined token representations are derived as:

$$\mathbf{Z}^c = \mathrm{SelfAttention}(\mathbf{G}^c) \tag{6}$$

where $\mathbf{G}^c$ is the representation matrix composed of $\mathbf{g}_i^c$. For details of the self-attention layer, please refer to Vaswani et al. (2017).

To predict triggers, we devise a pair of binary taggers. For each token w_i, we predict whether it corresponds to a start or end position of a trigger as:

$$\begin{aligned} \hat{t}_i^{sc} &= p(t_s|w_i, c) = \sigma(\mathbf{w}_{t_s}^\intercal \mathbf{z}_i^c + b_{t_s}) \\ \hat{t}_i^{ec} &= p(t_e|w_i, c) = \sigma(\mathbf{w}_{t_e}^\intercal \mathbf{z}_i^c + b_{t_e}) \end{aligned} \tag{7}$$

where σ denotes sigmoid function, and $\mathbf{z}_i^c$ denotes the i-th token representation in $\mathbf{Z}^c$. We select tokens with $\hat{t}_i^{sc} > \xi_2$ as the start positions, and those with $\hat{t}_i^{ec} > \xi_3$ as end positions, where $\xi_2, \xi_3 \in [0, 1]$ are scalar thresholds. To obtain the trigger word t, we enumerate all the start positions and search the nearest following end position in the sentence, and the tokens between the start and end position form an entire trigger. In this way, the overlapped triggers can be extracted separately according to the type in separate phases. All the predicted trigger t of type c in sentence s forms the set $\mathcal{T}_{c,s}$. The decoder parameter θ_{te} includes all the parameters in the condition fusion function, the self-attention layer and the trigger taggers.

3.4 Argument Extraction Decoder

To tackle the overlapped argument problem, we extract role-specific arguments conditioned on both the specific event type $c \in \mathcal{C}_s$ and event trigger $t \in \mathcal{T}_{c,s}$. This decoder also contains a condition fusion function, a self-attention layer, and a group of role-specific binary tagger pairs for arguments.

We further integrate the trigger information into the typed textual representation $\mathbf{g}_i^c$ in Eq. (4) with function ϕ achieved by CLN. Here we take the average of the start and end position token representations of t as the trigger embedding. We also adopt a self-attention layer to derive the refined textual representations $\mathbf{Z}^{ct'}$. To be aware of the trigger position, we adopt the relative position embedding as used in Chen et al. (2015), which indicates the relative distance from current token to the trigger boundary token. Finally, the token representations $\mathbf{Z}^{ct}$ for argument extraction are derived as:

$$\mathbf{Z}^{ct} = [\mathbf{Z}^{ct'}; \mathbf{P}] \tag{8}$$

where $\mathbf{P} \in \mathbb{R}^{N \times d_p}$ is the relative position embeddings, d_p is the dimension, and $[\cdot;\cdot]$ denotes the concatenation of representations.

To predict arguments in roles, we devise a group of role-specific tagger pairs. For each token w_i, we predict whether it corresponds to a start or end position of an argument of the role $r \in \mathcal{R}$ as:

$$\begin{aligned}\hat{r}_i^{sct} &= p(a_r^s | w_i, c, t) = \mathrm{I}(r, c)\sigma(\mathbf{w}_{r_s}^\intercal \mathbf{z}_i^{ct} + b_{r_s})\\ \hat{r}_i^{ect} &= p(a_r^e | w_i, c, t) = \mathrm{I}(r, c)\sigma(\mathbf{w}_{r_e}^\intercal \mathbf{z}_i^{ct} + b_{r_e})\end{aligned} \tag{9}$$

where σ denotes sigmoid function, and $\mathbf{z}_i^{ct}$ denotes the i-th token representation in $\mathbf{Z}^{ct}$. Since not all roles belonging to the specific type c, we adopt an indicator function $\mathrm{I}(r, c)$ to indicate whether the role r belongs to the type c according to the pre-defined event scheme. To make the indicator function derivable, we parameterize $\mathrm{I}(r, c)$ to learn with the model parameters. Specifically, given the type embedding $\mathbf{c} \in \mathcal{C}$, we build the connection between the type and roles as:

$$\mathrm{I}(r, c) = \sigma(\mathbf{w}_r^\intercal \mathbf{c} + b_r) \tag{10}$$

where σ denotes sigmoid function, $\mathbf{w}_r, b_r$ are parameters associated with the role r. For each role r, we select tokens with $\hat{r}_i^{sct} > \xi_4$ as the start positions, and those with $\hat{r}_i^{ect} > \xi_5$ as end positions, where $\xi_4, \xi_5 \in [0, 1]$ are scalar thresholds. To obtain the argument word a with role r, we enumerate all the start positions and search the nearest following end position in the sentence, and the tokens between the start and end position form an entire argument. In this way, the overlapped arguments can be extracted separately according to the different types and triggers with role-specific taggers. All the predicted argument a_r with type c and trigger t in sentence x forms the set $\mathcal{A}_{t,c,x}$. The decoder parameter θ_{ae} includes the type embedding matrix $\mathbf{C}$, and all parameters in the condition fusion function, the self-attention layer and the argument taggers.

3.5 Model Training

To train the model, we take log of Eq (1), and the overall objective $\mathcal{J}(\Theta)$ is deployed as:

$$\begin{aligned}&\sum_{x \in \mathcal{D}} \Big[\sum_{c \in \mathcal{C}_x} \log p_{\theta_1}(c|x) + \\ &\quad \sum_{t \in \mathcal{T}_{x,c}} \log p_{\theta_2}(t|x, c) + \sum_{a_r \in \mathcal{A}_{x,c,t}} \log p_{\theta_3}(a_r|x, c, t) \Big]\end{aligned} \tag{11}$$

where $\Theta \triangleq \{\theta_1, \theta_2, \theta_3\}$; $p_{\theta_1}(c|x)$, $p_{\theta_2}(t|x, c)$, and $p_{\theta_3}(a_r|x, c, t)$ for the subtasks are defined as:

$$\begin{aligned}p_{\theta_1}(c|x) &= (\hat{c})^{\bar{c}}(1 - \hat{c})^{(1-\bar{c})}\\ p_{\theta_2}(t|x, c) &= \prod_{z \in \{s,e\}} \prod_{i=1}^{N} (\hat{t}_i^{zc})^{\bar{t}_i^{zc}} (1 - \hat{t}_i^{zc})^{(1-\bar{t}_i^{zc})}\\ p_{\theta_3}(a_r|x, c, t) &= \prod_{r \in \mathcal{R}} \prod_{z \in \{s,e\}} \prod_{i=1}^{N} (\hat{r}_i^{zct})^{\bar{r}_i^{zct}} (1 - \hat{r}_i^{zct})^{(1-\bar{r}_i^{zct})}\end{aligned} \tag{12}$$

where $\hat{c}$, $\hat{t}_i^{sc}$, $\hat{t}_i^{ec}$, $\hat{r}_i^{sct}$, $\hat{r}_i^{ect}$ are the predicted probabilities in Eq (3), Eq (7), Eq (9), and $\bar{c}$, $\bar{t}_i^{sc}$, $\bar{t}_i^{ec}$, $\bar{r}_i^{sct}$, $\bar{r}_i^{ect}$ are the true 0/1 labels of the training data, respectively. $\theta_1 \triangleq \{\theta_{bert}, \theta_{td}\}$, $\theta_2 \triangleq \{\theta_{bert}, \theta_{te}\}$, $\theta_3 \triangleq \{\theta_{bert}, \theta_{ae}\}$, where $\theta_{bert}, \theta_{td}, \theta_{te}, \theta_{ae}$ denote parameters in BERT, type detection, trigger extraction and argument extraction, respectively. We train the model by maximizing $\mathcal{J}(\Theta)$ through Adam stochastic gradient descent (Kingma and Ba, 2015) over the shuffled mini-batches.

	#Overlap	#Normal	#Sentence	#Event
Training	1,560	5,625	7,185	10,277
Validation	205	694	899	1,281
Testing	210	688	898	1,332
All	1,975	7,007	8,982	12,890

Table 1: Statistics of the dataset. Each column denotes the number of the sentences with overlapped elements, the sentences without overlapped elements, all the sentences and all the events.

4 Experiments

In this section, we conduct experiments to evaluate the performance of CasEE.

4.1 Dataset and Evaluation Metric

We conduct experiments[2] on a Chinese financial event extraction benchmark FewFC (Zhou et al., 2021). We split data with 8:1:1 for training/validation/testing. Table 1 shows more details.

For evaluation, we follow the traditional evaluation metrics (Chen et al., 2015; Du and Cardie, 2020b): 1) Trigger Identification (TI): A trigger is correctly identified if the predicted trigger span matches with a golden span; 2) Trigger Classification (TC): A trigger is correctly classified if it is correctly identified and assigned to the correct type; 3) Argument Identification (AI): An argument is correctly identified if its event type is correctly recognized and the predicted argument span matches with a golden span; 4) Argument Classification (AC): an argument is correctly classified if it is correctly identified and the predicted role matches with a golden role. We report Precision (P), Recall (R) and F measure (F1) for each of the four metrics.

4.2 Comparision Methods

Though various models have recently been developed for EE, few researches are investigated to solve overlapping event extraction. We attempt to develop the following baselines based on current solutions. For the realistic consideration, no candidate entities are previously known for EE.

Joint sequence labeling methods. This kind of method formulates event extraction into a sequence labeling task. **BERT-softmax** (Devlin et al., 2019) adopts BERT to learn textual representations and uses hidden states for classifying event triggers and arguments. **BERT-CRF** adopts conditional random field (CRF) to capture label dependencies, which is adopted in (Du and Cardie, 2020a) for document-level event extraction. **BERT-CRF-joint** borrows idea from joint extraction of entity and relation (Zheng et al., 2017), which adopts joint labels of the type and role as `B/I/O-type-role`. All the above methods can not solve the overlapping problem due to label conflicts.

Pipelined event extraction methods. This kind of method solves event extraction with a pipeline manner. **PLMEE** (Yang et al., 2019) solves overlapped argument problem by extracting role-specific arguments according to the trigger. Motivated by current Machine Reading Comprehension (MRC) based EE studies (Li et al., 2020; Du and Cardie, 2020b; Liu et al., 2020; Chen et al., 2020), we train multiple MRC BERTs for overlapping event extraction. We extend MQAEE (Li et al., 2020) for multi-span extraction and re-assemble the following methods[3] to consider conditions in EE: 1) The method first predicts types, and then predicts overlapped triggers/arguments according to the type, termed as **MQAEE-1**. 2) The method first predicts overlapped triggers with types, and then predicts overlapped arguments according to the typed triggers, termed as **MQAEE-2**. 3) The method sequentially predicts types, predicts overlapped triggers according to the type, and predicts overlapped arguments according to the type and trigger, termed as **MQAEE-3**. All the above pipeline methods could solve (or partly solve) overlapping event extraction.

4.3 Implementation Details

We adopt source code for PLMEE with its best hyper-parameters reported in the original literature. To achieve other baselines, we implement the code based on the Transformers library (Wolf et al., 2020). For all the methods, we adopt Chinese BERT-Base model[4] as the textual encoder, which has 12 layers, 768 hidden units and 12 attention heads. We use the same value for the common hyper-parameters among the methods, including the optimizer, learning rate, batch size and epoch. For all the hyper-parameters, we adopt grid search

[2]Though ACE 2005 dataset is usually used to evaluate traditional EE models, we observe that it contains a low proportion of sentences with overlapped argument problem (nearly 10% reported in Yang et al. (2019)), and doesn't exist sentences with overlapped trigger problem.

[3]For more details, please refer to the Appendix B.

[4]`https://huggingface.co/bert-base-chinese`

	TI(%)			TC(%)			AI(%)			AC(%)		
	P	R	F1	P	R	F1	P	R	F1	P	R	F1
BERT-softmax	89.8	79.0	84.0	80.2	61.8	69.8	74.6	62.8	68.2	72.5	60.2	65.8
BERT-CRF	**90.8**	80.8	85.5	**81.7**	63.6	71.5	75.1	64.3	69.3	72.9	61.8	66.9
BERT-CRF-joint	89.5	79.8	84.4	80.7	63.0	70.8	**76.1**	63.5	69.2	**74.2**	61.2	67.1
PLMEE	83.7	85.8	84.7	75.6	74.5	75.1	74.3	67.3	70.6	72.5	65.5	68.8
MQAEE-1	90.1	85.5	87.7	77.3	76.0	76.6	62.9	71.5	66.9	51.7	70.4	59.6
MQAEE-2	89.1	85.5	87.4	79.7	76.1	77.8	70.3	68.3	69.3	68.2	66.5	67.3
MQAEE-3	88.3	86.1	87.2	75.8	76.5	76.2	69.0	67.9	68.5	67.2	65.9	66.5
CasEE	89.4	**87.7**	**88.6**	77.9	**78.5**	**78.2**	72.8	**73.1**	**72.9**	71.3	**71.5**	**71.4**

Table 2: Results of event extraction on FewFC dataset, where TI, TC, AI, AC denote trigger identification, trigger classification, argument identification and argument classification, respectively.

strategy. We train all the methods with an Adam weight decay optimizer. The initial learning rate is tuned in $[1e^{-5}, 5e^{-5}]$ for BERT parameters and $[1e^{-4}, 3e^{-4}]$ for other parameters. The warming up proportion for learning rate is 10%, and the max training epoch is set to 20. The batch size is set to 8. For CasEE, the dimension d_p of the relative position embedding is tuned in $\{16, 32, 64\}$. To avoid overfitting, we apply dropout to BERT hidden states with the rate tuned in $[0, 1]$. Besides, the thresholds $\xi_1, \xi_2, \xi_3, \xi_4, \xi_5$ for prediction are tuned in $[0, 1]$. We select the best model leading to the highest performance on the validation data. The optimal hyper-parameter settings are tuned by grid search, listed in the Appendix A.

4.4 Main Results

The performance of all methods on the FewFC dataset is shown in 2. The table reveals that:

(1) Compared to the joint sequence labeling methods, CasEE achieves better performance on the F1 score. Specifically, CasEE achieves improvements of 4.5% over BERT-CRF and 4.3% over BERT-CRF-joint on F1 score of AC, respectively. Besides, CasEE produces higher results on the recall of the evaluation metrics, since the sequence labeling methods have label conflicts that only one label can be predicted for those multi-label tokens. The results demonstrate the effectiveness of CasEE on overlapping event extraction.

(2) Compared to the pipeline methods, our method also outperforms them on the F1 score. The results show that CasEE achieves 3.1% and 2.6% improvements on F1 score of TC and AC over PLMEE, indicating the importance of solving the overlapped trigger problem in EE. Though the MRC based baselines can extract the overlapped triggers and arguments, CasEE still achieves better performance. Specifically, CasEE improves by a relative margin of 4.1% against the strong baseline MQAEE-2. The reason may be that CasEE jointly learns textual representations for subtasks, building helpful interactions and connections among the subtasks. The results demonstrate the superiority of CasEE over the above pipeline baselines.

Variants	TI (%)	TC (%)	AI (%)	AC (%)
BERT-softmax	76.5	49.0	56.1	53.5
BERT-CRF	77.9	52.4	61.0	58.4
BERT-CRF-joint	77.8	52.0	58.8	56.8
PLMEE	80.7	66.6	63.2	61.4
MQAEE-1	87.0	73.4	69.4	62.3
MQAEE-2	83.6	70.4	62.1	60.1
MQAEE-3	87.5	73.7	64.3	62.2
Ours	**89.0**	**74.9**	**71.5**	**70.3**

Table 3: Results of overlap sentences in testing. F1 scores are reported for each evaluation metric.

Variants	TI (%)	TC (%)	AI (%)	AC (%)
BERT-softmax	86.9	79.9	**76.2**	**74.1**
BERT-CRF	88.4	80.8	74.9	72.8
BERT-CRF-joint	86.9	79.9	76.1	74.0
PLMEE	86.4	79.7	75.7	74.0
MQAEE-1	88.0	78.5	65.1	57.7
MQAEE-2	**89.0**	**82.0**	74.2	72.3
MQAEE-3	87.1	77.6	71.3	69.6
Ours	88.4	80.2	74.0	72.3

Table 4: Results of normal sentences in testing. F1 scores are reported for each evaluation metric.

4.5 Analysis on Overlap/Normal Data

To further understand the performance in testing, we divide the original test data into two groups: the sentences with overlapped elements and the sentences without overlapped elements.

Variants	P (%)	R (%)	F1 (%)
MaxP	88.1	89.2	88.5
MeanP	88.3	89.8	88.6
CLS	**88.9**	88.7	88.5
CasEE	87.5	**91.9**	**89.2**

Table 5: Results of type detection decoder variants. The evaluation metric is precision, recall and F1 score in macro average of type classification.

Variants	P (%)	R (%)	F1 (%)
w/o self-attention	89.2	88.1	88.6
w/o condition	86.5	87.6	87.0
repl. concatenation	89.3	87.7	88.5
repl. addition	**90.4**	88.8	89.6
repl. gate mechanism	90.2	88.2	89.1
CasEE	90.1	**90.2**	**90.1**

Table 6: Results of trigger extraction decoder variants. The evaluation metric is precision, recall and F1 score on TC metric with oracle results of type detection.

Performance on Overlap Sentences. As shown in table 3, our method significantly outperforms previous methods on the overlap sentences. The improvements may come from the property that our method avoids label conflicts compared to the sequence labeling methods, and builds more effective feature-level connections among subtasks compared to the pipeline methods.

Performance on Normal Sentences. As shown in table 4, our method still performs acceptable results on the normal sentences without overlapped event elements. The sequence labeling methods reveal similiar results on trigger extraction but relatively better results on argument extraction, where they avoid potential propagation errors of the cascade decoding. Besides, PLMEE performs similar results on trigger extraction but relatively better results on argument extraction, where the reason may be that it adopts elaborate re-weighting losses for different argument roles as in its original literature. In addition, MQAEE-2 predicts more accurate triggers since it jointly predicts triggers with types, but it unfortunately ignores feature-level connections among the subtasks, making the argument extraction results similar to CasEE. Even so, the vanilla CasEE still conducts acceptable performance on the normal sentences compared to the baselines. We would further tackle the potential propagation errors and improve the performance for the general event extraction in the future work.

Variants	P (%)	R (%)	F1 (%)
w/o self-attention	82.8	81.7	82.2
w/o indicator function	84.1	81.4	82.7
w/o position embedding	83.2	81.5	82.3
w/o condition	**84.7**	78.2	81.3
repl. concatenation	84.0	79.3	81.6
repl. addition	84.2	78.2	81.1
repl. gate mechanism	84.6	80.2	82.4
CasEE	84.1	**83.7**	**83.9**

Table 7: Results of argument extraction decoder variants. The evaluation metric is precision, recall and F1 score on AC metric with oracle results of type detection and trigger extraction.

4.6 Discussion for Model Variants

To investigate the effectiveness of each module, we conduct variant experiments for CasEE.

Detection Module Variants. Table 5 shows performance of type detection variants. Specifically, MaxP/MeanP aggregates textual representations by applying max/mean pooling over BERT hidden states; CLS utilizes the hidden state of the special token `<CLS>` as the sentence representation. The results show that our method outperforms all the above variants on F1 score, indicating that learning sentence representation adaptive to the event type produces better representation for type detection.

Extraction Module Variants. Table 6 and Table 7 show performance of decoder variants for trigger extraction and argument extraction, respectively. We remove the self-attention layer in the both extraction decoders, and remove the relative position embeddings and the indicator function in the argument extraction decoder. The results demonstrate the effectiveness of each module.

Furthermore, we conduct experiments to explore the impact of condition fusion function ϕ. The experiments include: 1) we simply remove condition integrate function; 2) we achieve ϕ by concatenating the condition and token representations; 3) we achieve ϕ by simply adding the condition embedding to token representations; 4) we achieve ϕ by the gate mechanism, which adds the condition embedding to token representations according to a learnable trade-off factor. The results show that the performance without condition fusion function decline significantly on the F1 score in the two decoders, since the model can not discern different targets to extract in the sentence. Besides, empirical results also show that CLN performs better performance than other fusion functions on F1 scores in the two decoders, indicating that CLN can generate better conditional token representations for downstream subtasks.

5 Conclusion

This paper proposes a joint learning framework with cascade decoding for overlapping event extraction, termed as CasEE. Previous studies usually assume that events appear in sentences without overlaps, which are not applicable to the complicated overlapping scenarios. CasEE sequentially performs type detection, trigger extraction and argument extraction, where the overlapped targets are separately extracted conditioned on former predictions. All subtasks are jointly learned to capture dependencies among subtasks. Experiments on the public dataset demonstrate that our model outperforms previous competitive methods on overlapping event extraction. Our future work may further tackle the potential error propagation problem in the cascade decoding paradigm, and improve the performance for the general event extraction.

Acknowledgments

Corresponding authors: Lihong Wang and Shu Guo. We thank all the anonymous reviewers for their insightful comments and constructive suggestions. This work is supported by the National Natural Science Foundation of China (No.61772151), and the Youth Innovation Promotion Association of CAS (Grant No. 2021153).

References

Lei Jimmy Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. 2016. Layer normalization. *CoRR*.

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.*

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@EMNLP 2020, pages 74–83.*

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019*, pages 4171–4186.

Xinya Du and Claire Cardie. 2020a. Document-level event role filler extraction using multi-granularity contextualized encoding. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, pages 8010–8020.*

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

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations*.

Fayuan Li, Weihua Peng, Yuguang Chen, Quan Wang, Lu Pan, Yajuan Lyu, and Yong Zhu. 2020. Event extraction as multi-turn question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings, pages 829–838.*

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.*

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, pages 1641–1651.*

Shulin Liu, Yang Li, Feng Zhang, Tao Yang, and Xinpeng Zhou. 2019. Event detection without triggers. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers), pages 735–744.*

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.*

Ying Luo and Hai Zhao. 2020. Bipartite flat-graph network for nested named entity recognition. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, pages 6408–6418.*

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

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

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 Thirty-Second AAAI Conference on Artificial Intelligence, pages 5916–5923*.

Jianlin Su. 2019. Conditional text generation based on conditional layer normalization.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems, pages 5998–6008*.

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, pages 5783–5788*.

Yucheng Wang, Bowen Yu, Yueyang Zhang, Tingwen Liu, Hongsong Zhu, and Limin Sun. 2020. Tplinker: Single-stage joint extraction of entities and relations through token pair linking. In *Proceedings of the 28th International Conference on Computational Linguistics, pages 1572–1582*.

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*.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45.

Nuo Xu, Haihua Xie, and Dongyan Zhao. 2020. A novel joint framework for multiple chinese events extraction. In *Chinese Computational Linguistics - 19th China National Conference, pages 174–183*.

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 Conference of the Association for Computational Linguistics, pages 5284–5294*.

Bowen Yu, Zhenyu Zhang, Jiawei Sheng, Tingwen Liu, Yubin Wang, Yucheng Wang, and Bin Wang. 2021. Semi-open information extraction. In *Proceedings of the Web Conference*.

Bowen Yu, Zhenyu Zhang, Xiaobo Shu, Tingwen Liu, Yubin Wang, Bin Wang, and Sujian Li. 2020. Joint extraction of entities and relations based on a novel decomposition strategy. In *ECAI 2020 - 24th European Conference on Artificial Intelligence, pages 2282–2289*.

Xiangrong Zeng, Daojian Zeng, Shizhu He, Kang Liu, and Jun Zhao. 2018. Extracting relational facts by an end-to-end neural model with copy mechanism. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, pages 506–514*.

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*.

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 AAAI-21*.

A Hyper-parameter Settings

Our implementation is based on PyTorch[5]. We trained our models with a NVIDIA TESLA T4 GPU. For re-implementation, we report our hyper-parameter settings on the dataset in Table 8. Note that the hyper-parameter settings are tuned on the validation data by grid search with 3 trials.

Hyper-parameter	CasEE
type embedding dimension d	768
position embedding dimension d_p	64
dropout rate of decoders	0.3
batch size	8
training epoch	20
initial learning rate of BERT	$2e^{-5}$
learning rate of decoders	$1e^{-4}$
threshold ξ_1	0.5
threshold ξ_2	0.5
threshold ξ_3	0.5
threshold ξ_4	0.5
threshold ξ_5	0.5

Table 8: Hyper-parameter settings of CasEE.

B Details of MRC Based Baselines

Here we describe the details of the extended MRC baselines. Since the MRC paradigm could place condition information in the questions, we extend it to solve the overlapping event extraction.

MQAEE-1 contains two models: 1)A BERT classifier to detect event types; 2) A MRC BERT to extract triggers and arguments. The question template is like `<type>` to predict triggers with type `type`, and `<role>` to predict arguments with role `role`. Though this method neglects associations between the trigger and argument, it tackles overlapped trigger problem and overlapped argument problem since the overlapped targets are extracted separately according to different questions.

MQAEE-2 contains two models: 1) A MRC BERT to extract all triggers with types. The question template is a single word `trigger` to predict all typed triggers. 2) A MRC BERT to extract arguments in different roles. The question template is like `<type>and<trigger>` to predict all arguments associated with the type `type` and the trigger `trigger`. This method tackles overlapped trigger problem with multiple taggers, and tackles overlapped argument problem by extracting argument separately according to both the type and trigger.

MQAEE-3 contains three models: 1)A BERT classifier to detect event types; 2) A MRC BERT to extract triggers with different types. The question template is like `<type>` to predict triggers with type `type`. 3) A MRC BERT to extract arguments in different roles. The question template is like `<type>and<trigger>` to predict all arguments associated with the type `type` and the trigger `trigger`. This method tackles overlapped trigger problem by extracting triggers according to the type, and tackles overlapped argument problem according to both the type and trigger.

[5]https://pytorch.org/