Extracting Trigger-sharing Events via an Event Matrix

Jun Xu*, **Weidi Xu***, **Mengshu Sun**†, **Taifeng Wang and Wei Chu**
Ant Group, Hangzhou, China
{xj169860, weidi.xwd, mengshu.sms, taifeng.wang, weichu.cw}@antgroup.com

Abstract

A growing interest emerges in event extraction which aims to extract multiple events with triggers and arguments. Previous methods mitigate the problem of multiple events extraction by predicting the arguments conditioned on the event trigger and event type, assuming that these arguments belong to a single event. However, the assumption is invalid in general as there may be multiple events. Therefore, we present a unified framework called MatEE for trigger-sharing events extraction. It resolves the kernel bottleneck by effectively modeling the relations between arguments by an event matrix, where trigger-sharing events are represented by multiple cliques. We verify the proposed method on 3 widely-used benchmark datasets of event extraction. The experimental results show that it beats all the advanced competitors, significantly improving the state-of-the-art performances in event extraction.

1 Introduction

Event Extraction (EE) is a structured prediction task that aims to recognize and extract the events in the text. An event of a particular event type typically contains an event trigger and several relevant arguments. The EE task has long been challenging in the information extraction field as it involves various sub-tasks, e.g., named entity recognition (Li et al., 2021b,a), relation extraction (Ahmad et al., 2021). The primary studies (Sha et al., 2018; Chen et al., 2015; Du and Cardie, 2020; Xu and Sun, 2022; Wang et al., 2019) of EE focus on extracting single event by simply detecting and labeling the spans. Then several improved methods were proposed for extracting multiple events (Li et al., 2020a; Chen et al., 2020; Du et al., 2021; Sheng et al., 2021; Nguyen et al., 2021; Lin et al., 2020; Wang et al., 2019). These models typically formulate the extraction process as a sequence of conditional predictions, i.e., first recognize the event trigger, categorize the event type, and then extract the corresponding arguments in the events conditioned on the event trigger and event type. However, they still cannot extract multiple events jointly and mitigate the problem of multiple events extraction by multiple fine-grained conditional predictions. They make an implicit assumption that with a fine-grained condition (trigger and event type) there can be only one event, but such an assumption is generally invalid. Even with the specified event trigger and event type, there can be multiple events. In fact, there are around 16% samples in the FewFC (Zhou et al., 2021) dataset violating this assumption. An example of such multiple events is illustrated in Figure 1. There is a common trigger "increased" for four events where two events have the same event type. Therefore, the capability of joint extraction of such multiple events with the same trigger is indispensable.

高领资本在第二季度增持索菲亚1000万股，智慧能源300万股。
Hillhouse increased its holdings of Sofia by 10 million shares and Smart Energy by 3 million shares in Q2.

	Event-1	Event-2	Event-3	Event-4
event type	Equity Investment	Equity Investment	Equity Transfer	Equity Transfer
trigger	increased	increased	increased	increased
subject	Hillhouse	Hillhouse		
sub-org				
object	Smart Energy	Sofia		
obj-org			Hillhouse	Hillhouse
company			Sofia	Smart Energy
date	Q2	Q2	Q2	Q2
number			10 million	3 million
collateral			shares	shares

arguments

Figure 1: An example of multiple events (best viewed in color). In this case, even when the trigger and event type are given, i.e., "increased" and "equity investment", there still are two different events.

*These authors contributed equally to this work.
†Corresponding author.

Findings of the Association for Computational Linguistics: EMNLP 2022, pages 1189–1201
December 7-11, 2022

To this end, we propose a unified framework, MatEE, to predict trigger-sharing events in a single stage by using a novel predication formalism. Instead of predicting the argument spans independently, we turn to model the co-event relationship between the arguments as the token-token classification so that trigger-sharing events can be represented by a single event matrix. An example of an event matrix is illustrated in Figure 2. The grid in blue is used to indicate the argument boundary. The grid in other colors except white expresses two kinds of information: (1) two arguments co-exist in the same event, and (2) the argument role of the entity is denoted by the color in the grid. By this definition, trigger-sharing events can be represented by several cliques in the event matrix. Therefore, we can recover the events by extracting the maximal cliques and identify the arguments by retrieving the value of grids.

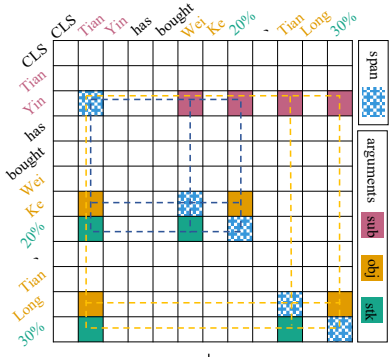

Figure 2: An example of event matrix of two events (best viewed in color), i.e., {sub: "Tian Yin", obj: "Wei Ke", stk: "20%"} and {sub: "Tian Yin", obj: "Tian Long", stk: "30%"}. The sentence is the translation of a Chinese example "天音收购唯科20%,天珑30%". The blue color indicates the argument spans, i.e., an grid (i, j) in blue denotes there is an argument with span (j, i). The grid (i, j) in other colors except white contains two meanings: (1) the two arguments co-exist in the same event where the former ends with i-th token and the latter starts with j-th token, (2) the role of the former argument is denoted by the color. The dashed lines indicate the cliques of the two events.

With the novel formalism, we present a neural framework for extracting those trigger-sharing events jointly (cf. the overview in Figure 3). First, the BERT (Devlin et al., 2019) is used to provide contextualized word representations, based on which we construct a trigger extractor for trigger extraction. Then an argument extractor is used to derive the event matrix by capturing the interactions between argument spans and the co-event relationships. Both the trigger extractor and argument extractor are trained to maximize the likelihood of labeled data, along with a particular contrastive learning objective for the trigger extractor.

Overall, our main contributions are two-fold:

- We present MatEE as a unified framework that represents the events by an event matrix for multiple trigger-sharing events extraction.
- Experimental results on three widely-used datasets reveal that our model achieves significant improvements over competitive methods.

2 Trigger-sharing Events Representation

Previous methods for multiple events extraction typically predict the arguments conditioned on the trigger and event type which are predicted ahead of arguments. For instance, MQAEE (Li et al., 2020a) predicts the arguments by constructing the questions with the trigger and event type. CasEE (Sheng et al., 2021) similarly predicts the spans of arguments with the conditional inputs. Although the predicted arguments may belong to multiple events, these methods are unable to distinguish various events from these arguments. Therefore, the capability of extracting multiple trigger-sharing events jointly is critical.

Event Matrix. To this end, we propose a novel prediction formalism to represent multiple trigger-sharing events in a single event matrix, motivated by recent advances in the pair representation (Li et al., 2021b; Wang et al., 2020). The event matrix effectively represents the co-event relationship between the arguments. Specifically, for a text with L tokens $x = \{w_1, w_2, ..., w_L\}$, the event matrix $\mathbf{M} = \{m_{i,j}\}$ is of size $\mathbb{R}^{L \times L}$. With N_a types of arguments roles, the grid $m_{i,j}$ can take the value in $\{0, 1, ..., N_a + 1\}$ where 0 denotes the blank tag, 1 denotes the span tag and $\{2, ..., N_a+1\}$ correspond to the N_a argument roles such as "sub", "obj" and "stk". The span tag is used to identify the argument spans in the events. $m_{i,j}$ is 1 if and only if there is an argument that starts with j-th token and ends with i-th token. $m_{i,j}$ in $\{2, ..., N_a + 1\}$ is used to indicate the co-event relationship of arguments. First, it means that there are two arguments (a_1, a_2) co-exist in the same event, a_1 ends with j-th token and a_2 starts with i-th token. We assume that every argument has a unique boundary so that the spans of a_1 and a_2 can be directly inferred by (i, j). Second, its value indicates the argument role of a_1 so

that we can align the spans with the argument roles. By this definition, the co-existence of argument pairs can determine all the arguments of a single event. It is because if the arguments are in the same event, every two arguments are connected and the arguments form a clique in the graph. Therefore, we can use a maximal clique algorithm to recover the events from the event matrix.

Figure 2 shows an event matrix where the value of $m_{i,j}$ is denoted by various colors. Let purple denote the argument role of "sub". The color of the grid ("Yin', "Wei") is purple since the argument ending with "Yin" and the argument starting with "Wei" are in the same event, and the former argument is a "sub". To derive the corresponding arguments "Tian Yin" and "Wei Ke" by "Yin" and "Wei", we can refer to the blue grids ("Yin", "Tian") and ("Ke", "Wei") which denote the spans of the arguments. Taking the whole event matrix into consideration, we can know the "Tian Yin", "Wei Ke" and "20%" are in the same event and their argument roles are "sub", "obj" and "stk" by the color of grids. Consequently, the two events {sub: "Tian Yin", obj: "Wei Ke", stk: "20%"} and {sub: "Tian Yin", obj: "Tian Long", stk: "30%"} can be represented by this single event matrix.

Representing Single Argument. There is a special case for such formalism. When the event has only one argument, we can not construct an argument pair to indicate the role of this argument. To mitigate this problem, we leverage the CLS token as a proxy to pair the individual argument. Therefore, for convenience, we let the first token w_1 in the text be the CLS token.

3 The MatEE Framework

The architecture of our framework is illustrated in Figure 3, which mainly consists of three components. First, the widely-used pre-trained language model BERT is used as the encoder to yield word representations. Then a trigger extractor is used to extract the triggers based on sequence labeling. Afterward, an argument extractor that contains a multi-head biaffine network is used to predict the event matrix by the maximal clique decoding.

3.1 Encoder Layer

We leverage BERT (Devlin et al., 2019) to process the text for our model due to its effectiveness in the event extraction (Sheng et al., 2021; Yang et al., 2021). The text with L tokens is encoded by BERT to derive the vector representations $\mathbf{H} = \{\mathbf{h}_1, \mathbf{h}_2, ..., \mathbf{h}_L\} \in \mathbb{R}^{L \times D}$, where D is the dimension of the embedding.

3.2 Trigger Extraction

We formalize the trigger extraction as a sequence labeling task. Since previous studies (Li et al., 2020b, 2021a) have demonstrated that the span-based sequence labeling can solve the problem of overlapping in the entity recognition, we apply it to extract the trigger.

Triggers Extractor. First, we extract the trigger spans for each event type from the text under the span-based framework. This module is mainly divided into two parts, i.e., start index prediction, and end index prediction. In the start index prediction, with the text representation $\mathbf{H}$ from the BERT encoder the module predicts the probability of each token being a start index of k-th event type:

$$\mathbf{T}_s^k = \text{sigmoid}(\text{MLP}_s^k(\mathbf{H}))\,, \tag{1}$$

where $\mathbf{T}_s^k \in \mathbb{R}^L$ is the start index probability distribution of the k-th event type and MLP denotes a linear transformation layer. With the start index, in the end index prediction, this module concatenates the text representation $\mathbf{H}$ with the start index prediction $\mathbf{T}_s$ and calculates the probability of each token being an end index by:

$$\mathbf{T}_e^k = \text{sigmoid}(\text{MLP}_e^k([\mathbf{H}; \mathbf{T}_s^k]))\,, \tag{2}$$

where $[\cdot;\cdot]$ denotes the concatenation operation (i.e., $[\mathbf{H}; \mathbf{T}_s^k] \in \mathbb{R}^{L \times (D+1)}$), $\mathbf{T}_e^k \in \mathbb{R}^L$ is the end index probability distribution of the k-th event type. When selecting the trigger spans during the prediction, for each event type and a trigger start index, we pick all end positions within the maximum length of triggers to obtain the candidate triggers for following argument extraction. In this way, our model can extract multiple triggers. The objective of trigger extraction is to maximize the likelihood:

$$\mathcal{L}_t = \sum_{k=1}^{N_e} \frac{1}{2}[\text{CE}(\mathbf{T}_s^k, \mathbf{T}_s^{k*}) + \text{CE}(\mathbf{T}_e^k, \mathbf{T}_e^{k*})]\,, \tag{3}$$

where N_e denotes the number of event types and $\mathbf{T}_s^{k*}, \mathbf{T}_e^{k*} \in \mathbb{R}^L$ are the target start and end labels of k-th event type. CE is the cross-entropy loss.

Discounted Contrastive Learning. In practice, we found that there is a lot of confusion about event types, especially "收购", "股权股份转让" and "投

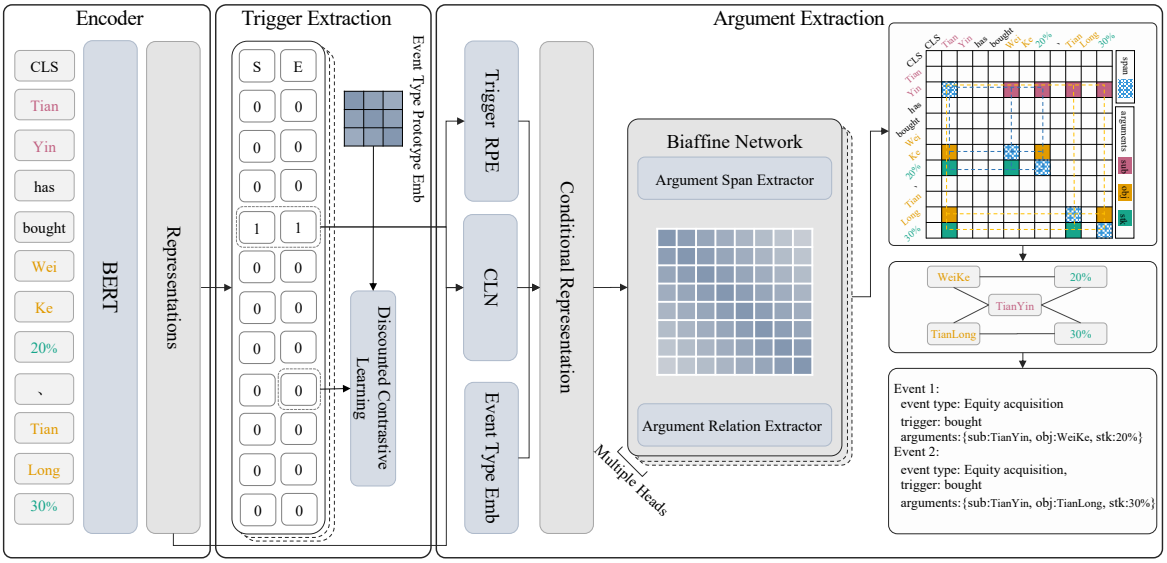

Figure 3: The overview of MatEE framework (best viewed in color). It contains a BERT encoder, a trigger extractor, and an arguments extractor. CLN and Trigger RPE represent conditional layer normalization and trigger relative position encoding, respectively.

资" in FewFC. Therefore, we design a discounted contrastive learning for the confusing event type.

$$\mathcal{L}_c = -\log \frac{\sum_{j\in\{\mathcal{T}^+\}} \exp(f(\mathbf{T}, \mathbf{E}_e^j)/\tau)}{\sum_{j\in\{\mathcal{T}^+,\mathcal{T}^-\}} \exp(f(\mathbf{T}, \mathbf{E}_e^j)/\tau)}, \tag{4}$$

where $\mathbf{E}_e \in \mathbb{R}^{N_e \times N_e}$ is the prototype embedding of event types and $\mathbf{E}_e^j$ represents the prototype embedding of event type j. $\mathbf{T} \in \mathbb{R}^{L\times N_e}$ is the start or end index prediction. τ is a temperature hyperparameter, $\mathcal{T}^+$ ($\mathcal{T}^-$) is the set of positive (negative) event types. The score function f is expressed as follows:

$$f(\mathbf{T}, \mathbf{E}_e^j) = \frac{\text{cosine\_similarity}(\mathbf{T}, \mathbf{E}_e^j)}{\log_2(\text{rank}_j + 1)}, \tag{5}$$

where cosine_similarity is the cosine similarity measures. $\text{rank}_j \in [1, N_e]$ represents the similarity ranking of the j-th event type with $\mathbf{T}$.

3.3 Argument Extraction

The argument extractor is responsible for extracting multiple events jointly conditioned on the event trigger and event type. As mentioned above, previous methods simply extract the arguments w.r.t. the trigger and event type, and cannot distinguish the events for the extracted arguments. In contrast, we leverage the event matrix to represent the multiple trigger-sharing events in a single stage. The right side of Figure 3 illustrates the framework of the argument extractor. It receives the trigger and event type from the trigger extractor, performs a series of computations to aggregate the information between the tokens, and then predicts an event matrix.

Conditional Representation. To integrate the trigger and event type, we adopt three components, i.e., the conditional layer normalization, event type embedding, and trigger relative position embedding. Conditional Layer Normalization (CLN) (Yu et al., 2021) is a mechanism to effectively fuse the identified trigger and the textual representation $\mathbf{h} \in \mathbb{R}^D$. The textual representation $\hat{\mathbf{C}} = \{\hat{\mathbf{c}}_1, ..., \hat{\mathbf{c}}_L\} \in \mathbb{R}^{L\times D}$ used to extract the arguments is calculated as follows:

$$\hat{\mathbf{C}} = \text{CLN}(\mathbf{t}, \mathbf{H}), \tag{6}$$

where $\mathbf{t} \in \mathbb{R}^D$ is the vector representing the trigger by performing the average pooling over the trigger tokens. Refer to Yu et al. (2021) for more details of CLN.

To encode the event type, we vectorize the event type as $\mathbf{e} \in \mathbb{R}^D$ by retrieving an embedding matrix. Besides, since the relative position is shown to

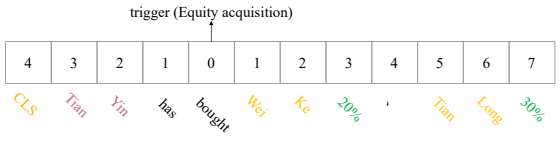

Figure 4: The trigger relative position is based on the trigger position and relative distance.

be significant in relevant tasks (Yang et al., 2016), we also use a relative position vector as shown in Figure 4 to indicate the position of the trigger. Denoting $\mathbf{P}_t \in \mathbb{R}^{L\times D}$ as the trigger relative position embedding of trigger t, the fused representation is obtained as follows:

$$\mathbf{C} = \mathrm{MLP}_c([\hat{\mathbf{C}}; \mathbf{E}; \mathbf{P}_t]) \,, \tag{7}$$

where $\mathbf{E} \in \mathbb{R}^{L\times D}$ repeats event type embedding $\mathbf{e}$ L times so that we can directly concatenate these representations. The output of MLP is $\mathbf{C} \in \mathbb{R}^{L\times D}$.

Multi-Heads Biaffine Network. The representation $\mathbf{C}$ integrates the text with the information from the event trigger and the event type. Then we can use it to predict the event matrix. To fully aggregate the information of arguments and their relationships, inspired by Li et al. (2021b), we propose two feature extractors, i.e., a span extractor and a relation extractor, to extract argument span and the relationship between arguments respectively.

Specifically, the span extractor obtains the representations of spans $\hat{\mathbf{S}} \in \mathbb{R}^{L\times D}$ as follows:

$$\begin{aligned} \mathbf{A}_e &= \mathrm{softmax}(\mathrm{MLP}(\mathbf{C})) \,, \\ \hat{\mathbf{S}} &= \mathrm{MLP}([\mathbf{C}; \mathbf{A}_e]) \,, \end{aligned} \tag{8}$$

where $\mathbf{A}_e \in \mathbb{R}^{L\times 2}$ is used to identify the end index of arguments. The two MLPs denote two different dense layers but we omit the subscripts for clarity.

To capture the relationships between the arguments, the relation extractor derives the vector representations for the tokens as:

$$\begin{aligned} \mathbf{A}_r &= \sigma(\mathbf{C}\mathbf{W}_r\mathbf{C}^T) \,, \\ \hat{\mathbf{R}} &= \mathrm{MLP}([\mathbf{C}; \mathbf{A}_r]) \,, \end{aligned} \tag{9}$$

where $\mathbf{W}_r \in \mathbb{R}^{D\times D}$ is a learnable matrix for the bilinear transformation, $\mathbf{A}_r \in \mathbb{R}^{L\times L}$ is used to represent the relationship between arguments, $[\mathbf{C}; \mathbf{A}_r] \in \mathbb{R}^{L\times(D+L)}$ is the input of the argument relation extractor and $\hat{\mathbf{R}} \in \mathbb{R}^{L\times D}$ is the output.

To jointly extract the argument span and relation, we apply a biaffine network Yu et al. (2020) to fuse the outputs of the span extractor and the relation extractor. However, the biaffine model is insensitive to the entity length and the boundaries, often rendering the predictions of entity boundaries incorrect. To mitigate this problem, we add the relative distance between the tokens (cf. Appendix A) as the input and vectorize them by an embedding matrix. With the relative position embedding of the arguments $\mathbf{P}_a \in \mathbb{R}^{L\times L}$, we concatenate it with the argument span representation $\hat{\mathbf{S}}$ and the argument relation representation $\hat{\mathbf{R}}$ to gather the position information. To maintain the dimension, we use an MLP to map the concatenated vectors to the D-dimensional vectors. Then, the biaffine network is used to fuse the representations of two extractors:

$$\begin{aligned} \mathbf{S} &= \mathrm{MLP}([\hat{\mathbf{S}}; \mathbf{P}_a]) \,, \\ \mathbf{R} &= \mathrm{MLP}([\hat{\mathbf{R}}; \mathbf{P}_a]) \,, \\ \mathbf{v}_{i,j} &= \mathbf{s}_i^T \mathbf{U} \mathbf{r}_j + \mathbf{W}_b[\mathbf{s}_i; \mathbf{r}_j] + \mathbf{b} \,, \end{aligned} \tag{10}$$

where $\mathbf{U} \in \mathbb{R}^{D\times F\times D}$ and $\mathbf{W}_b \in \mathbb{R}^{F\times 2D}$ are the trainable parameters (F is the dimension of $\mathbf{v}_{i,j}$), $\mathbf{b} \in \mathbb{R}^F$ is the bias, $\mathbf{s}_i$ and $\mathbf{r}_j$ denote the i-th and j-th element of $\mathbf{S}$ and $\mathbf{R}$ respectively. Motivated by Transformer (Vaswani et al., 2017), we use multiple biaffine heads to capture the interactions between argument spans and argument relations from different perspectives. Finally, the output of the multi-head biaffine layer is:

$$\mathbf{M} = \mathrm{softmax}(\mathrm{MLP}([\mathbf{V}^1; ...; \mathbf{V}^{N_h}])) \,, \tag{11}$$

where N_h is the number of heads, $\mathbf{V}^k = \{\mathbf{v}^k_{i,j}\}$ denotes the output of k-th biaffine head. $\mathbf{M} \in \mathbb{R}^{L\times L\times(N_a+2)}$ is the predicted probability of the event matrix after aggregating multiple biaffine heads, where $N_a + 2$ is the number of categories in the event matrix (N_a argument roles and two extra categories, i.e., span and blank). The overall objective of argument extraction of a data sample is as follows:

$$\mathcal{L}_a = \sum_{k,t\in\mathcal{T}_k} \mathrm{CE}(\mathbf{M}, \mathbf{M}^*) \,, \tag{12}$$

where $\mathcal{T}_k$ is the set of triggers of k-th event type, $\mathbf{M}^* \in \mathbb{R}^{L\times L}$ is the target event matrix.

Maximal Clique Decoding. As shown in Algorithm 1, to decode multi-events from the prediction $\mathbf{M}$, we construct an argument role matrix $\mathbf{O}$ to identify the argument role in each event, and an undirected relation graph $\mathbf{G}$ to represent the co-event relationship between arguments. The algorithm of graph construction is shown in Algorithm 1, which aims to find all argument spans and relations to reconstruct the events.

An example of constructed graphs is shown in Figure 5. With the graph $\mathbf{G}$, we can decode all maximal cliques based on the Bron-Kerbosch algorithm (cf. Appendix B) to find the corresponding

Algorithm 1: Graph Construction

Input : The predicted event matrix $\mathbf{M}$, the maximal number of arguments Q, the size of $\mathbf{M}$ L.

Output : The undirected graph $\mathbf{G}$ and argument role matrix $\mathbf{O}$.

$M \leftarrow \arg\max \mathbf{M}$;
$A \leftarrow \varnothing$;
$O, G \leftarrow int[Q][Q] = 0$;
for $col \leftarrow 0$ **to** L **do**
  for $row \leftarrow 0$ **to** L **do**
    if $M[row][col] = 1$ *and* $|A| < Q$ **then**
      $A \leftarrow A \cup set((col, row))$
for i, a_i *in* A **do**
  for j, a_j *in* A **do**
    $row\_i, col\_j = a_i[1], a_j[0]$;
    if $M[row\_i][col\_j] \notin [0, 1]$ **then**
      $O[i][j] \leftarrow M[row\_i][col\_j]$;
      $G[i][j] \leftarrow 1$; $G[j][i] \leftarrow 1$;
return G, O

arguments in various events. As shown in Section 2, the nodes in a maximal clique correspond to the arguments in an event.

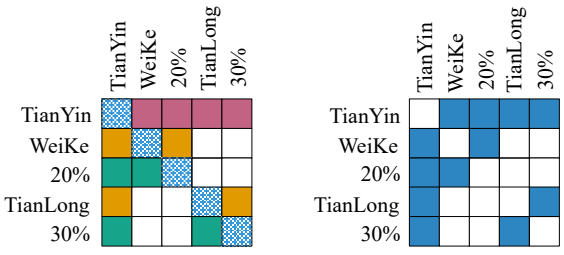

Figure 5: Left: the argument matrix $\mathbf{O}$ is used to identify the argument role in each event. Right: the undirected relation graph $\mathbf{G}$ is used to represent the co-event relationship between arguments.

For example, in the right of Figure 5, we can decode two sets of event arguments {"Tian Yin", "Wei Ke", "20%"} and {"Tian Yin", "Tian Long", "30%"}. Then, by the values in ("Tian Yin", "Wei Ke") of the left subgraph, it can be inferred that the argument role of "Tian Yin" in the event is "sub". Similarly, the argument role of "Wei Ke" is "obj". The role of the tail argument is inferred from tail "20%" to head "Tian Yin", so the role of "20%" is "stk". Therefore, we can extract two sets of arguments ({sub: "Tian Yin", obj: "Wei Ke", stk: "20%"} and {sub: "Tian Yin", obj: "Tian Long", stk: "30%"}) for the trigger "bought".

3.4 Model Training

The overall training objective to be minimized is:

$$\mathcal{L} = \sum_{\mathcal{D}} (\lambda_1 \mathcal{L}_t + \lambda_2 \mathcal{L}_a + \lambda_3 \mathcal{L}_c) \,, \tag{13}$$

where $\lambda_1, \lambda_2, \lambda_3$ are hyper-parameters to weight the components in the overall training objective, $\mathcal{D}$ denotes the training samples. The three losses are jointly trained in an end-to-end fashion.

4 Experimental Settings

Dataset. Our experiments are conducted on 3 widely-used datasets (cf. Table 1), including FewFC (Zhou et al., 2021), DuEE (Li et al., 2020c), and iFLYTEK*. For a fair comparison, we split the same training, validation, and test sets as in previous studies (Sheng et al., 2021; Zhou et al., 2021; Li et al., 2020a,c).

	#sample	#event	#ele-shr	#tri-shr
FewFC	8,982	12,890	5,345	848
DuEE	13,400	15,643	2,658	118
iFLYTEK	5,758	7,999	2,403	151

Table 1: Dataset Statistics. "#" denotes the amount. "ele-shr" means the number of events sharing elements, either argument or trigger. "tri-shr" represents the number of events sharing trigger.

Evaluation Metric. The evaluation metric is similar to previous studies (Sheng et al., 2021; Yang et al., 2021). TI, TC, AI, and AC denote trigger identification, trigger classification, argument identification, and argument classification, respectively. For more details, please refer to Appendix C.

Implementation Details. The number of heads of the biaffine layer is 3 with the dimension of head $F = 20$. We set the maximum length of triggers to 12 and the maximum length of arguments to 25. The worst-case time complexity of maximal clique decoding is exponential. Therefore, we set the maximum number of identified arguments Q in Algorithm 1 to 30. The weights of loss is $\lambda_1 = 1.0$, $\lambda_2 = 5.0$, $\lambda_3 = 0.5$ and the temperature τ in the trigger contrastive learning is 0.1. For other hyper-parameters and details, please refer to Appendix D.

*http://challenge.xfyun.cn/topic/info?option=phb

5 Experimental Results

5.1 Comparison Methods

BERT (Kruengkrai et al., 2020), which assigns a tag to each word with joint labels, such as B/I/O-type-trigger and B/I/O-type-role. **PLMEE** (Yang et al., 2019) solves overlapped argument problem by extracting role-specific arguments according to the trigger. **MQAEE** (Li et al., 2020a), which splits the event extraction task into two sub-tasks: trigger classification and argument extraction. The problem is solved as multi-turn QA in a pipeline. **CasEE** (Sheng et al., 2021) proposes a cascade decoding for overlapping event extraction, which sequentially performs type detection, trigger extraction, and argument extraction. For more implementation details and hyper-parameters of the compared methods, please refer to Appendix E.

5.2 Overall Results

We evaluate our framework on three widely-used event extraction datasets. Table 2 presents the comparisons between our model and other baselines. As seen, our model outperforms the previous best model CasEE by 1.80%, 2.13%, and 3.65% in AC F1 score in the FewFC, DuEE, and iFLYTEK datasets, respectively. The main reason is that our event matrix can extract trigger-sharing events. Besides, in trigger classification, our model outperforms the best baseline by 2.06% on FewFC, 1.45% on DuEE, and 2.10% on iFLYTEK. There are two key points to the significant improvement of our model. Firstly, the previous work CasEE produces event types with multi-label classification, and then performs trigger identification according to event types, while the two-stage model suffers from cumulative error. Secondly, we apply the contrastive learning to learn an effective representation of sentences and triggers, thus reducing false positives.

	Model	TC (%)			AC (%)		
		P	R	F1	P	R	F1
FewFC	BERT	82.55	65.38	72.97	75.27	62.44	68.26
	PLMEE	76.82	75.51	76.16	73.59	65.51	69.32
	MQAEE	81.87	70.42	75.71	74.25	63.46	68.43
	CasEE	77.87	78.25	78.06	71.19	71.33	71.26
	MatEE	80.40	78.88	**79.64**	74.56	70.46	**72.45**
DuEE	BERT	78.94	72.37	75.51	73.65	68.69	71.08
	PLMEE	78.89	76.95	77.91	74.78	73.41	74.09
	MQAEE	79.27	73.42	76.23	76.78	72.22	74.43
	CasEE	81.74	79.45	80.58	77.32	74.98	76.13
	MatEE	83.34	80.22	**81.75**	79.22	76.34	**77.75**
iFLYTEK	BERT	83.77	69.83	76.17	74.72	62.10	67.83
	PLMEE	79.92	76.31	78.07	71.94	69.79	70.85
	MQAEE	80.32	74.70	77.41	75.60	67.96	71.58
	CasEE	80.81	79.54	80.17	74.38	72.56	73.46
	MatEE	82.25	81.47	**81.86**	78.01	74.35	**76.14**

Table 2: Overall results of event extraction on FewFC, DuEE, and iFLYTEK.

5.3 Results For Trigger-sharing Events

We divide test data into three parts: 1) **Trigger-sharing** multiple events share a trigger; 2) **Element-sharing** multiple events share an argument or a trigger; 3) **Normal** elements are not shared between events. Take the FewFC dataset as an example, as shown in Table 3, **BERT** performs significantly worse in element-sharing events, the key reason is that they cannot extract element-sharing events. While **PLMEE**, **MQAEE** and **CasEE** can extract element-sharing events partially, but they cannot deal with multiple events sharing a trigger in the same event type. As can be seen from the table, the AC F1 score of our model in trigger-sharing events has improved by 11.84%. In element-sharing events, the AC F1 score of our model is greater than the AI F1 score. The major contribution is that our event matrix assigns the same argument to multiple events.

		TI(%)	TC(%)	AI(%)	AC(%)
Tri-sharing	BERT	82.62	67.14	39.41	37.08
	PLMEE	81.52	70.96	49.74	38.18
	MQAEE	82.42	71.21	49.56	46.95
	CasEE	85.82	73.14	55.18	54.73
	MatEE	**86.21**	**75.22**	**62.34**	**61.21**
Ele-sharing	BERT	83.34	68.61	63.42	62.31
	PLMEE	84.41	72.65	69.22	64.78
	MQAEE	85.33	73.47	69.32	65.80
	CasEE	86.73	76.25	72.16	70.54
	MatEE	**87.42**	**77.14**	**75.59**	**73.98**
Normal	BERT	84.66	75.85	74.53	72.19
	PLMEE	86.91	78.48	**79.01**	**72.31**
	MQAEE	88.08	77.20	78.83	70.17
	CasEE	89.55	79.26	73.91	71.74
	MatEE	**89.65**	**81.47**	75.12	71.33

Table 3: Results of trigger-sharing, element-sharing and normal in FewFC test set. F1 scores are reported for each evaluation metric.

5.4 Model Ablation Studies

We ablate each part of our model on the FewFC, as shown in Table 4. First, without discounted contrastive learning (DCL), we observe performance

drops of 2.29% on TC and 3.08% on AC, which verifies the usefulness of trigger contrastive learning. By removing CLN, event type embedding, and trigger relative position embedding, the performance drops slightly. Furthermore, after removing argument relative position encoding, the performance decrease is most significant. The main reason is that the argument span extractor and argument relation extractor are not position-sensitive, resulting in boundary errors. Instead of using biaffine, stack two extractors directly, the performance also drops obviously. This shows that the biaffine module increases the interaction of the two extractors, thereby increasing the effect of AC. At last, when head numbers are replaced by 1 or 2, we observe slight performance drops on this dataset.

	TI(%)	TC(%)	AI(%)	AC(%)
MatEE	88.71	79.64	75.32	72.45
w/o DCL	86.49	77.82	74.73	70.22
w/o CLN	86.78	78.25	75.01	71.47
w/o Event Type Emb.	87.01	78.88	74.19	71.60
w/o Trigger RPE	86.58	77.02	73.92	70.39
w/o Relative PE	84.29	76.43	73.26	70.10
w/o Biaffine	85.72	77.82	74.53	71.12
repl. Multi-Heads=1	86.98	78.34	74.39	71.89
Multi-Heads=2	87.12	78.91	74.57	72.20

Table 4: Model ablation studies (F1 score).

5.5 Case Study

In addition to the quantitative results, we visualize several event matrices for better comprehension of the behavior of the model. Figure 6 shows an example of predicted event matrix (left) with the target event matrix (right). We use the black color to denote the blank tag and the white color to denote the span tag to make the contrast clear. We can see that the predicted matrix can correctly represent two events by two cliques and the argument roles are also correct. This example qualitatively verifies the ability of the model to handle trigger-sharing events. We also show an incorrect case where the model fails to aggregate the arguments in a single event in Figure 7. Several co-event relationships between the arguments are missed so that the resulting events are incomplete. Compared to previous models, our model not only predicts the arguments and their corresponding roles but also predicts whether they exist in the same event. The additional prediction task may make the learning objective more difficult but the experimental results show that the including of such information is quite helpful for representing the events, and thus benefits the extraction process. We attach more visualization examples in Appendix H.

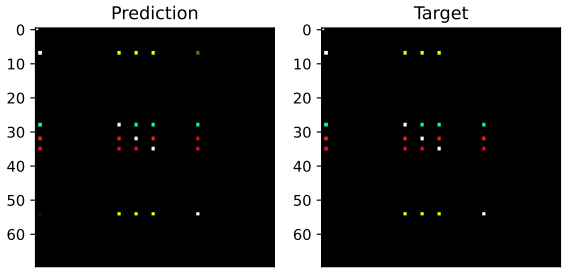

Figure 6: A prediction with two correct events. The input text is "2017年4月,公司开始筹划重大事项,计划收购辣妹子食品100%的股权,意在扩充公司产品线,2017年10月,因双方诉求未达一致,终止。".

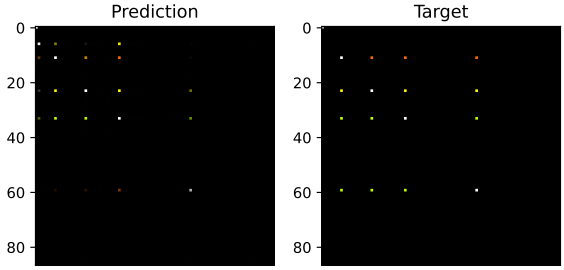

Figure 7: A prediction with two incorrect events. The input text is "10月31日*ST凯瑞公告称,公司10月30日收到法院关于德棉集团起诉公司名誉权侵权的通知后,当日下午向法院对德棉集团提起诉讼,要求其归还公司会计资料并赔偿3000万元。".

6 Related Work

Single Event Extraction. The primary studies typically considered the task as a sequence labeling problem, to assign each token a tag from a predefined scheme (e.g., BIO). Mainstream studies combine the CRF (Chen et al., 2015) with neural architecture, such as CNN (Chen et al., 2015), bi-directional LSTM (Sha et al., 2018), and Transformer (Du and Cardie, 2020). However, these methods fail to address the problem of multiple events. **Multiple Events Extraction.** There have been several studies (Li et al., 2020a; Chen et al., 2020; Ahmad et al., 2021; Liu et al., 2018; Hsu et al., 2022; Ma et al., 2022; Hwang et al., 2022; Feng et al., 2022) that cast multiple events extraction as a sequence of conditional predictions, i.e., recognizing trigger with event type first, and then extracting the corresponding arguments. Conditions are used in a variety of ways, such as Graph

Aggregation (Liu et al., 2018), Graph Representation (Xu et al., 2018), Seq2Seq (Du et al., 2021), MRC (Zhou et al., 2021), and Cascade Decoding (Sheng et al., 2021). However, the above methods make an implicit assumption that with a fine-grained condition (trigger and event type) there can be only one event. Inspired by extracting overlapped NER (Li et al., 2021b) and SPO (Wang et al., 2020), we propose an event matrix to deal with trigger-sharing events.

7 Conclusion

We have presented a unified framework MatEE based on the event matrix which is a novel formalism to represent multiple events jointly. Our framework is useful for various events extraction, especially trigger-sharing events. The empirical comparison and the results of analytical experiments verify its effectiveness. Beyond EE, our work may shed light on other complicated structured prediction tasks where the components are hard to predict sequentially. In the future, our work will focus on generalizing the MatEE to the case with multi-role arguments and the incorporation of the inherent prior constraints.

8 Acknowledgements

We would like to thank the anonymous reviewers for their constructive comments. This work was supported by Ant Group through Ant Innovative Research Program.

Limitations

Nonetheless, these results must be interpreted with caution and several limitations should be borne in mind. First of all, limited by the definition of the event matrix, an argument in an event can play at most one role as the grid can take one value. Second, the event matrix is unable to represent two events when an event is a subset of another event, although we did not find such cases in the datasets. Third, the worst-case time complexity of the maximal clique decoding algorithm is $\mathcal{O}(3^{\frac{n}{3}})$ for an n-vertex graph. Therefore, it is not suitable for document-level event extraction. Finally, we also notice that the predicted matrix may violate the definition of the event matrix. For instance in Figure 6, there is a dark yellow grid on the top-right corner but there is no bright grid on the bottom-left corner. Actually, the two grids form a pair and they should both be either blank or some colors of the argument roles. Taking such constraints into consideration can improve the confidence of the prediction and we leave it to further work.

References

Wasi Uddin Ahmad, Nanyun Peng, and Kai-Wei Chang. 2021. GATE: graph attention transformer encoder for cross-lingual relation and event extraction. In *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2-9, 2021*, pages 12462–12470. AAAI Press.

Coenraad Bron and Joep Kerbosch. 1973. Finding all cliques of an undirected graph (algorithm 457). *Commun. ACM*, 16(9):575–576.

Pei Chen, Hang Yang, Kang Liu, Ruihong Huang, Yubo Chen, Taifeng Wang, and Jun Zhao. 2020. Reconstructing event regions for event extraction via graph attention networks. In *Proceedings of the 1st Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 10th International Joint Conference on Natural Language Processing, AACL/IJCNLP 2020, Suzhou, China, December 4-7, 2020*, pages 811–820. Association for Computational Linguistics.

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, ACL 2015, July 26-31, 2015, Beijing, China, Volume 1: Long Papers*, pages 167–176. The Association for Computer Linguistics.

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, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 4171–4186. Association for Computational Linguistics.

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 2020, Online, November 16-20, 2020*, pages 671–683. Association for Computational Linguistics.

Xinya Du, Alexander M. Rush, and Claire Cardie. 2021. GRIT: generative role-filler transformers for

document-level event entity extraction. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume, EACL 2021, Online, April 19 - 23, 2021*, pages 634–644. Association for Computational Linguistics.

Yi Feng, Chuanyi Li, and Vincent Ng. 2022. Legal judgment prediction via event extraction with constraints. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2022, Dublin, Ireland, May 22-27, 2022*, pages 648–664. Association for Computational Linguistics.

I-Hung Hsu, Kuan-Hao Huang, Elizabeth Boschee, Scott Miller, Prem Natarajan, Kai-Wei Chang, and Nanyun Peng. 2022. DEGREE: A data-efficient generation-based event extraction model. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL 2022, Seattle, WA, United States, July 10-15, 2022*, pages 1890–1908. Association for Computational Linguistics.

EunJeong Hwang, Jay-Yoon Lee, Tianyi Yang, Dhruvesh Patel, Dongxu Zhang, and Andrew McCallum. 2022. Event-event relation extraction using probabilistic box embedding. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers), ACL 2022, Dublin, Ireland, May 22-27, 2022*, pages 235–244. Association for Computational Linguistics.

Canasai Kruengkrai, Thien Hai Nguyen, Sharifah Aljunied Mahani, and Lidong Bing. 2020. Improving low-resource named entity recognition using joint sentence and token labeling. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 5898–5905. Association for Computational Linguistics.

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, Online Event, 16-20 November 2020*, volume EMNLP 2020 of *Findings of ACL*, pages 829–838. Association for Computational Linguistics.

Fei Li, Zhichao Lin, Meishan Zhang, and Donghong Ji. 2021a. A span-based model for joint overlapped and discontinuous named entity recognition. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, ACL/IJCNLP 2021, (Volume 1: Long Papers), Virtual Event, August 1-6, 2021*, pages 4814–4828. Association for Computational Linguistics.

Jingye Li, Hao Fei, Jiang Liu, Shengqiong Wu, Meishan Zhang, Chong Teng, Donghong Ji, and Fei Li. 2021b. Unified named entity recognition as word-word relation classification. *Proceedings of the AAAI Conference on Artificial Intelligence, AAAI 2021*.

Xiaoya Li, Jingrong Feng, Yuxian Meng, Qinghong Han, Fei Wu, and Jiwei Li. 2020b. A unified MRC framework for named entity recognition. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 5849–5859. Association for Computational Linguistics.

Xinyu Li, Fayuan Li, Lu Pan, Yuguang Chen, Weihua Peng, Quan Wang, Yajuan Lyu, and Yong Zhu. 2020c. Duee: A large-scale dataset for chinese event extraction in real-world scenarios. In *Natural Language Processing and Chinese Computing - 9th CCF International Conference, NLPCC 2020, Zhengzhou, China, October 14-18, 2020, Proceedings, Part II*, volume 12431 of *Lecture Notes in Computer Science*, pages 534–545. Springer.

Ying Lin, Heng Ji, Fei Huang, and Lingfei Wu. 2020. A joint neural model for information extraction with global features. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 7999–8009. Association for Computational Linguistics.

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, Brussels, Belgium, October 31 - November 4, 2018*, pages 1247–1256. Association for Computational Linguistics.

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), ACL 2022, Dublin, Ireland, May 22-27, 2022*, pages 6759–6774. Association for Computational Linguistics.

Minh Van Nguyen, Viet Dac Lai, and Thien Huu Nguyen. 2021. Cross-task instance representation interactions and label dependencies for joint information extraction with graph convolutional networks. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2021, Online, June 6-11, 2021*, pages 27–38. Association for Computational Linguistics.

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, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intel-*

ligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018, pages 5916–5923. AAAI Press.

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, Online Event, August 1-6, 2021*, volume ACL/IJCNLP 2021 of *Findings of ACL*, pages 164–174. Association for Computational Linguistics.

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 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 5998–6008.

Xiaozhi Wang, Ziqi Wang, Xu Han, Zhiyuan Liu, Juanzi Li, Peng Li, Maosong Sun, Jie Zhou, and Xiang Ren. 2019. HMEAE: hierarchical modular event argument 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 2019, Hong Kong, China, November 3-7, 2019*, pages 5776–5782. Association for Computational Linguistics.

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, COLING 2020, Barcelona, Spain (Online), December 8-13, 2020*, pages 1572–1582. International Committee on Computational Linguistics.

Jun Xu, Siqi Shen, Dongsheng Li, and Yongquan Fu. 2018. A network-embedding based method for author disambiguation. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management, CIKM 2018, Torino, Italy, October 22-26, 2018*, pages 1735–1738. ACM.

Jun Xu and Mengshu Sun. 2022. DPNPED: dynamic perception network for polysemous event trigger detection. *IEEE Access*, 10:104801–104810.

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, ACL/IJCNLP 2021, (Volume 1: Long Papers), Virtual Event, August 1-6, 2021*, pages 6298–6308. Association for Computational Linguistics.

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, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 5284–5294. Association for Computational Linguistics.

Yunlun Yang, Yunhai Tong, Shulei Ma, and Zhi-Hong Deng. 2016. A position encoding convolutional neural network based on dependency tree for relation classification. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*, pages 65–74. The Association for Computational Linguistics.

Bowen Yu, Zhenyu Zhang, Jiawei Sheng, Tingwen Liu, Yubin Wang, Yucheng Wang, and Bin Wang. 2021. Semi-open information extraction. In *WWW '21: The Web Conference 2021, Virtual Event / Ljubljana, Slovenia, April 19-23, 2021*, pages 1661–1672. ACM / IW3C2.

Juntao Yu, Bernd Bohnet, and Massimo Poesio. 2020. Named entity recognition as dependency parsing. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 6470–6476. Association for Computational Linguistics.

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 *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2-9, 2021*, pages 14638–14646. AAAI Press.

A Argument Relative Position Encoding

The relative positions of the argument span extractor and argument relation extractor are shown in Figure 8. The argument relative position encoding

0	1	2	...	L-1
1	0	1	...	L-2
2	1	0	...	L-3
...	...	...	...	...
L-1	L-2	L-3	...	0

Figure 8: The argument relative position is based on all words position and relative distance.

$\mathbf{P}_a \in \mathbb{R}^{L \times L}$ is expressed as follows:

$$\mathbf{P}_a(i,j) = \begin{cases} sin(\omega_{v_{ij}} \cdot i), & \text{if } v_{ij} = 2k \\ cos(\omega_{v_{ij}} \cdot i), & \text{if } v_{ij} = 2k+1 \end{cases}$$

where $\omega_{v_{ij}} = \frac{1}{10000^{v_{ij}/L}}$, and v_{ij} is the value of the i-th row and the j-th column in Figure 8.

B Bron-KerBosch Algorithm

The basic form of the Bron–Kerbosch algorithm (Bron and Kerbosch, 1973) is a recursive backtracking algorithm that searches for all maximal cliques in a given graph $\mathbf{G}$. More generally, given three disjoint sets of vertices R, C, and X, it finds the maximal cliques that include all of the vertices in R, some of the vertices in C, and none of the vertices in X. In each call to the algorithm, C and X are disjoint sets whose union consists of those vertices that form cliques when added to R. In other words, $C \cup X$ is the set of vertices that are joined to every element of R. When C and X are both empty there are no further elements that can be added to R, so R is a maximal clique.

Algorithm 2: Maximal Clique Decoding

Input : The undirected relation graph $\mathbf{G}$.
Output : All argument sets P of multiple events.

$R, X, P \leftarrow \varnothing$;
$C \leftarrow$ *the node set of* $\mathbf{G}$;
def BKerb(*R,C,X*)**:**
  if $C = \varnothing$ *and* $X = \varnothing$ **then**
    add R *to* P
  for *v in C* **do**
    $N(v) \leftarrow$ *the neighbor set of* v;
    BKerb $(R \cup \{v\}, C \cap N(v), X \cap N(v))$;
    $C \leftarrow C \setminus \{v\}$;
    $X \leftarrow X \cup \{v\}$;
BKerb(*R,C,X*);
return P

C Evaluation Metric

We use the following criteria to measure each predicted event: 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. Finally, we use Precision (P), Recall (R), and F measure (F1) as the evaluation metrics.

D Implementation Details

We leverage the BERT-Base model as the textual encoder, which has 12 layers, 768 hidden units, and 12 attention heads. We train the model with an Adam weight decay optimizer. The initial learning rate is $2e^{-5}$ for BERT parameters and $1e^{-4}$ for other parameters, and the dropout rate is 0.1. The warming-up proportion for the learning rate is 0.1, and the max training epoch is set to 40. The batch size is set to 16. The max sequence length is 400.

E Comparision Methods Details

In this section, we provide more implementation details of baselines. For a fair comparison, all of these models are implemented using PyTorch and tested using the NVIDIA TESLA V100 GPU. **BERT** takes event extraction as a sequence labeling problem. It uses the "B/I/O-type-trigger" schema to identify triggers firstly and then uses the "B/I/O-type-role" schema to identify arguments [†]. The initial BERT learning rate is $2e^{-5}$ and the CRF learning rate is $5e^{-4}$. The batch size, training epoch, and sequence length are the same as MatEE. **PLMEE** uses "BERT+softmax" in each token to extract the event trigger and its event type. Based on the identified trigger, arguments are extracted by "BERT+span" [‡]. The initial learning rate for BERT is $2e^{-5}$ and $5e^{-4}$ for others. Other parameters take default values. **MQAEE** is reimplemented by us. The implemented framework is similar to PLMEE. Firstly, MRC is used to extract triggers and their event types. Based on the extracted triggers, a series of trigger templates are constructed (e.g., "股权投资事件中增持的主体是？", the template is "event type 事件中 trigger 的 argument role 是？") , and MRC is used again to extract their arguments. **CasEE** implementation and parameters are consistent with the author's official code [§].

F Multiple Events

The case of multiple events is illustrated in Figure 9, which can be mainly divided into two parts. Firstly, multiple events share a trigger with the same event type. Secondly, multiple events in other situations.

[†]https://github.com/lonePatient/BERT-NER-Pytorch
[‡]https://github.com/boy56/PLMEE
[§]https://github.com/JiaweiSheng/CasEE

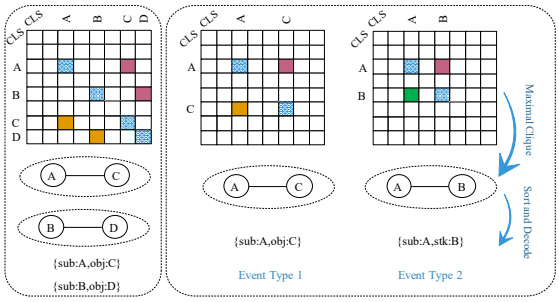

Figure 9: Left: multiple events share a trigger with the same event type. Right: multiple events in other situations.

For the first case, we apply the Bron-Kerbosch algorithm to split arguments into different maximal cliques, and then sort and decode to generate multiple events. For the second case, in the trigger extraction module, we extract triggers of different event types, and then extract the arguments of these triggers respectively.

G Argument-sharing Events

As can be seen from Figure 10, there are two types of argument-sharing events. First, multiple events share an argument with the same role. Then, an

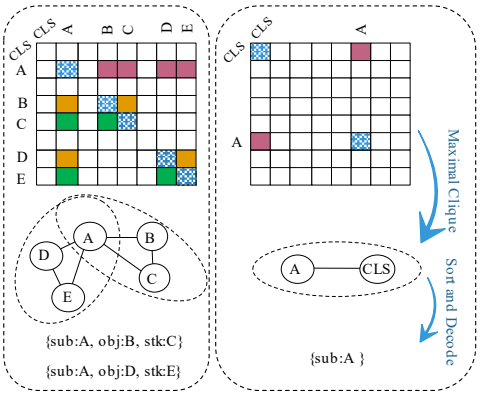

Figure 10: Left: multiple events share an argument with the same role. Right: a special case uses for an event with only one argument.

event has only one argument. For the left case, we apply the Bron-Kerbosch algorithm to generate two maximal cliques directly. For the right case, some events have only one argument. In this case, we use CLS as a proxy firstly, and then the rows and columns of CLS are filled with the only argument role. In practice, <CLS,A> is filled with "sub" and <A,CLS> is filled with "sub".

H More Visualizations

We also show several predicated event matrices here. Figure 11 illustrates an example of an event with only one argument pairing with the CLS token, which is a special case mentioned in Secion 2.

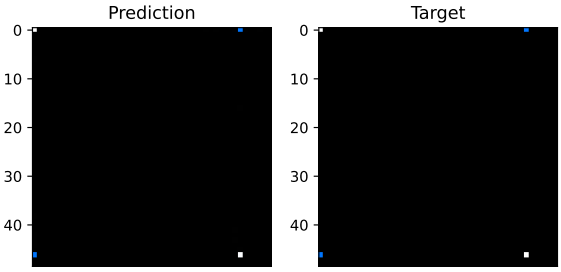

Figure 11: A correct prediction with only one argument.

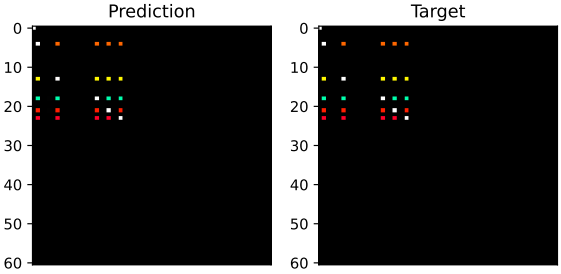

Figure 12: A correct prediction with high confidence.

Figure 12 shows a case where the prediction is very

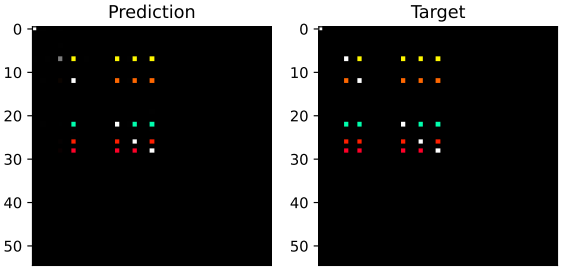

Figure 13: An incorrect prediction where an argument is missed.

close to the target event matrix. Figure 13 is an example of an incorrect event where an argument is missed in the prediction.