Document-level Event Extraction via Parallel Prediction Networks

Hang Yang[1,2], **Dianbo Sui**[1,2], **Yubo Chen**[1,2], **Kang Liu**[1,2], **Jun Zhao**[1,2], **Taifeng Wang**[3]
[1]National Laboratory of Pattern Recognition, Institute of Automation,
Chinese Academy of Sciences, Beijing, 100190, China
[2]School of Artificial Intelligence, University of Chinese Academy of Sciences,
Beijing, 100049, China
[3]Ant Group, Hangzhou, 310013, China
{hang.yang, dianbo.sui, yubo.chen, kliu, jzhao}@nlpr.ia.ac.cn ,
taifeng.wang@antgroup.com

Abstract

Document-level event extraction (DEE) is indispensable when events are described throughout a document. We argue that sentence-level extractors are ill-suited to the DEE task where event arguments always scatter across sentences and multiple events may co-exist in a document. It is a challenging task because it requires a holistic understanding of the document and an aggregated ability to assemble arguments across multiple sentences. In this paper, we propose an end-to-end model, which can extract structured events from a document in a parallel manner. Specifically, we first introduce a document-level encoder to obtain the document-aware representations. Then, a multi-granularity non-autoregressive decoder is used to generate events in parallel. Finally, to train the entire model, a matching loss function is proposed, which can bootstrap a global optimization. The empirical results on the widely used DEE dataset show that our approach significantly outperforms current state-of-the-art methods in the challenging DEE task. Code will be available at https://github.com/HangYang-NLP/DE-PPN.

Input document

…
[S3] On **November 1, 2018**, Shenzhen 007 Co., Ltd. received a notice that the corporate shareholder **Shanghai Fukong Co., Ltd** and the actual controller **Jing Yan** were judicial frozen.
…
[S7] The corporate shareholder holds 150000 shares of the company. The **10000 shares** were frozen by the **Shenzhen Intermediate Peoples Court** from **October 30, 2018** to **October 30, 2019**.
[S8] The controller of the company holds 310000 shares of the company. The **20000 shares** were frozen by the **Shenzhen Inter - mediate People's Court** on **November 1, 2018**.
…

Event Table of Equity Freeze

Event Role	Event-1	Event -2
Equity Holder	**Shanghai Fukong Co., Ltd**	**Jing Yan**
Froze Shares	**10000 shares**	**20000 shares**
Legal Institution	**Shenzhen Intermediate People's Court**	
Start Date	**October 30, 2018**	**October 30, 2019**
End Date	**November 1, 2018**	\

Figure 1: An example of a document contains two *Equity Freeze* type events: *Event-1* and *Event-2*. Words in bold-faced are arguments that scatter across multiple sentences.

1 Introduction

The goal of event extraction (EE) is to identify events of a pre-specified type along with corresponding arguments from plain texts. A great number of previous studies (Ahn, 2006; Ji and Grishman, 2008; Liao and Grishman, 2010; Hong et al., 2011; Li et al., 2013; Chen et al., 2015; Nguyen et al., 2016; Yang and Mitchell, 2016; Chen et al., 2017; Huang et al., 2018; Yang et al., 2019; Liu et al., 2020) focus on the sentence-level EE (SEE), while most of these works are based on the ACE evaluation (Doddington et al., 2004). [1] However, these SEE-based methods make predictions within a sentence and fail to extract events across sentences. To this end, document-level EE (DEE) is needed when the event information scatters across the whole document.

In contrast to SEE, there are two specific challenges in DEE: **arguments-scattering** and **multi-events**. Specifically, arguments-scattering indicates that arguments of an event may scatter across multiple sentences. For example, As shown in Figure 1, the arguments of *Event-1* are distributed in different sentences ([S3] and [S7]) and extraction within an individual sentence will lead to incomplete results. So this challenge requires the DEE model to have a holistic understanding of the entire document and an ability to assemble all relevant

[1] https://www.ldc.upenn.edu/collaborations/past-projects/ace

Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, pages 6298–6308
August 1–6, 2021.

arguments across sentences. Furthermore, it will be more difficult when coupled with the second challenge: multi-events, where multiple events are contained in a document.[2] As shown in Figure 1, there are two events *Event-1* and *Event-2* in a document with the same event type and there is no obvious textual boundary between the two events. The multi-events problem requires the DEE method to recognize how many events are contained in a document and achieve accurate arguments assembling (i.e., assign arguments to the corresponding event). As a result of these two complications, SEE methods are ill-suited for the DEE task, which calls for a model that can integrate document-level information, assemble relevant arguments across multiple sentences and capture multiple events simultaneously.

To handle these challenges in DEE, previous works (Yang et al., 2018; Zheng et al., 2019) formulate DEE as an event table filling task, i.e., filling candidate arguments into a predefined event table. Specifically, they model the DEE as a **serial prediction** paradigm, in which arguments are predicted in a predefined role order and multiple events are also extracted in predefined event order. Such a manner is restricted to the extraction of individual arguments, and the former extraction will not consider the latter extraction results. As a result, errors will be propagated and the extraction performance is under satisfaction.

In this paper, to avoid the shortage of serial prediction and tackle the aforementioned challenges in DEE, we propose an end-to-end model, named **D**ocument-to-**E**vents via **P**arallel **P**rediction **N**etworks (**DE-PPN**). DE-PPN is based on an encoder-decoder framework that can extract structured events from a whole document in a parallel manner. In detail, we first introduce a document-level encoder to obtain the document-aware representations. In such a way, a holistic understanding of the entire document is obtained. Then, we leverage a multi-granularity decoder to generate events, which consists of two key parts: a role decoder and an event decoder. The role decoder is designed for handling the argument-scattering challenge, which can assemble arguments for an event based on document-aware representations. For addressing the challenge of multi-events effectively, an event decoder is designed to support generating multiple events. Both of them are based on the non-autoregressive mechanism (Gu et al., 2018), which supports the extraction of multiple events in parallel. Finally, for comparing extracted events to ground truths, we propose a matching loss function inspired by the Hungarian algorithm (Kuhn, 1955; Munkres, 1957). The proposed loss function can perform a global optimization by computing a bipartite matching between predicted and ground-truth events.

In summary, our contributions are as follows:

- We propose an encoder-decoder model, DE-PPN, that is based on a document-level encoder and a multi-granularity decoder to extract events in parallel with document-aware representations.
- We introduce a novel matching loss function to train the end-to-end model, which can bootstrap a global optimization.
- We conduct extensive experiments on the widely used DEE dataset and experimental results demonstrate that DE-PPN can significantly outperform state-of-the-art methods when facing the specific challenges in DEE.

2 Methodology

Before introducing our proposed approach for DEE in this section, we first describe the task formalization of DEE. Formally, we denote $\mathcal{T}$ and $\mathcal{R}$ as the set of pre-defined event types and role categories, respectively. Given an input document comprised of N_s sentences $\mathcal{D} = \{S_i\}_{i=1}^{N_s}$, the DEE task aims to extract one or more structured events $Y = \{y_i\}_{i=1}^{k}$, where each event y_i^t with event type t contains a series of roles $(r_i^1, r_i^2, \ldots, r_i^n)$ filled by arguments $(a_i^1, a_i^2, \ldots, a_i^n)$. k is the number of events contained in the document, n is the number of pre-defined roles for the event type t, $t \in \mathcal{T}$ and $r \in \mathcal{R}$.

The key idea of our proposed model, DE-PPN, is that aggregate the document-level context to predict events in parallel. Figure 2 illustrates the architecture of DE-PPN, which consists of five key components: (1) candidate argument recognition, (2) document-level encoder, (3) multi-granularity decoder, (4) events prediction, and (5) matching loss function.

[2] According to our statistics, there are about 30% documents include multiple events in the widely used ChFinAnn (Zheng et al., 2019)

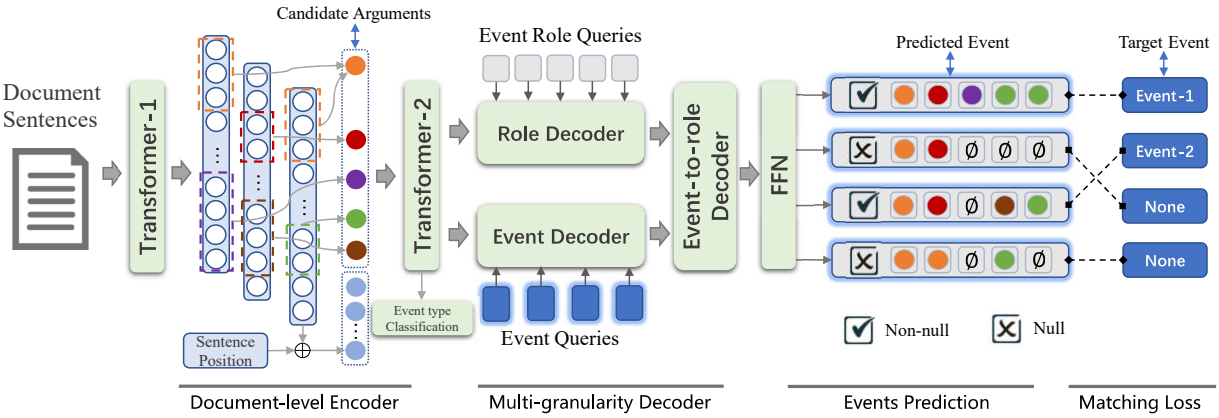

Figure 2: The overall architecture of DE-PPN. Given a document, the DE-PPN first encodes each sentence separately and recognizes candidate arguments from it. Then a document-level encoder is designed to get the document-level representations. And a multi-granularity decoder is used to generate events in parallel based on document-aware representations. Finally, the matching loss function can produce an optimal bipartite matching between predicted and ground-truth events, which bootstrap a global optimization.

2.1 Candidate Argument Recognition

Given a document $\mathcal{D} = \{S_i\}_{i=1}^{N_s}$ with N_s sentences, each sentence S_i with a sequence of tokens is first embedded as $[\mathbf{w}_{i,1}, \mathbf{w}_{i,2}, \ldots, \mathbf{w}_{i,l}]$, where l is the sentence length. Then, the word embeddings are fed into an encoder to obtain the contextualized representation. In this paper, we adopt the Transformer (Vaswani et al., 2017) as the primary context encoder. Through the encoder, we can get the context-aware embedding $\mathbf{C}_i$ of sentence S_i:

$$\mathbf{C}_i = \text{Transformer-1}(S_i) \tag{1}$$

where $\mathbf{C}_i \in \mathbb{R}^{l \times d}$ and d is the size of the hidden layer, and we represent each sentence in the given document as $\{\mathbf{C}_i\}_{i=1}^{N_s}$.

Finally, following Zheng et al. (2019), we model the sentence-level candidate argument recognition as a typical sequence tagging task. Through candidate argument recognition, we can obtain candidate arguments $\mathcal{A} = \{a_i\}_{i=1}^{N_a}$ from the given sentence S_i, where N_a is the number of recognized candidate arguments.

2.2 Document-level Encoder

To enable the awareness of document-level contexts for sentences and candidate arguments, we employ a document-aware encoder to facilitate the interaction between all sentences and candidate arguments. Formally, given an argument a_i with its span covering j-th to k-th in sentence S_i, we conduct a max-pooling operation over the token-level embedding $[\mathbf{c}_{i,j}, \ldots, \mathbf{c}_{i,k}] \in \mathbf{C}_i$ to get the local embedding $\mathbf{c}_i^a \in \mathbb{R}^d$ for it. Similarly, the sentence embedding $\mathbf{c}_i^s \in \mathbb{R}^d$ can be obtained by the max-pooling operation over the token sequence representation $\mathbf{C}_i$ of sentence S_i. Then, we employ the Transformer module, Transformer-2, as the encoder to model the interaction between all sentences and candidate arguments by a multi-head self-attention mechanism. Then we can get the document-aware representations for sentences and arguments. Note that we add the sentence representation with sentence position embeddings to inform the sentence order before feeding them into Transformer-2.

$$[\mathbf{H}^a; \mathbf{H}^s] = \text{Transformer-2}(\mathbf{c}_1^a \ldots \mathbf{c}_{N_a}^a; \mathbf{c}_1^s \ldots \mathbf{c}_{N_s}^s) \tag{2}$$

since arguments may have many mentions in a document, we utilize the max-pooling operation to merge multiple argument embeddings with the same char-level tokens into a single embedding. After the document-level encoding stage, we can obtain the document-aware sentences representation $\mathbf{H}^s \in \mathbb{R}^{N_s \times d}$ and candidate arguments $\mathcal{A}' = \{a_i\}_{i=1}^{N'_a}$ with representation $\mathbf{H}^a \in \mathbb{R}^{N'_a \times d}$.

Before decoding, we stack a linear classifier over the document representation by operating the max-pooling over $\mathbf{H}^s$ to conduct a binary classification for each event type. Then, for the predicted event type t with pre-defined role types, DE-PPN learns to generate events according to the document-aware candidate argument representations $\mathbf{H}^a \in \mathbb{R}^{N'_a \times d}$ and sentence representations $\mathbf{H}^s \in \mathbb{R}^{N_s \times d}$.

2.3 Multi-Granularity Decoder

To effectively address arguments-scattering and multi-events in DEE, we introduce a multi-granularity decoder to generate all possible events in parallel based on document-aware representations ($\mathbf{H}^a$ and $\mathbf{H}^s$). The multi-granularity decoder is composed of three parts: event decoder, role decoder, and event-to-role decoder. All of these decoders are based on the non-autoregressive mechanism (Gu et al., 2018), which supports the extraction of all events in parallel.

Event Decoder. The event decoder is designed to support the extraction of all events in parallel and is used to model the interaction between events. Before the decoding stage, the decoder needs to know the size of events to be generated. We use m learnable embeddings as the input of the event decoder, which are denoted as event queries $\mathbf{Q}^{event} \in \mathbb{R}^{m \times d}$. m is a hyperparameter that denotes the number of the generated events. In our work, m is set to be significantly large than the average number of events in a document. Then, the event query embeddings $\mathbf{Q}^{event}$ are fed into a non-autoregressive decoder which is composed of a stack of N identical Transformer layers. In each layer, there are a multi-head self-attention mechanism to model the interaction among events and a multi-head cross-attention mechanism to integrate the document-aware representation $\mathbf{H}^s$ into event queries $\mathbf{Q}^{event}$. Formally, the m event queries are decoded into m output embeddings $\mathbf{H}^{event}$ by:

$$\mathbf{H}^{event} = \text{Event-Decoder}(\mathbf{Q}^{\text{event}}; \mathbf{H}^{\text{s}}) \quad (3)$$

where $\mathbf{H}^{event} \in \mathbb{R}^{m \times d}$.

Role Decoder. The role decoder is designed to support the filling of all roles in an event in parallel and model the interaction between roles. As the predicted event type t with semantic role types $(r_1, r_2, \ldots, r_n)$, we use n learnable embeddings as the input of the role decoder, which are denoted as event queries $\mathbf{Q}^{role} \in \mathbb{R}^{n \times d}$. Then, the role query embeddings $\mathbf{Q}^{role}$ are fed into the decoder, which has the same architecture as the event decoder. Specifically, the self-attention mechanism can model the relationship among roles, and the cross-attention mechanism can fuse the information of the document-aware candidate argument representations $\mathbf{H}^a$. Formally, the n role queries are decoded into n output embeddings $\mathcal{H}_{role}$ by:

$$\mathbf{H}^{role} = \text{Role-Decoder}(\mathbf{Q}^{role}; \mathbf{H}^a) \quad (4)$$

where $\mathbf{H}^{role} \in \mathbb{R}^{n \times d}$.

Event-to-Role Decoder. To generate diversiform events with relevant arguments for different event queries, an event-to-role decoder is designed to model the interaction between the event queries $\mathbf{H}^{event}$ and the role queries $\mathbf{H}^{role}$:

$$\mathbf{H}^{e2r} = \text{Event2Role-Decoder}(\mathbf{H}^{role}; \mathbf{H}^{event}) \quad (5)$$

where $\mathbf{H}^{e2r} \in \mathbb{R}^{m \times n \times d}$.

2.4 Events Prediction

After the multi-granularity decoding, the m event queries and n role queries are transformed into m predicted events and each of them contains n role embeddings. To filter the spurious event, the m event queries $\mathbf{H}^{event}$ are fed into a feed-forward networks (FFN) to judge each event prediction is non-null or null. Concretely, the predicted event can be obtained by:

$$\mathbf{p}^{event} = \text{softmax}(\mathbf{H}^{event}\mathbf{W}_e) \quad (6)$$

where $\mathbf{W}_e \in \mathbb{R}^{d \times 2}$ is learnable parameters.

Then, for each predicted event with pre-defined roles, the predicted arguments are decoded by filling the candidate indices or the null value with $(N'_a + 1)$-class classifiers:[3]

$$\mathbf{P}^{role} = \text{softmax}(\tanh(\mathbf{H}^{e2r}\mathbf{W}_1 + \mathbf{H}^a\mathbf{W}_2) \cdot \mathbf{v}_1) \quad (7)$$

where $\mathbf{W}_1 \in \mathbb{R}^{d \times d}$, $\mathbf{W}_2 \in \mathbb{R}^{d \times d}$ and $\mathbf{v}_1 \in \mathbb{R}^d$ are learnable parameters, and $\mathbf{P}^{role} \in \mathbb{R}^{m \times n \times (N'_a+1)}$.

After the prediction network, we can obtain the m events $\hat{Y} = (\hat{Y}_1, \hat{Y}_2, \ldots, \hat{Y}_m)$ where each event $\hat{Y}_i = (\mathbf{P}_i^1, \mathbf{P}_i^2, \ldots, \mathbf{P}_i^n)$ contains n predicted arguments with role types. Where $\mathbf{P}_i^j = \mathbf{P}^{role}[i, j, :] \in \mathbb{R}^{(N'_a+1)}$.

2.5 Matching Loss

The main problem for training is that how to assign predicted m events with a series of arguments to the ground truth k events. Inspired by the assigning problem in the operation research (Kuhn, 1955; Munkres, 1957), we propose a matching loss function, which can produce an optimal bipartite matching between predicted and ground-truth events.

Formally, we denote predicted and ground truth events as $\hat{Y} = (\hat{Y}_1, \hat{Y}_2, \ldots, \hat{Y}_m)$ and $Y = (Y_1, Y_2, \ldots, Y_k)$, respectively. Where k is the

[3]Note that we append candidate argument representations $\mathbf{H}^a$ with a learnable embedding to represent the null value.

real number of events in the document and m is fixed size for generated events. Note that $m \geqslant k$. The i-th predicted event is denoted as $\hat{Y}_i = (\mathbf{P}_i^1, \mathbf{P}_i^2, \ldots, \mathbf{P}_i^n)$, where $\mathbf{P}_i^j$ can be calculated by the Equation 7. And the i-th ground truth event is denoted as $Y_i = (r_i^1, r_i^2, \ldots, r_i^n)$, where r_i^j is the candidate argument indix for j-the role type in i-th target event.

To find a bipartite matching between these two sets, we search for a permutation of m elements with the lowest cost:

$$\hat{\sigma} = \underset{\sigma \in \prod(m)}{\operatorname{argmax}} \sum_{i}^{m} \mathcal{C}_{match}(\hat{Y}_{\sigma(i)}, Y_i) \tag{8}$$

where $\prod(m)$ is the space of all m-length permutations and $\mathcal{C}_{match}(\hat{Y}_{\sigma(i)}, Y_i)$ is a pair-wise matching cost between ground truth y_i and a prediction $\hat{Y}_{\sigma(i)}$ with index $\sigma(i)$. By taking into account all of the prediction arguments for roles in an event, we define $\mathcal{C}_{match}(\hat{Y}_{\sigma(i)}, Y_i)$ as:

$$\mathcal{C}_{match}(\hat{Y}_{\sigma(i)}, Y_i) = -\mathbb{1}_{\{\text{judge}_i \neq \phi\}} \sum_{j=1}^{n} \mathbf{P}^j_{\sigma(i)}(r_i^j)) \tag{9}$$

where the judge_i is the judgement of event i to be non-null or null that is calculated by the Equation 6. The optimal assignment $\sigma(i)$ can be computed effectively with the Hungarian algorithm. [4] Then for all pairs matched in the previous step, we define the loss function with negative log-likelihood as:

$$\mathcal{L}(\hat{Y}, Y) = \sum_{i=1}^{m} \mathbb{1}_{\{\text{judge}_i \neq \phi\}} [\sum_{j=1}^{n} -\log \mathbf{P}^j_{\hat{\sigma}(i)}(r_i^j)] \tag{10}$$

Where $\hat{\sigma}$ is the optimal assignment computed in the Equation 8.

2.6 Optimization

During training, we sum the matching loss for events prediction with preconditioned steps before decoding as follows:

$$\mathcal{L}_{all} = \lambda_1 \mathcal{L}_{see} + \lambda_2 \mathcal{L}_{ec} + \lambda_3 \mathcal{L}(Y, \hat{Y}) \tag{11}$$

where $\mathcal{L}_{ae}$ and $\mathcal{L}_{ec}$ are the cross-entropy loss function for sentence-level candidate argument recognition and event type classification, respectively. λ_1, λ_2 and λ_3 are hyper-parameters.

[4] https://en.wikipedia.org/wiki/Hungarianalgorithm

3 Experiments and Analysis

In this section, we present empirical studies to answer the following questions:

1. What is the overall performance of our DE-PPN compared to the state-of-the-art (SOTA) method evaluated on the DEE task?

2. How does DE-PPN perform when facing the arguments-scattering and multi-event challenges in DEE?

3. How does each design of our proposed DE-PPN matter?

4. What is the influence of setting different numbers of the generated events on the results?

3.1 Experimental Setup

Dataset. Following Zheng et al. (2019), we use the ChFinAnn dataset[5] to evaluate our proposed DEE method. The ChFinAnn is a large-scale DEE dataset, which contains 32,040 documents in total and includes five financial event types: *Equity Freeze* (EF), *Equity Repurchase* (ER), *Equity Underweight* (EU), *Equity Overweight* (EO) and *Equity Pledge* (EP).

Evaluation Metrics. For a fair comparison, we adopt the evaluation standard used in Doc2EDAG (Zheng et al., 2019). Specifically, for each predicted event, the most similar ground-truth is selected without replacement to calculate the Precision (P), Recall (R), and F1-measure (F1-score). As an event type often includes multiple roles, micro-averaged role-level scores are calculated as the final DEE metric.

Implementation Details. For a document as input, we set the maximum number of sentences and the maximum sentence length as 64 and 128, respectively. We adopt the basic Transformer, each layer has 768 hidden units, and 8 attention heads, as the encoder and decoder architecture. During training, we employ the AdamW optimizer (Kingma and Ba, 2014) with the learning rate 1e-5 with batch size 16. Testing set performance is chosen by the best development set performance step within 100 epochs. We leave detailed hyper-parameters and additional results in the Appendix.

[5] https://github.com/dolphin-zs/Doc2EDAG/blob/master/Data.zip

Models	EF			ER			EU			EO			EP		
	P	R	F1	P	R	F1	P	R	F1	P	R	F1	P	R	F1
DCFEE-O	66.0	41.6	51.1	84.5	81.8	83.1	62.7	35.4	45.3	51.4	42.6	46.6	64.3	63.6	63.9
DCFEE-M	51.8	40.7	45.6	83.7	78.0	80.8	49.5	39.9	44.2	42.5	47.5	44.9	59.8	66.4	62.9
GreedyDec	**79.5**	46.8	58.9	83.3	74.9	78.9	68.7	40.8	51.2	69.7	40.6	51.3	85.7	48.7	62.1
Doc2EDAG	77.1	64.5	70.2	**91.3**	83.6	87.3	**80.2**	65.0	71.8	**82.1**	69.0	75.0	80.0	**74.8**	77.3
DE-PPN-1	77.8	55.8	64.9	75.6	76.4	76.0	76.4	63.7	69.4	77.1	54.3	63.7	**85.5**	43.0	57.2
DE-PPN	78.2	**69.4**	**73.5**	89.3	**85.6**	**87.4**	69.7	**79.9**	**74.4**	81.0	**71.3**	**75.8**	83.8	73.7	**78.4**

Table 1: Overall event-level precision (P), recall (R) and F1-score (F1) evaluated on the test set.

Models	EF		ER		EU		EO		EP		Avg.		
	S.	M.	S.	M.	S.	M.	S.	M.	S.	M.	S.	M.	S.& M.
DCFEE-O	56.0	46.5	86.7	54.1	48.5	41.2	47.7	45.2	68.4	61.1	61.5	49.6	58.0
DCFEE-M	48.4	43.1	83.8	53.4	48.1	39.6	47.1	42.0	67.0	60.0	58.9	47.7	55.7
GreedyDec	75.9	40.8	81.7	49.8	62.2	34.6	65.7	29.4	**88.5**	42.3	74.8	39.4	60.5
Doc2EDAG	80.0	61.3	**89.4**	68.4	77.4	64.6	79.4	69.5	85.5	72.5	82.3	67.3	76.3
DE-PPN-1	**82.4**	46.3	78.3	53.9	**82.2**	45.6	78.1	39.3	82.8	38.5	80.7	44.7	66.2
DE-PPN	82.1	**63.5**	89.1	**70.5**	79.7	**66.7**	**80.6**	**69.6**	88.0	**73.2**	**83.9**	**68.7**	**77.9**

Table 2: F1-score for all event types and the averaged ones (Avg.) on single-event (S.) and multi-event (M.) sets.

Models	ASR$\leqslant$0.5	0.5$\leqslant$ASR$\leqslant$1	ASR$\geqslant$1
DCFEE-M	65.7	53.5	42.2
Doc2EDAG	78.4	74.4	64.4
DE-PPN	**79.5**	**76.1**	**67.1**

Table 3: Averaged F1-score for different ASR intervals.

3.2 Baselines

We compare our DE-PPN with the SOTA methods as follows: **DCFEE** (Yang et al., 2018) proposed a key-event detection to guide event table filled with the arguments from key-event mention and surrounding sentences. There are two versions of DCFEE: **DCFEE-O** only extracts one event and **DCFEE-M** extracts multiple events from a document. **Doc2EDAG** (Zheng et al., 2019) proposed an end-to-end model for DEE, which transforms DEE as directly filling event tables with entity-based path expending. There is a simple baseline of Doc2EDAG, named **GreedyDec**, which only fills one event table entry greedily. Besides, we further introduce a simple baseline of DE-PPN, named as **DE-PPN-1**, which only generates one event.

3.3 Main results

DE-PPN vs. SOTA. Table 1 shows the comparison between DE-PPN and baseline methods on the test set for each event type. Overall, our proposed model DE-PPN significantly outperforms other baselines and achieves SOTA performance in all event types. Specifically, DE-PPN improves 3.3, 0.1, 2.6, 0.8, 1.1, 1.6 F1-score over the SOTA method, Doc2EDAG, on the event type EF, ER, EU, EO, EP and the average F1-score, respectively. The improved performance indicates that the encoder-decoder generative framework of DE-PPN is effective, which can predict events in parallel with a global optimization for training. Besides, as the baseline of our proposed method, DE-PPN-O can achieve the best performance compared with DCFEE-O and GreedyDec while all of them only predict one event for a document, which also proves the effectiveness of the document-aware end-to-end modeling of DE-PPN.

Results on Arguments-Scattering. To show the extreme difficulty of the arguments-scattering challenge in DEE, we conduct experiments on different scenarios. We introduce an arguments-scattering ratio (ASR) to measure the scatter of arguments in an event for a document. The ASR is calculated by:

$$\mathrm{ASR} = \mathrm{Num}_{\mathrm{ments}}/\mathrm{Num}_{\mathrm{args}} \quad (12)$$

where $\mathrm{Num}_{\mathrm{ments}}$ denotes the number of event mentions (i.e., sentences that contains arguments) and $\mathrm{Num}_{\mathrm{args}}$ denotes the number of arguments. The higher the ASR, the more scattering of the arguments in an event. Table 3 shows the results with the different intervals of ASR. We can observe that it is more difficult to extract scattering arguments as the ASR increase. But DE-PPN still

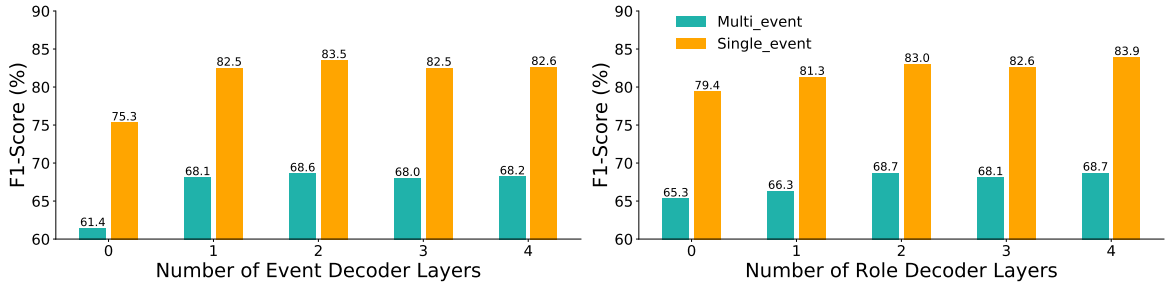

Figure 3: F1-score for performance differences of event decoder and role decoder layers.

maintains the best performance and the results indicate that the encoder-decoder framework can better assemble arguments to the corresponding event across sentences with the parallel prediction and the document-aware representations.

Single-Event vs. Multi-Event. To show the extreme difficulty when arguments-scattering meets multi-events for DEE, we conduct experiments on two scenarios: single-event (i.e., documents contain one event) and multi-event (i.e., documents contain multiple events). Table 2 shows the F1-score on single-event and multi-event sets for each event type and the averaged (Avg.). We can observe that multi-events is extremely challenging as the extraction performance of all models drops significantly. But DE-PPN still improves the average F1-score from 67.3% to 68.7% over the Doc2EDAG. The results demonstrate the effectiveness of our proposed method when handling the challenge of multi-events. This performance improvement benefits from the event decoder which can generate multiple events in parallel and the matching loss function which can perform a global optimization. Besides, the DE-PPN-1 model achieves an acceptable performance on the scenario of single event extraction which demonstrates the effectiveness of our end-to-end model. But DE-PPN-1 only generates one event and cannot deal with the multi-events problem, resulting in low performance on the multi-event sets.

3.4 Ablation Studies

To verify the effectiveness of each component of DE-PPN, we conduct ablation tests on the next variants: 1) *-DocEnc*: removing the Transformer-based document-level encoder, which can support the document-aware information for decoding. 2) *-MultiDec*: replacing the multi-granularity decoder module with simple embedding initialization for event queries and role queries. 3) *-MatchingLoss*: replacing the matching loss function with normal cross-entropy loss. The results are shown in Table 4 and we can observe that: 1) the document-level encoder is of prime importance that enhances the document-aware representations for the generative decoder and contributes +2.6 F1-score on average; 2) the multi-granularity decoder alleviates the challenges of argument-scattering and multi-events by assembling arguments and generating events in parallel, improving by +4.3 F1-score on average. 3) the matching loss function is a very important component for events extraction with +13.4 F1-score improvement which indicates that the matching loss guide a global optimization between predicted and ground-truth events during training.

Model	EF	ER	EU	EO	EP	Avg.
DE-PPN	73.5	87.4	74.4	75.8	78.4	77.9
-DcoEnc	-2.1	-3.4	-1.7	-2.6	-3.2	-2.6
-MultiDec	-5.1	-3.8	-4.3	-4.7	-3.6	-4.3
-MatchingLoss	-9.2	-12.8	-13.1	-17.5	-14.3	-13.4

Table 4: F1-score of ablation studies on DE-PPN variants for each event type and the averaged (Avg.).

3.5 Effect of Different Decoder Layers

To investigate the importance of the multi-granularity decoder, we explore the effect of different layers of the event decoder and the role decoder on the results. Specifically, the number of decoder layers is set to 0,1,2,3 and 4, where 0 means removing this decoder. 1) The effect of different event decoder layers are shown in the left of Figure 3, and our method can achieve the best average F1-score when the number of layers is set to be 2. We conjecture that more layers of the non-autoregressive decoder allow for better modeling the interaction between event queries and generating diversiform events. However, when the layer is set to be large, it is easy to generate redundant events. 2) The

effect of different role decoder layers are shown in the right of Figure 3, and we can observe that the more decoder layers, the better performance on the results. We conjecture that more layers of the decoder with the more self-attention modules allow for better modeling the relationship between event roles and more inter-attention modules allow for integrating information of candidate arguments into roles.

3.6 Effect of Different Generated Sets

For the training and testing process of the DE-PPN, the number of generated events is an important hyperparameter. In this section, we explore the influence of setting different numbers of generated events on the results. We divide the development set into 5 sub-class where each class contains 1,2,3,4 and $\geqslant 5$ events. Table 5 shows the statistics of the documents with different annotated events in the development set. To validate the impact of the number of generated events on the performance, we evaluate DE-PPN with various numbers of generated events: 1, 2, 5, 10, named DE-PPN-1, DE-PPN-2, DE-PPN-5, DE-PPN-10, respectively. The results of DE-PPN with different generated events are shown in Figure 4, which are also compared with the SOTA model Doc2EDAG. We can observe that as the number of events increases, it is more difficult for events prediction, which can be reflected in the decline of all performance. In general, DE-PPN almost achieves the best performance on the average F1-score when the number of generated sets is set to be 5. Besides, there is a performance gap between Doc2EDAG and our method DE-PPN when the number of annotated events is large than 2 in a document. It also demonstrates that our proposed parallel decoder can better handle the challenge of multi-events in DEE.

Number of Events	1	2	3	4	$\geqslant 5$	Total
Number of Documents	2207	609	203	77	91	3187

Table 5: The statistics about the documents annotated with different numbers of events in the development set.

4 Related Work

4.1 Sentence-level Event Extraction

Most work in EE has focused on the sentence level and is based on the benchmark dataset ACE 2005 (Doddington et al., 2004). Many approaches

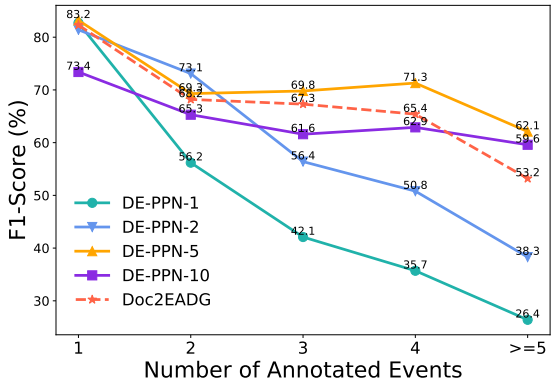

Figure 4: F1-score for performance differences of generated events.

have been proposed to improve performance on this task. These studies are mainly based on hand-designed features (Li et al., 2013; Kai and Grishman, 2015) and neural-based to learn features automatically (Chen et al., 2015; Nguyen et al., 2016; Björne and Salakoski, 2018; Yang et al., 2019; Chan et al., 2019; Yang et al., 2019; Liu et al., 2020). A few methods make extraction decisions beyond individual sentences. Ji and Grishman (2008) and Liao and Grishman (2010) used event type co-occurrence patterns for event detection. Yang and Mitchell (2016) introduced event structure to jointly extract events and entities within a document. Although these approaches make decisions beyond sentence boundary, their extractions are still done at the sentence level.

4.2 Document-level Event Extraction

Many real-world applications need DEE, in which the event information scatters across the whole document. MUC-4 (1992) proposed the MUC-4 template-filling task that aims to identify event role fillers with associated role types from a document. Recent works explore the local and additional context to extract the role fillers by manually designed linguistic features (Patwardhan and Riloff, 2009; Huang and Riloff, 2011, 2012) or neural-based contextual representation (Chen et al., 2020; Du et al., 2020; Du and Cardie, 2020). Recently, Ebner et al. (2020) published the Roles Across Multiple Sentences (RAMS) dataset, which contains annotation for the task of multi-sentence argument linking. A two-step approach (Zhang et al., 2020) is proposed for argument linking by detecting implicit argument across sentences. Li et al. (2021) extend this task and compile a new benchmark dataset

WIKIEVENTS for exploring document-level argument extraction task. Then, Li et al. (2021) propose an end-to-end neural event argument extraction model by conditional text generation. However, these works focused on the sub-task of DEE (i.e., role filler extraction or argument extraction) and ignored the challenge of multi-events.

To simultaneously address both challenges for DEE (i.e., arguments-scattering and multi-events), previous works focus on the ChFinAnn (Zheng et al., 2019) dataset and model DEE as an event table filling task, i.e., filling candidate arguments into predefined event table. Yang et al. (2018) proposed a key-event detection to guide event table filled with the arguments from key-event mention and surrounding sentences. Zheng et al. (2019) transforms DEE into filling event tables following a predefined order of roles with an entity-based path expanding, which achieved the SOTA for DEE. However, these methods suffered from a serial prediction which will lead to error propagation and individual argument prediction.

5 Conclusion and Future Work

In this paper, we propose an encoder-decoder model, DE-PPN, to extract events in parallel from a document. For addressing the challenges (i.e., arguments-scattering and multi-events) in DEE, we introduce a document-level encoder and a multi-granularity decoder to generate events in parallel with document-aware representations. For training the parallel networks, we propose a matching loss function to perform a global optimization. Experimental results show that DE-PPN can significantly outperform SOTA methods especially facing the specific challenges in DEE.

Acknowledgements

We thank the anonymous reviewers for their constructive and insightful comments. This work is supported by the National Natural Science Foundation of China (No. U1936207, No. 61922085 and No. 61806201), Beijing Academy of Artificial Intelligence (No. BAAI2019QN0301), the Key Research Program of the Chinese Academy of Sciences (No. ZDBS-SSW-JSC006), independent research project of National Laboratory of Pattern Recognition and a grant from Ant Group.

References

David Ahn. 2006. The stages of event extraction. In *Proceedings of the Workshop on Annotating and Reasoning about Time and Events*, pages 1–8.

Jari Björne and Tapio Salakoski. 2018. Biomedical event extraction using convolutional neural networks and dependency parsing. In *Proceedings of the BioNLP 2018 workshop*, pages 98–108, Melbourne, Australia. Association for Computational Linguistics.

Yee Seng Chan, Joshua Fasching, Haoling Qiu, and Bonan Min. 2019. Rapid customization for event extraction. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 31–36, Florence, Italy. Association for Computational Linguistics.

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*, pages 811–820, Suzhou, China. Association for Computational Linguistics.

Yubo Chen, Shulin Liu, Xiang Zhang, Kang Liu, and Jun Zhao. 2017. Automatically labeled data generation for large scale event extraction. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 409–419.

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 ACL*.

George R Doddington, Alexis Mitchell, Mark A Przybocki, Lance A Ramshaw, Stephanie M Strassel, and Ralph M Weischedel. 2004. The automatic content extraction (ace) program-tasks, data, and evaluation. In *Lrec*, volume 2, page 1. Lisbon.

Xinya Du and Claire Cardie. 2020. 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, Online. Association for Computational Linguistics.

Xinya Du, Alexander Rush, and Claire Cardie. 2020. Document-level event-based extraction using generative template-filling transformers. *arXiv preprint arXiv:2008.09249*.

Seth Ebner, Patrick Xia, Ryan Culkin, Kyle Rawlins, and Benjamin Van Durme. 2020. Multi-sentence argument linking. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8057–8077, Online. Association for Computational Linguistics.

Jiatao Gu, James Bradbury, Caiming Xiong, Victor OK Li, and Richard Socher. 2018. Non-autoregressive neural machine translation. In *International Conference on Learning Representations*.

Yu Hong, Jianfeng Zhang, Bin Ma, Jianmin Yao, Guodong Zhou, and Qiaoming Zhu. 2011. Using cross-entity inference to improve event extraction. In *Proceedings of the 49th annual meeting of the association for computational linguistics: human language technologies*, pages 1127–1136.

Lifu Huang, Heng Ji, Kyunghyun Cho, Ido Dagan, Sebastian Riedel, and Clare Voss. 2018. Zero-shot transfer learning for event extraction. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2160–2170, Melbourne, Australia. Association for Computational Linguistics.

Ruihong Huang and Ellen Riloff. 2011. Peeling back the layers: detecting event role fillers in secondary contexts. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 1137–1147. Association for Computational Linguistics.

Ruihong Huang and Ellen Riloff. 2012. Bootstrapped training of event extraction classifiers. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 286–295. Association for Computational Linguistics.

Heng Ji and Ralph Grishman. 2008. Refining event extraction through cross-document inference. In *Proceedings of ACL-08: Hlt*, pages 254–262.

Xiang Li Thien Huu Nguyen Kai and Cao Ralph Grishman. 2015. Improving event detection with abstract meaning representation. *ACL-IJCNLP 2015*, page 11.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Harold W Kuhn. 1955. The hungarian method for the assignment problem. *Naval research logistics quarterly*, 2(1-2):83–97.

Qi Li, Heng Ji, and Liang Huang. 2013. Joint event extraction via structured prediction with global features. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 73–82.

Sha Li, Heng Ji, and Jiawei Han. 2021. Document-level event argument extraction by conditional generation. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 894–908, Online. Association for Computational Linguistics.

Shasha Liao and Ralph Grishman. 2010. Using document level cross-event inference to improve event extraction. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 789–797.

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

MUC-4. 1992. Fourth Message Uunderstanding Conference (MUC-4). In *In Proceedings of FOURTH MESSAGE UNDERSTANDING CONFERENCE (MUC4)*, McLean, Virginia.

James Munkres. 1957. Algorithms for the assignment and transportation problems. *Journal of the society for industrial and applied mathematics*, 5(1):32–38.

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

Siddharth Patwardhan and Ellen Riloff. 2009. A unified model of phrasal and sentential evidence for information extraction. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 1-Volume 1*, pages 151–160. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

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

Hang Yang, Yubo Chen, Kang Liu, Yang Xiao, and Jun Zhao. 2018. Dcfee: A document-level chinese financial event extraction system based on automatically labeled training data. In *Proceedings of ACL 2018, System Demonstrations*, pages 50–55.

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

Zhisong Zhang, Xiang Kong, Zhengzhong Liu, Xuezhe Ma, and Eduard Hovy. 2020. A two-step approach

for implicit event argument detection. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7479–7485, Online. Association for Computational Linguistics.

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

A Appendix

In the appendix, we incorporate the following details that are omitted in the main body due to the space limit.

- Section A.1 introduce the Hungarian Algorithm.
- Section A.2 complements additional evaluation results for event classification and candidate arguments extraction.
- Section A.3 show the hyper-parameter setting.

A.1 Hungarian Algorithm

The linear sum assignment problem is also known as minimum weight matching in bipartite graphs. A problem instance is described by a matrix C, where each $C_{i,j}$ is the cost of matching vertex i of the first partite set (a "worker") and vertex j of the second set (a "job"). The goal is to find a complete assignment of workers to jobs of minimal cost.

Formally, let X be a boolean matrix where $X_{i,j} = 1$ if row i is assigned to column j. $C_{i,j}$ is the cost matrix of the bipartite graph. Then the optimal assignment has cost:

$$\min \sum_i \sum_j C_{i,j} X_{i,j} \tag{13}$$

s.t. each row is assignment to at most one column, and each column to at most one row. This function can also solve a generalization of the classic assignment problem where the cost matrix is rectangular. If it has more rows than columns, then not every row needs to be assigned to a column, and vice versa. The method used is the Hungarian algorithm, also known as the Munkres or Kuhn-Munkres algorithm.

A.2 Additional Results

Table 6 shows the results of event type classification and candidate argument extraction. They are the two preceding sub-tasks for decoder to predict events with corresponding arguments in parallel. We can observe that: 1) the document-level event type classification can achieve a good performance which proves that event classification is not a difficult problem in this task. 2) how to assemble candidate arguments to corresponding events is the key challenge for DEE.

	P	R	F1
Equity Freeze	100.0	99.6	99.8
Equity Repurchase	100.0	99.5	99.8
Equity Underweight	98.0	98.1	98.0
Equity Overweight	97.5	94.9	96.1
Equity Pledge	99.5	99.9	99.7
Candidate Argument Recognition	90.0	89.5	89.7

Table 6: Evaluation results of candidate argument extraction and event type classification on the test set.

A.3 Hyperparameter setting

The detail hyperparameter is shown in Table 7

Hyper-parameter	Value
number of generated events	5
Embedding size	768
Hidden size	768
tagging scheme	BIO (Begin, Inside, Other)
Layers of Transformer-1	4
Layers of Transformer-2	4
Layers of event decoder	2
Layers of role decoder	4
Layers of event-to-role decoder	2
Optimizer	AdamW
Learning rate for encoder	$1e^{-5}$
Learning rate for decoder	$2e^{-5}$
Batch size	16
λ_1	0.1
λ_2	0.4
λ_3	0.5
Dropout	0.1
Training epoch	100

Table 7: The hyper-parameter setting.