PESE: Event Structure Extraction using Pointer Network based Encoder-Decoder Architecture

Alapan Kuila
IIT Kharagpur, India
alapan.cse@iitkgp.ac.in

Sudeshna Sarkar
IIT Kharagpur, India
sudeshna@cse.iitkgp.ac.in

Abstract

The task of event extraction (EE) aims to find the events and event-related argument information from the text and represent them in a structured format. Most previous works try to solve the problem by separately identifying multiple substructures and aggregating them to get the complete event structure. The problem with the methods is that it fails to identify all the interdependencies among the event participants (event-triggers, arguments, and roles). In this paper, we represent each event record in a unique tuple format that contains trigger phrase, trigger type, argument phrase, and corresponding role information. Our proposed pointer network-based encoder-decoder model generates an event tuple in each time step by exploiting the interactions among event participants and presenting a truly end-to-end solution to the EE task. We evaluate our model on the ACE2005 dataset, and experimental results demonstrate the effectiveness of our model by achieving competitive performance compared to the state-of-the-art methods.

1 Introduction

Event extraction (EE) from text documents is one of the crucial tasks in natural language processing and understanding. Event extraction deals with the identification of event-frames from natural language text. These event-frames have a complex structure with information regarding event-trigger, event type, event-specific arguments, and event-argument roles. For example,

> In Baghdad, a cameraman **died** when an American tank **fired** on the Palestine hotel.

In this sentence **died** and **fired** are the event triggers for the event types *Die* and *Attack* respectively. The sentence contains entities phrases: *Baghdad, a cameraman, an American tank* and *Palestine hotel.* Some of these entities play a specific role in these mentioned events and termed as event arguments. For event type *Die*, (argument; role) pairs are: (Baghdad; Place), (A cameraman; victim), (American tank; instrument). Whereas, for *Attack* event, (argument; role) pairs are: (Baghdad; Place), (A cameraman; target), (American tank; instrument), and (Palestine Hotel; Target). Apparently, a sentence may contain multiple events; an entity may be shared by multiple event frames; moreover, a specific argument may play different roles in different event frames. Therefore an ideal event extraction system will identify all the trigger words, classify the correct event types, extract all the event-specific arguments and correctly predict the event-argument roles. Each of these subtasks is equally important and challenging.

Most existing works decompose the EE task into these predefined subtasks and later aggregate those outputs to get the complete event frames. Some of these models follow a pipelined approach where triggers and corresponding arguments are identified in separate stages. In contrast, others rely on joint modeling that predicts triggers and relevant arguments simultaneously. However, the pipeline approaches have to deal with error propagation problems, and the joint models have to exploit the information sharing and inter-dependency among the event triggers, arguments, and corresponding roles. The interaction among the event participants are of the following types: 1) inter-event interaction: usually event types in one sentence are interdependent of one another (Chen et al., 2018) 2) intra-event argument interaction: arguments of a specific event-mention have some relationship among themselves (Sha et al., 2016) (Sha et al., 2018) (Hong et al., 2011a) 3) inter-event argument interaction: target entities or arguments shared by two different event mention present in a sentence generally have some inter-dependencies (Hong et al., 2011a) (Nguyen et al., 2016a) 4) event type-role interaction: Each event frame has a distinct set of ar-

Proceedings of the 2nd Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 12th International Joint Conference on Natural Language Processing (Volume 1: Long Papers), pages 1091–1100
November 20–23, 2022. ©2022 Association for Computational Linguistics

gument roles based on its schema definition; hence event type and argument roles have an assiduous relationship. (Xi et al., 2021) 5) argument-role interaction: the event-argument role is dependent on the entity types of the candidate arguments (Xi et al., 2021) as well. Significant efforts have been devoted to exploiting these interactions but despite their promising results, most of these existing systems failed to capture all these inter-dependencies (Xi et al., 2021) (Nguyen and Nguyen, 2019).

In order to exploit the interactions among the event participants mentioned above, we propose a neural network-based sequence to structure learning model that can generate sentence-level event frames from the input sentences. Each event frame holds a (trigger, argument) phrase pair along with corresponding trigger type(event type) and role-label information. Inspired from the models used for joint entity-relation extraction (Nayak and Ng, 2020) (Chen et al., 2021), aspect sentiment triplet extraction (Mukherjee et al., 2021) and semantic role labeling (Fei et al., 2021), we design a **P**ointer network-based **E**vent **S**tructure **E**xtraction (PESE) framework [1] that utilizes the event-argument-role interdependencies to extract the event frames from text. The encoder encodes the input sentence, whereas the decoder identifies an event frame in each time step based on the input sentence encoding and the event frames generated in the previous time steps. The innovation lies in the effectiveness of this type of modeling: 1) instead of decomposing the whole task into separate subtasks, our model can detect the trigger, argument, and role labels together 2)The system is capable of extracting multiple events present in a single sentence by generating each event-tuple in consecutive time steps, 3) the model is also able to extract multiple event-tuples with common trigger or argument phrases and 4)experimental results show that the model can identify the overlapping argument phrases present in the sentence as well. In summary, the contributions of this paper are:

(1) We propose a new representation schema for event frames where each frame contains information regarding an (event, argument) phrase pair.

(2) We present a sentence-level end-to-end event extraction model which exploits the event-argument-role inter-relatedness and tries to find the trigger, argument spans, and corresponding labels within a sentence. The proposed EE system takes a sentence as input and generates all the unique event frames present in that sentence as output.

(3) We have applied our proposed method to the ACE2005 dataset[2] and the experimental results show that our approach outperforms several state-of-the-art baselines models.

2 Event Frame Representation

Given a sentence, our proposed end-to-end EE model extracts all the event-frames present in that sentence. These event frames are the structured representation of the event-specific information: (1) Event trigger phrase, (2) Event type, (3) Argument phrase, (4) Role label. Inside the sentences, each trigger and argument phrase appears as a continuous sequence of words; hence, an effective way to represent these phrases is by their corresponding start and end locations. Therefore in this paper, we represent each event-frame using a 6-tuple structure that stores all the records, as mentioned earlier. The 6-tuple contains: 1) start index of trigger phrase, 2) end index of trigger phrase, 3) event type, 4) start index of argument phrase, 5) end index of argument phrase 6) trigger-argument role label. The start and end index of the trigger phrase(1-2) denotes the event-trigger span, whereas the start and end index of argument phrase(4-5) represent the event-argument span and the other two records(3 and 6) are two labels: event type and role type. Table 1 represents sample sentences and corresponding event frames present in those sentences with their 6-tuple representations. However, there are instances when an event-trigger is present in a sentence without any argument phrase. In order to generalize the event-tuple representation, we concatenate two extra tokens: `[unused1]` and `[unused2]` in front of each sentence with position 1^{st} and 2^{nd} respectively 1. In the absence of an actual argument phrase in the sentence, the `[unused2]` token is used as the dummy argument, and the corresponding start and end index of the argument phrase in the event-tuple are represented by 1, and the role-type is represented by "NA" (see Table 1). The token `[unused1]` is used to indicate the absence of any valid event-trigger word in the sentence.

[1] codes are available at https://github.com/alapanju/PESE.git

[2] https://catalog.ldc.upenn.edu/LDC2006T06

Input Sentence	[unused1] [unused2] Orders went out today to deploy 17,000 U.S. Army soldiers in the Persian Gulf region .
Output Tuple	7 7 Movement:Transport 8 11 Artifact , 7 7 Movement:Transport 13 16 Destination
Input Sentence	[unused1] [unused2] The more they learn about this invasion , the more they learn about this occupation , the less they support it .
Output Tuple	8 8 Conflict:Attack 1 1 NA

Table 1: Event tuple representation for Encoder decoder model

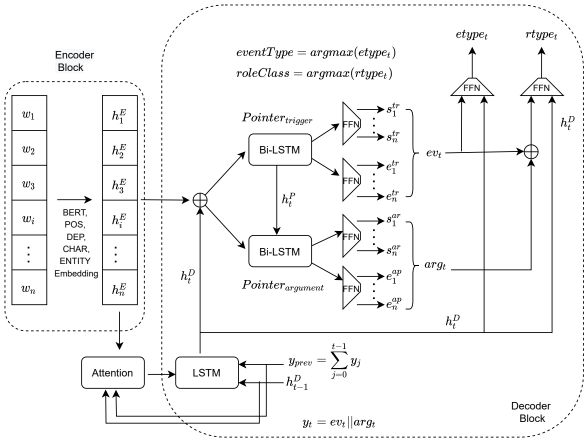

Figure 1: Pointer network based encoder decoder model architecture

2.1 Problem Formulation

To formally define the EE task, first we consider two predefined set E and R where $E \in \{E_1, E_2, E_3, \ldots, E_p\}$ is the set of event types, and $R \in \{R_1, R_2, R_3, \ldots, R_r\}$ is the set of role labels. Here p and r are number of event types and role types respectively. Now, given a sentence $S = [w_1, w_2, w_3, ..., w_n]$ where n is the sentence length and w_i is the ith token, our objective is to extract a set of event-tuples $ET = \{et_i\}_{i=1}^{|ET|}$ where $et_i = [s_i^{tr}, e_i^{tr}, E_i, s_i^{ar}, e_i^{ar}, R_i]$ and $|ET|$ indicates number of event frames present in sentence S. In the ith event-tuple (et_i) representation, s_i^{tr} and e_i^{tr} respectively represent the start and end index of trigger phrase span, E_i indicates the event type of the candidate trigger from set E, s_i^{ar} and e_i^{ar} respectively denote the start and end index of argument phrase span and R_i indicates role-label of the (trigger, argument) pair from set R.

3 Our Proposed EE Framework

We employ a encoder-decoder architecture for the end-to-end EE task. The overview of the model architecture is depicted in Figure 1. The input to our model is a sentence (i.e. a sequence of tokens) and as output, we get a list of event tuples present in that sentence. We use pre-trained BERT (Devlin et al., 2019) at the encoder and LSTM (Hochreiter and Schmidhuber, 1997)-based network at the decoder in our model.

3.1 Sentence Encoding

We use pre-trained BERT model as the sentence encoder to obtain the contextual representation of the tokens. However, part-of-speech (POS) tag information is a crucial feature as most trigger phrases are nouns, verbs or adjectives. Besides, the dependency tree feature (DEP) is another informative clue in sentence-level tasks (Sha et al., 2018). We also use the entity type information (ENT) information (BIO tags) as feature. We combine the POS, DEP, ENT, and character-level features with the BERT embeddings to represent each token in the input sentence. So along with pre-trained BERT embedding we use four other embeddings: 1) POS embeddings $E_{pos} \in \mathbb{R}^{|POS| \times d_{pos}}$ 2) DEP embeddings $E_{dep} \in \mathbb{R}^{|DEP| \times d_{dep}}$ 3) Entity type embeddings $E_{ent} \in \mathbb{R}^{|ENT| \times d_{ent}}$ and 4) character-level embeddings $E_{char} \in \mathbb{R}^{|V_c| \times d_{char}}$. Here, $|POS|$, $|DEP|$, $|ENT|$ and $|V_c|$ indicates respectively the count of unique pos tags, dependency relation tags, entity tags and unique character alphabets. Whereas, d_{pos}, d_{dep}, d_{ent} and d_{char} represents the corresponding dimensions of pos, dependency, entity and character features respectively. Similar to (Chiu and Nichols, 2016) we apply convolution neural network with max-pooling to obtain the character-level feature vector of dimension d_c for each token in the sentence S. All these feature representations are concatenated to get the aggregated vector representation h_i^E for each token w_i present in the sentence S. More specifically, $h_i^E \in \mathbb{R}^{d_h}$ where $d_h = d_{BERT} + d_{pos} + d_{dep} + d_{ent} + d_c$.

3.2 Extraction of Event Frames

Our proposed decoder generates a sequence of event tuples. The decoder comprises sequence-generator LSTM, two pointer networks, and two classification networks. The event frame sequence is generated by the sequence-generator LSTM. The trigger and argument spans of the events are identified by the pointer networks. The classification networks determine the type of event and the trigger-argument role label. Each of these modules is described in greater detail below.

Sequence Generating Network We use an LSTM cell to generate the sequence of the events frame. In each time step t, this LSTM takes attention weighted sentence embedding (e_t) and aggregation of all the previously generated tuple embeddings ($eTup_{prev}$) as input and generates an intermediate hidden representation $h_t^D (\in \mathbb{R}^{d_h})$. To obtain the sentence embedding $e_t \in \mathbb{R}^{d_h}$, we use an attention mechanism depicted in (Bahdanau et al., 2015) where we use both h_{t-1}^D and $eTup_{prev}$ as the query. The hidden state of the decoder-LSTM is represented as:

$$h_t^D = LSTM(e_t \oplus eTup_{prev}, h_{t-1}^D)$$

While generating the present tuple, we consider the previously generated tuple representations with the aim to capture the event-participant's inter-dependencies and to avoid generation of duplicate tuples. The sentence embedding vector e_t is generated by applying attention method depicted later. The aggregated representation of all the event tuples generated before current time step $eTup_{prev} = \sum_{k=0}^{t-1} eTup_k$ where $eTup_0$ is a zero tensor. The event tuple generated at time step t is represented by $eTup_t = tr_t \oplus ar_t$, where tr_t and ar_t are the vector representations of the trigger and entity phrases respectively that are acquired from the pointer networks (depicted later) at time step t. Here, $\oplus$ represents concatenation operation. While generating each event tuple, we consider these previously generated event tuples to capture the event-event inter-dependencies.

Pointer Network for Trigger/Argument Span Detection The pointer networks are used to identify the trigger and argument phrase-span in the source sentence. Each pointer network contains a Bi-LSTM network followed by two feed-forward neural networks. Our architecture contains two such pointer networks to identify the start and end index of the trigger and argument phrases respectively. In each time step t, we first concatenate the intermediate vector h_t^D (obtained from previous LSTM layer) with the hidden vectors h_i^E (obtained from the encoder) and feed them to the Bi-LSTM layer with hidden dimension d_p of the first pointer network. The Bi-LSTM network produces a hidden vector $h_i^{pt} \in \mathbb{R}^{2d_p}$ for each token in the input sentence. These hidden representations are simultaneously passed to two feed-forward networks with a softmax layer to get two normalized scalar values ($\hat{s}_i^{tr}$ and $\hat{e}_i^{tr}$) between 0 and 1 for each token in the sentence. These two values represent the probabilities of the corresponding token to be the start and end index of the trigger phrase of the current event tuple.

$$s_i^{tr} = W_s^1 * h_i^{pt} + b_s^1, \quad \hat{s}^{tr} = softmax(s^{tr})$$
$$e_i^{tr} = W_e^1 * h_i^{pt} + b_e^1, \quad \hat{e}^{tr} = softmax(e^{tr})$$

Here, $W_s^1 \in \mathbb{R}^{2d_p \times 1}$, $W_e^1 \in \mathbb{R}^{2d_p \times 1}$, b_s^1 and b_e^1 represents the weight and bias parameters of the first pointer network.

The second pointer network that extracts the argument phrase of the tuple also contains a similar Bi-LSTM with two feed-forward networks. At each time step, we concatenate the hidden vector h_i^{pt} from the previous Bi-LSTM network with h_t^D and h_i^E and pass them to the second pointer network, which follows similar equations as the first pointer network to obtain $\hat{s}_i^{ar}$ and $\hat{e}_i^{ar}$. These two scalars represent the normalized probability scores of the ith source token to be the start and end index of the argument phrase. We consider feeding the trigger pointer network's output vector to the argument pointer network's input to exploit the trigger-argument inter-dependencies. However, the normalized probabilities $\hat{s}_i^{tr}$, $\hat{e}_i^{tr}$, $\hat{s}_i^{ar}$ and $\hat{e}_i^{ar}$ collected from the two pointer networks are used to get the vector representations of the trigger and argument phrase, ev_t and arr_t:

$$ev_t = \sum_{i=1}^{n} \hat{s}_i^{tr} * h_i^{pt} \oplus \sum_{i=1}^{n} \hat{e}_i^{tr} * h_i^{pt}$$
$$arg_t = \sum_{i=1}^{n} \hat{s}_i^{ar} * h_i^{pa} \oplus \sum_{i=1}^{n} \hat{e}_i^{ar} * h_i^{pr}$$

Feed-Forward Layer for Classification We require two feed-forward neural network-based classification layers to identify the event type, argument type and role label in each event tuple. First, we concatenate the vector representation of trigger phrase ev_t with h_t^D and feed the aggregated vector to the first classification layer followed by a softmax layer to find the correct event type of the detected trigger phrase.

$$eType_t = softmax(W_{tr}(ev_t \oplus h_t^D) + b_{tr})$$

$$\overline{eType_t} = argmax(e\hat{Type_t})$$

Finally, the concatenation of ev_t, arg_t and h_t^D are feed to the second feed-forward network followed by a softmax layer to predict the correct argument-role label while exploiting the event-role and argument-role inter-dependencies.

$$rType_t = softmax(W_r(ev_t \oplus arg_t \oplus h_t^D) + b_r)$$

$$\overline{rType_t} = argmax(r\hat{Type_t})$$

3.3 Training Procedure

To train our model, we minimize the sum of negative log-likelihood loss for identifying the four position-indexes of the corresponding trigger and argument spans and two classification tasks: 1)event type classification and 2) role classification.

$$Loss = -\frac{1}{B \times ET} \sum_{b=1}^{B} \sum_{et=1}^{ET} [\log(s_{b,et}^{tr}, e_{b,et}^{tr}) + log(s_{b,et}^{ar}, e_{b,et}^{ar}) + log(eType_{b,et}) + log(rType_{b,et})]$$

Here, B is the batch size and ET represents maximum number of event-tuples present in a sentence, b indicates bth training instance and et referes to the etth time step. Besides, $s_{*,*}^{*}$, $e_{*,*}^{*}$, $eType_{*,*}$ and $rType_{*,*}$ are respectively represents the normalized softmax score of the true start and end index location of the trigger and entity phrases and their corresponding event type and role label.

3.4 Inference of Trigger/Argument span

At each time step t, the pointer decoder network gives us four normalized scalar scores: $\hat{s}_i^{tr}$, $\hat{e}_i^{tr}$, $\hat{s}_i^{ar}$ and $\hat{e}_i^{ar}$ denoting the probability of ith token to be the start and end index of trigger and argument span respectively. Similarly, for each token in the source sentence S (of length n) we get a set of four probability scores based on which the valid trigger and argument span will be extracted. We identify the start and end position of the trigger and argument phrase such that the aggregated probability score is maximized with the constraint that within an event-tuple the trigger phrase and argument phrase does not have any overlapping tokens and $1 \leqslant b \leqslant e \leqslant n$ where b and e are the start and end position of the corresponding phrase and n is the length of the sentence. First, we choose the beginning(b) and end(e) position index of the trigger phrase such that: $\hat{s}_b^{tr} \times \hat{e}_e^{tr}$ is maximum. Similarly, we select the argument phrase's beginning and end position index so that the extracted argument phrase does not overlap with the event phrase span. Hence, we get four position indexes with their corresponding probability scores. We repeat the whole process, but by interchanging the sequence, i.e., first, the argument span is identified, followed by the trigger phrase span. Thus we will obtain another set of four position indexes with corresponding probability scores. To identify the valid trigger and argument phrase span, we select that index set that gives the higher product of probability scores.

4 Experiments

4.1 Dataset

The ACE2005 corpus used in this paper contains a total of 599 documents. We use the same data split as the previous works (Li et al., 2013). The training data contains 529 documents (14669 sentences), validation data includes 30 documents (873 sentences) and the test data consists of 40 articles (711 sentences). The corpus contains 33 event subtypes, 13 types of arguments, and 36 unique role labels. Here we are dealing with a sentence-level event extraction task i.e., our proposed system finds event-frames based on the information present in the sentences. There are three types of sentences that exist in the dataset:

- Single trigger with no argument: Sentence contains only one event trigger and no argument information.
- Single event and related arguments: Sentence contains only one event trigger and related argument information.
- Multiple event and related arguments: Sentence contains more than one event trigger (of the same or different event types) with corresponding argument phrases. Each of the arguments plays the same or different roles for the mentioned triggers.
- No information: These sentences do not contain any event trigger corresponding to predefined event types.

For preprocessing, tokenization, pos-tagging, and generating dependency parse trees, we use spaCy library[3]. The model variant that achieves the best

[3] https://github.com/explosion/spaCy

performance (F_1 score) in the validation dataset, is considered for final evaluation on the test dataset.

4.2 Parameter Settings

In the encoder section of our model we adopt cased version of pre-trained BERT-base model (Devlin et al., 2019). Similar to Bert base model, the token embedding length(d_{BERT}) is 768. We set the dimension of the POS embedding dimension (d_{pos})= 50, DEP feature embedding dimension (d_{dep})= 50, Entity feature embedding dimension (d_{ent})= 50, character embedding dimension (d_{char} = 50) and character-level token embedding dimension ($d_c = 50$). The CNN layer that is used to extract character-level token embedding has filter size = 3 and consider tokens with maximum length =10. We also set the hidden dimension of the decoder-LSTM (d_h)= 968 and hidden dimension of the Bi-LSTM in pointer networks (d_p)= 968. The model is trained for 40 epochs with batch size 32 and we use Adam optimizer with learning rate 0.001 and weight decay 10^{-5} for parameter optimization. We set dropout probability to 0.50 to avoid overfitting. In our experiments we use P100-PCIE 16GB GPU and total number of parameters used is $\approx 220M$. The model variant with the highest F_1 score on development dataset is selected for evaluation on the test data. We adopt the same correctness metrics as defined by the previous works (Li et al., 2013) (Chen et al., 2015) to evaluate the predicted results.

4.3 Baselines

In order to evaluate our proposed model we compare our performance with some of the SOTA models that we consider as our baseline models:

1. JointBeam (Li et al., 2013): Extract events based on structure prediction by manually designed features.
2. DMCNN (Chen et al., 2015): Extract triggers and arguments using dynamic multi-pooling convolution neural network in pipelined fashion.
3. JRNN (Nguyen et al., 2016b): Exploit bidirectional RNN models and also consider event-event and event -argument dependencies in their model.
4. JMEE (Liu et al., 2018): Use GCN model with highway network and self-attention for joint event and argument extraction.
5. DBRNN (Sha et al., 2018): Add dependency arcs over bi-LSTM network to improve event-extraction.
6. Joint3EE (Nguyen and Nguyen, 2019): Propose to share common encoding layers to enable the information sharing and decode trigger, argument and roles separately.
7. GAIL (Zhang et al., 2019b): Propose an inverse reinforcement learning method using generative adversarial network (GAN).
8. TANL (Paolini et al., 2021): Employ a sequence generation based method for event extraction.
9. TEXT2EVENT (Lu et al., 2021): Propose a sequence to structure network and infuse event schema by constrained decoding and curriculum learning.
10. PLMEE (Yang et al., 2019) Propose a method to automatically generate labelled data and try to overcome role overlap problem in EE task.

5 Results & Discussion

Table 2 reports the overall performance of our proposed model(called PESE) compared to the other state-of-the-art EE models. We show the average scores over 4 runs of the experiment in row PESE$_{avg}$. The row named PESE$_{best}$ describes our best F_1 scores in each subtask. We can see that, in TI, TC and AI task our model outperforms all the baseline models by a significant margin. Besides, for the argument-role classification (ARC) task our model achieves competitive results. The result table deduces some important observations: (1) In the TI task our model PESE$_{avg}$ outperforms all the baseline models and beat the second best model (**PLMEE**) by 6% higher F_1 score. (2) Similarly, in the case of TC our model achieves the best performance by outperforming the second best model (**PLMEE**) by 2.7% higher F_1 score. Moreover, the performance of our model in the trigger classification (TC) task is better than the best models that work specifically on TC subtask (Xie et al., 2021) (Tong et al., 2020). (3) However, the F_1 score of TC is reduced by more than 6% compared to TI in both PESE$_{avg}$ and PESE$_{best}$ which indicates that in some cases, the model can correctly detect the trigger words but fails to identify the proper event types. In the ACE2005 dataset, among 33 event types approximately 50% events appear less than 100 times. This imbalance in the training set may be a reason behind this fall in the F_1 score. (4) In the case of AI, our model achieves the best performance among all the baseline models achieving

Model	Trigger Identify (TI)			Trigger classify (TC)			Argument Identify (AI)			Argument-Role Classify (ARC)		
	P	R	F1	P	R	F1	P	R	F1	P	R	F1
JointBeam	76.9	65.0	70.4	73.7	62.3	67.5	69.8	47.9	56.8	64.7	44.4	52.7
DMCNN*	80.4	67.7	73.5	75.6	63.6	69.1	68.8	51.9	59.1	62.2	46.9	53.5
JRNN	68.5	75.7	73.5	66.0	73.0	69.3	61.4	64.2	62.8	54.2	56.7	55.4
DBRNN		-		74.1	69.8	71.9	71.3	64.5	67.7	66.2	52.8	58.7
JMEE	80.2	72.1	75.9	76.3	71.3	73.7	71.4	65.6	68.4	**66.8**	54.9	**60.3**
Joint3EE	70.5	74.5	72.5	68.0	71.8	69.8	59.9	59.8	59.9	52.1	52.1	52.1
GAIL	76.8	71.2	73.9	74.8	69.4	72.0	63.3	48.1	55.1	61.6	45.7	52.4
PLMEE*	84.8	83.7	84.2	81.0	80.4.4	80.7	71.4	60.1	65.3	62.3	54.2	58.0
TANL	-	-	72.9	-	-	68.4	-	-	50.1	-	-	47.6
TANL$_{multi}$	-	-	71.8	-	-	68.5	-	-	48.5	-	-	48.5
TEXT2EVENT		-		69.6	74.4	71.9		-		52.5	55.2	53.8
PESE$_{avg}$	**95.3**	**85.7**	**90.2**	**88.3**	**78.8**	**83.4**	**73.1**	65.5	**68.9**	61.9	56.2	58.4
PESE$_{best}$	**96.1**	**86.1**	**90.6**	**89.4**	**79.5**	**84**	**74.1**	**66.6**	**69.8**	63.3	**57.3**	59.3

Table 2: Performance comparison of our model against the previous state-of-the-art methods. "*" marked refers to the pipeline models and the remainings follow the joint learning approach

an average F_1 score of 68.9%. In the ACE2005 dataset, the maximum length of an argument is 38 whereas the maximum length of a trigger is just 7. It seems that the arguments with a long sequence of words and overlapping entities make the AI task more complex compared to the TI task where event triggers are mostly one or two words long. (5) In the ARC task, our proposed model achieves an average F_1 score of 58.4% and is positioned third among all the reported baseline models. Our best result PESE$_{best}$ yields F_1 score of 59.3% and only 1% less than the best result (**JMEE**). However, without the infusion of any event-ontology information, we consider this end-to-end performance quite promising. To further explore our model's effectiveness, we do some comparative experiments on the test dataset and report the performance on both single-event and multi-event scenarios in Table 3.

Item	Model	Count = 1	Count >1
TC	JMEE	75.2	72.7
	JRNN	75.6	64.8
	DMCNN	74.3	50.9
	PESE	**82.6**	**84.1**
AI	DBRNN	59.9	69.5
	PESE	**65.3**	**71.4**
Argument Overlap	BERD	-	60.1
	PESE	-	**74.3**
ARC	JMEE	**59.3**	57.6
	DMCNN	54.6	48.7
	DBRNN	54.6	60.9
	PESE	54.1	**61**

Table 3: Performance of our model with varied number of event records.

5.1 Multiple Event Scenario:

Similar to previous works (Liu et al., 2018) (Xie et al., 2021), we divide the test sentences based on the number of event-triggers present and separately perform an evaluation on those sentences. In both single and multi-trigger scenarios, the model performs greater than 90% in event type identification task. Interestingly, in the case of trigger classification (TC) also, the model performs comparatively better in multi-trigger instances, which presumes the effectiveness of our model in capturing the inter-event dependencies inside sentences.

5.2 Shared Argument Scenario

We also investigate our model's performance on the shared argument scenarios. In the ACE2005 dataset, an event instance may contain multiple arguments, or an argument phrase can be shared by multiple event instances. Compared to **DBRNN**, our model performs better in both single-argument and multi-argument scenarios.

5.3 Overlapping Argument Phrases

There are instances where parts of an entity phrase are considered as different arguments. For example, *former Chinese president* is an *Person* type argument whereas *Chinese* is an *GPE* type argument. When all the arguments inside a sentence are distinct, our model achieves 80.6% F_1 score in argument phrase identification. Alternatively, in the presence of overlapping arguments, the F_1 score is 74.3%, which is quite better than the results reported by BERD model (Xi et al., 2021).

5.4 Identifying Multiple Roles

Our model yields F_1 score of 54.1% when each event mention has only one argument-role record within a sentence. In the presence of multiple argument-role information, the F_1 score is 61%. All the results are reported in Table 3 Similar to (Yang et al., 2019), we also consider the cases when one specific argument has single or multiple role information inside a sentence. For single role type, the model achieves 82.4% F_1 score, and for multiple role instances, the corresponding F_1 score is 54.7%.

5.5 Ablation Study

To investigate the effects of external features employed in our model, we report the ablation study observations in Table 4. We see that entity-type information is very critical for end-to-end event extraction. It improves the F1 score on each subtask very significantly. The quantitative scores also validate the use of pos-tag and dependency-tag features. The use of character-level features also gives us tiny improvements in the model performance.

Model variation	F1-score			
	TI	TC	AI	ARC
PESE model	**90.2**	**83.4**	**68.9**	**58.4**
- gold std. entity feat	84.7	77.7	62.6	51.5
- pos tag feat	86.9	79.8	66.1	55.2
- dep feat	87.4	81.1	66.9	55.7
- char feat	89.7	81.9	67.1	56.3
- all external feat	82.3	75.9	61.3	49.9

Table 4: Ablation of external features on model performance.

6 Related Works

Based on the ACE2005 guidelines the task of EE is the composition of three to four subtasks corresponding to different aspects of the event definition (Nguyen and Nguyen, 2019). A large number of prior works on EE only focus on some specific subtasks like: event detection (Nguyen and Grishman, 2015) (Xie et al., 2021) (Tong et al., 2020) or argument extraction (Wang et al., 2019) (Zhang et al., 2020) (Ma et al., 2020). The models that are capable of extracting the complete event structure are categorized in mainly two ways: (1) pipelined-approach (Ahn, 2006) (Ji and Grishman, 2008) (Hong et al., 2011b) (Huang and Riloff, 2012) (Chen et al., 2015) (Yang et al., 2019) and (2) joint modeling approach (McClosky et al., 2011) (Li et al., 2013) (Yang and Mitchell, 2016) (Liu et al., 2018) (Zhang et al., 2019a) (Zheng et al., 2019) (Nguyen and Verspoor, 2019). Recently, methods like question-answering (Du and Cardie, 2020) (Li et al., 2020), machine reading comprehension (Liu et al., 2020), zero shot learning (Huang et al., 2018) are also used to solve the EE problem. Some of the recent works that follow sequence generation approach for event extraction also achieve promising results (Paolini et al., 2021) (Du et al., 2021). Among the previous methods the closest to our approach is TEXT2EVENT (Lu et al., 2021) that also generates the event structure from sentences in end-to-end manner. But they generates the event representations in token by token format that means in each time step the model generates one single token. Whereas our model generates one single event frame per time step which is more realistic in end-to-end event structure extraction.

7 Conclusion

In this paper, we present a joint event extraction model that captures the event frames from text, exploiting intra-event and inter-event interactions in an end-to-end manner. Unlike other methods that consider EE as a token classification problem or sequence labeling problem, we propose a sequence-to-tuple generation model that extracts an event-tuple containing trigger, argument, and role information in each time step. The experimental results indicate the effectiveness of our proposed approach. In the future, we plan to use cross-sentence context in our model and infuse event ontology information to improve our performance.

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, Sydney, Australia. Association for Computational Linguistics.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473.

Yubo Chen, Liheng Xu, Kang Liu, Daojian Zeng, and Jun Zhao. 2015. Event extraction via dynamic multi-pooling convolutional neural networks. In *ACL*.

Yubo Chen, Hang Yang, Kang Liu, Jun Zhao, and Yantao Jia. 2018. Collective event detection via a hierarchical and bias tagging networks with gated multi-level attention mechanisms. In *EMNLP*.

Yubo Chen, Yunqi Zhang, Changran Hu, and Yongfeng Huang. 2021. Jointly extracting explicit and implicit relational triples with reasoning pattern enhanced binary pointer network. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5694–5703, Online. Association for Computational Linguistics.

Jason P.C. Chiu and Eric Nichols. 2016. Named entity recognition with bidirectional LSTM-CNNs. *Transactions of the Association for Computational Linguistics*, 4:357–370.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. *ArXiv*, abs/1810.04805.

X. Du and Claire Cardie. 2020. Event extraction by answering (almost) natural questions. In *EMNLP*.

X. Du, Alexander M. Rush, and Claire Cardie. 2021. Grit: Generative role-filler transformers for document-level event entity extraction. In *EACL*.

Hao Fei, Fei Li, Bobo Li, and Dong-Hong Ji. 2021. Encoder-decoder based unified semantic role labeling with label-aware syntax. In *AAAI*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*.

Yu Hong, Jianfeng Zhang, Bin Ma, Jianmin Yao, Guodong Zhou, and Qiaoming Zhu. 2011a. 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, Portland, Oregon, USA. Association for Computational Linguistics.

Yu Hong, Jianfeng Zhang, Bin Ma, Jianmin Yao, Guodong Zhou, and Qiaoming Zhu. 2011b. Using cross-entity inference to improve event extraction. In *ACL*.

Lifu Huang, Heng Ji, Kyunghyun Cho, and Clare R. Voss. 2018. Zero-shot transfer learning for event extraction. In *ACL*.

Ruihong Huang and Ellen Riloff. 2012. Modeling textual cohesion for event extraction. In *AAAI*.

Heng Ji and Ralph Grishman. 2008. Refining event extraction through cross-document inference. In *Proceedings of ACL-08: HLT*, pages 254–262, Columbus, Ohio. Association for Computational Linguistics.

Fayuan Li, Weihua Peng, Y. Chen, Quan Wang, Lu Pan, Yajuan Lyu, and Yong Zhu. 2020. Event extraction as multi-turn question answering. In *FINDINGS*.

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, Sofia, Bulgaria. Association for Computational Linguistics.

Jiancai Liu, Yubo Chen, Kang Liu, Wei Bi, and Xiaojiang Liu. 2020. Event extraction as machine reading comprehension. In *EMNLP*.

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

Yaojie Lu, Hongyu Lin, Jin Xu, Xianpei Han, Jialong Tang, Annan Li, Le Sun, Meng Liao, and Shaoyi Chen. 2021. Text2Event: Controllable sequence-to-structure generation for end-to-end event extraction. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 2795–2806, Online. Association for Computational Linguistics.

Jie Ma, Shuai Wang, Rishita Anubhai, Miguel Ballesteros, and Yaser Al-Onaizan. 2020. Resource-enhanced neural model for event argument extraction. In *FINDINGS*.

David McClosky, Mihai Surdeanu, and Christopher D. Manning. 2011. Event extraction as dependency parsing. In *ACL*.

Rajdeep Mukherjee, Tapas Nayak, Yash Butala, Sourangshu Bhattacharya, and Pawan Goyal. 2021. PASTE: A tagging-free decoding framework using pointer networks for aspect sentiment triplet extraction. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 9279–9291, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Tapas Nayak and Hwee Tou Ng. 2020. Effective modeling of encoder-decoder architecture for joint entity and relation extraction. In *Proceedings of The Thirty-Fourth AAAI Conference on Artificial Intelligence (AAAI)*.

Dat Quoc Nguyen and Karin M. Verspoor. 2019. End-to-end neural relation extraction using deep biaffine attention. In *ECIR*.

Thien Huu Nguyen, Kyunghyun Cho, and Ralph Grishman. 2016a. Joint event extraction via recurrent neural networks. In *NAACL*.

Thien Huu Nguyen, Kyunghyun Cho, and Ralph Grishman. 2016b. 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, San Diego, California. Association for Computational Linguistics.

Thien Huu Nguyen and Ralph Grishman. 2015. Event detection and domain adaptation with convolutional neural networks. In *ACL*.

Trung Minh Nguyen and Thien Huu Nguyen. 2019. One for all: Neural joint modeling of entities and events. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33(01):6851–6858.

Giovanni Paolini, Ben Athiwaratkun, Jason Krone, Jie Ma, Alessandro Achille, Rishita Anubhai, Cicero Nogueira dos Santos, Bing Xiang, and Stefano Soatto. 2021. Structured prediction as translation between augmented natural languages. In *9th International Conference on Learning Representations, ICLR 2021*.

Lei Sha, Jing Liu, Chin-Yew Lin, Sujian Li, Baobao Chang, and Zhifang Sui. 2016. Rbpb: Regularization-based pattern balancing method for event extraction. In *ACL*.

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 *AAAI*.

Meihan Tong, Bin Xu, Shuai Wang, Yixin Cao, Lei Hou, Juanzi Li, and Jun Xie. 2020. Improving event detection via open-domain trigger knowledge. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5887–5897, Online. Association for Computational Linguistics.

Xiaozhi Wang, Ziqi Wang, Xu Han, Zhiyuan Liu, Juan-Zi Li, Peng Li, Maosong Sun, Jie Zhou, and Xiang Ren. 2019. Hmeae: Hierarchical modular event argument extraction. In *EMNLP*.

Xiangyu Xi, Wei Ye, Shikun Zhang, Quanxiu Wang, Huixing Jiang, and Wei Wu. 2021. Capturing event argument interaction via A bi-directional entity-level recurrent decoder. 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 210–219. Association for Computational Linguistics.

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

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.

Sen Yang, Dawei Feng, Linbo Qiao, Zhigang Kan, and Dongsheng Li. 2019. Exploring pre-trained language models for event extraction and generation. In *ACL*.

Junchi Zhang, Yanxia Qin, Yue Zhang, Mengchi Liu, and Donghong Ji. 2019a. Extracting entities and events as a single task using a transition-based neural model. In *IJCAI*.

Tongtao Zhang, Heng Ji, and Avirup Sil. 2019b. Joint entity and event extraction with generative adversarial imitation learning. *Data Intelligence*, 1:99–120.

Zhisong Zhang, X. Kong, Zhengzhong Liu, Xuezhe Ma, and Eduard H. Hovy. 2020. A two-step approach for implicit event argument detection. In *ACL*.

Shun Zheng, Wei Cao, Wei Xu, and Jiang Bian. 2019. Doc2edag: An end-to-end document-level framework for chinese financial event extraction. In *EMNLP*.