Capturing Event Argument Interaction via A Bi-Directional Entity-Level Recurrent Decoder

Xi Xiangyu[1,2], **Wei Ye**[1,†], **Shikun Zhang**[1,†], **Quanxiu Wang**[3], **Huixing Jiang**[2], **Wei Wu**[2]
[1] National Engineering Research Center for Software Engineering, Peking University, Beijing, China
[2] Meituan Group, Beijing, China
[3] RICH AI, Beijing, China
{xixy,wye,zhangsk}@pku.edu.cn

Abstract

Capturing interactions among event arguments is an essential step towards robust event argument extraction (EAE). However, existing efforts in this direction suffer from two limitations: 1) The argument role type information of contextual entities is mainly utilized as training signals, ignoring the potential merits of directly adopting it as semantically rich input features; 2) The argument-level sequential semantics, which implies the overall distribution pattern of argument roles over an event mention, is not well characterized. To tackle the above two bottlenecks, we formalize EAE as a Seq2Seq-like learning problem for the first time, where a sentence with a specific event trigger is mapped to a sequence of event argument roles. A neural architecture with a novel Bi-directional Entity-level Recurrent Decoder (BERD) is proposed to generate argument roles by incorporating contextual entities' argument role predictions, like a word-by-word text generation process, thereby distinguishing implicit argument distribution patterns within an event more accurately.

1 Introduction

Event argument extraction (EAE), which aims to identify the entities serving as event arguments and classify the roles they play in an event, is a key step towards event extraction (EE). For example, given that the word "fired" triggers an *Attack* event in the sentence "In Baghdad, a cameraman died when an American tank fired on the Palestine Hotel" , EAE need to identify that "Baghdad", "cameraman", "American tank", and "Palestine hotel" are arguments with *Place*, *Target*, *Instrument*, and *Target* as roles respectively.

Recently, deep learning models have been widely applied to event argument extraction and achieved significant progress(Chen et al., 2015; Nguyen et al., 2016; Sha et al., 2018; Yang et al., 2019; Wang et al., 2019b; Zhang et al., 2020; Du and Cardie, 2020). Many efforts have been devoted to improving EAE by better characterizing argument interaction, categorized into two paradigms. The first one, named **inter-event argument interaction** in this paper, concentrates on mining information of the target entity (candidate argument) in the context of other event instances (Yu et al., 2011; Nguyen et al., 2016), e.g., the evidence that a *Victim* argument for the *Die* event is often the *Target* argument for the *Attack* event in the same sentence. The second one is **intra-event argument interaction**, which exploits the relationship of the target entity with others in the same event instance (Yu et al., 2011; Sha et al., 2016, 2018). We focus on the second paradigm in this paper.

Despite their promising results, existing methods on capturing intra-event argument interaction suffer from two bottlenecks.

(1) The argument role type information of contextual entities is underutilized. As two representative explorations, dBRNN (Sha et al., 2018) uses an intermediate tensor layer to capture latent interaction between candidate arguments; RBPB (Sha et al., 2016) estimates whether two candidate argument belongs to one event or not, serving as constraints on a Beam-Search-based prediction algorithm. Generally, these works use the argument role type information of contextual entities as **auxiliary supervision signals for training** to refine input representation. However, one intuitive observation is that the argument role types can be utilized straightforwardly as semantically rich **input features**, like how we use entity type information. To verify this intuition, we conduct an experiment on ACE 2005 English corpus, in which CNN (Nguyen and Grishman, 2015) is utilized as a baseline. For an entity, we incorporate

†Corresponding authors.

Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, pages 210–219
August 1–6, 2021.

the ground-truth roles of its contextual arguments into the baseline model's input representation, obtaining model CNN(w. role type). As expected, CNN(w. role type) outperforms CNN significantly as shown in Table 1[1].

Model	P	R	F_1
CNN	57.8	55.0	56.3
CNN(w. role type)	59.8	60.3	60.0

Table 1: Experimental results of CNN and its variant on ACE 2005.

The challenge of the method lies in knowing the ground-truth roles of contextual entities in the inference (or testing) phase. That is one possible reason why existing works do not investigate in this direction. Here we can simply use predicted argument roles to approximate corresponding ground truth for inference. We believe that the noise brought by prediction is tolerable, considering the stimulating effect of using argument roles directly as input.

(2) The distribution pattern of multiple argument roles within an event is not well characterized. For events with many entities, distinguishing the overall appearance patterns of argument roles is essential to make accurate role predictions. In dBRNN (Sha et al., 2018), however, there is no specific design involving constraints or interaction among multiple prediction results, though the argument representation fed into the final classifier is enriched with synthesized information (the tensor layer) from other arguments. RBPB (Sha et al., 2016) explicitly leverages simple correlations inside each argument pair, ignoring more complex interactions in the whole argument sequence. Therefore, we need a more reliable way to learn the sequential semantics of argument roles in an event.

To address the above two challenges, we formalize EAE as a Seq2Seq-like learning problem (Bahdanau et al., 2014) of mapping a sentence with a specific event trigger to a sequence of event argument roles. To fully utilize both left- and right-side argument role information, inspired by the bi-directional decoder for machine translation (Zhang et al., 2018), we propose a neural architecture with a novel Bi-directional Entity-level Recurrent Decoder (BERD) to generate event argument roles entity by entity. The predicted argument role of an entity is fed into the decoding module for the next or previous entity recurrently like a text generation process. In this way, BERD can identify candidate arguments in a way that is more consistent with the implicit distribution pattern of multiple argument roles within a sentence, similar to text generation models that learn to generate word sequences following certain grammatical rules or text styles.

The contributions of this paper are:

1. We formalize the task of event argument extraction as a Seq2Seq-like learning problem for the first time, where a sentence with a specific event trigger is mapped to a sequence of event argument roles.

2. We propose a novel architecture with a Bi-directional Entity-level Recurrent Decoder (BERD) that is capable of leveraging the argument role predictions of left- and right-side contextual entities and distinguishing argument roles' overall distribution pattern.

3. Extensive experimental results show that our proposed method outperforms several competitive baselines on the widely-used ACE 2005 dataset. BERD's superiority is more significant given more entities in a sentence.

2 Problem Formulation

Most previous works formalize EAE as either a word-level sequence labeling problem (Nguyen et al., 2016; Zeng et al., 2016; Yang et al., 2019) or an entity-oriented classic classification problem (Chen et al., 2015; Wang et al., 2019b). We formalize EAE as a Seq2Seq-like learning problem as follows. Let $S = \{w_1, ..., w_n\}$ be a sentence where n is the sentence length and w_i is the i-th token. Also, let $E = \{e_1, ..., e_k\}$ be the entity mentions in the sentence where k is number of entities. Given that an event triggered by $t \in S$ is detected in ED stage , EAE need to map the sentence with the event to a sequence of argument roles $R = \{y_1, ..., y_k\}$, where y_i denotes the argument role that entity e_i plays in the event.

3 The Proposed Approach

We employ an encoder-decoder architecture for the problem defined above, which is similar to most Seq2Seq models in machine translation (Vaswani et al., 2017; Zhang et al., 2018), automatic text summarization (Song et al., 2019; Shi et al., 2021), and speech recognition (Tüske et al., 2019; Hannun et al., 2019) from a high-level perspective.

[1]In the experiment we skip the event detection phase and directly assume all the triggers are correctly recognized.

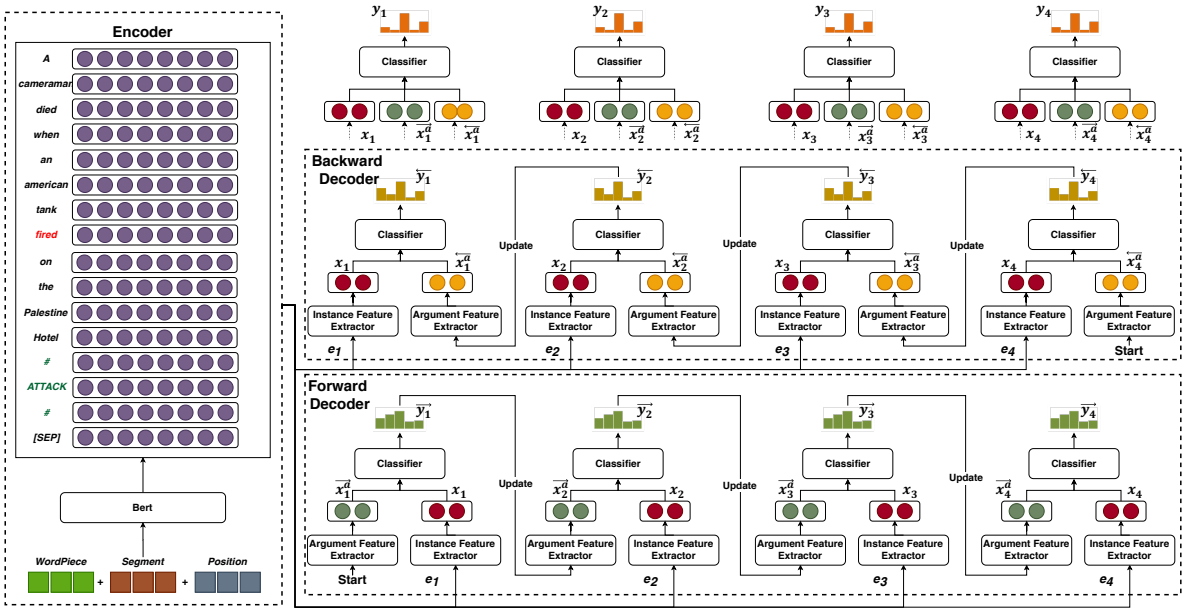

Figure 1: The detailed architecture of our proposed approach. The figure depicts a concrete case where a sentence contains an *Attack* event (triggered by "fired") and 4 candidate arguments $\{e_1, e_2, e_3, e_4\}$. The encoder on the left converts the sentence into intermediate continuous representations. Then the forward decoder and backward decoder generates the argument roles sequences in a left-to-right and right-to-left manner (denoted by $\overrightarrow{\boldsymbol{y}_i}$ and $\overleftarrow{\boldsymbol{y}_i}$) respectively. A classifier is finally adopted to make the final prediction $\boldsymbol{y}_i$. The forward and backward decoder shares the instance feature extractor and generate the same instance representation $\boldsymbol{x_i}$ for i-th entity. The histogram in green and brown denotes the probability distribution generated by forward decoder and backward decoder respectively. The orange histogram denotes the final predictions. Note that the histograms are for illustration only and do not represent the true probability distribution.

In particular, as Figure 1 shows, our architecture consists of an encoder that converts the sentence S with a specific event trigger into intermediate vectorized representation and a decoder that generates a sequence of argument roles entity by entity. The decoder is an entity-level recurrent network whose number of decoding steps is fixed, the same as the entity number in the corresponding sentence. On each decoding step, we feed the prediction results of the previously-processed entity into the recurrent unit to make prediction for the current entity. Since the predicted results of both left- and right-side entities can be potentially valuable information,we further incorporate a bidirectional decoding mechanism that integrates a forward decoding process and a backward decoding process effectively.

3.1 Encoder

Given the sentence $S = (w_1, ..., w_n)$ containing a trigger $t \in S$ and k candidate arguments $E = \{e_1, ..., e_k\}$, an encoder is adopted to encode the word sequence into a sequence of continuous representations as follows[2],

$$\boldsymbol{H} = (\boldsymbol{h}_1, ..., \boldsymbol{h}_n) = F(w_1, ..., w_n) \quad (1)$$

where $F(\cdot)$ is the neural network to encode the sentence. In this paper, we select BERT (Devlin et al., 2019) as the encoder. Considering representation $\boldsymbol{H}$ does not contain event type information, which is essential for predicting argument roles. We append a special phrase denoting event type of t into each input sequence, such as "# ATTACK #".

3.2 Decoder

Different from traditional token-level Seq2Seq models, we use a bi-directional entity-level recurrent decoder (BERD) with a classifier to generate a sequence of argument roles entity by entity. BERD consists of a forward and backward recurrent decoder, which exploit the same recurrent unit architecture as follows.

3.2.1 Recurrent Unit

The recurrent unit is designed to explicitly utilize two kinds of information: (1) the instance infor-

[2]The trigger word can be detected by any event detection model, which is not the scope of this paper.

mation which contains the sentence, event, and candidate argument (denoted by S, t, e); and (2) contextual argument information which consists of argument roles of other entities (denoted by A). The recurrent unit exploits two corresponding feature extractors as follows:

Instance Feature Extractor. Given the representation $\boldsymbol{H}$ generated by encoder, dynamic multi-pooling (Chen et al., 2015) is then applied to extract max values of three split parts, which are decided by the event trigger and the candidate argument. The three hidden embeddings are aggregated into an instance feature representation $\boldsymbol{x}$ as follows:

$$\begin{aligned}
[\boldsymbol{x}_{1,p_t}]_i &= \max\{[\boldsymbol{h}_1]_i, ..., [\boldsymbol{h}_{pt}]_i\} \\
[\boldsymbol{x}_{p_t+1,p_e}]_i &= \max\{[\boldsymbol{h}_{p_t+1}]_i, ..., [\boldsymbol{h}_{pe}]_i\} \\
[\boldsymbol{x}_{p_e+1,n}]_i &= \max\{[\boldsymbol{h}_{p_e+1}]_i, ..., [\boldsymbol{h}_n]_i\} \\
\boldsymbol{x} &= [\boldsymbol{x}_{1,p_t}; \boldsymbol{x}_{p_t+1,p_e}; \boldsymbol{x}_{p_e+1,n}]
\end{aligned} \tag{2}$$

where $[\cdot]_i$ is the i-th value of a vector, p_t, p_e are the positions of trigger t and candidate argument e [3].

Argument Feature Extractor. To incorporate previously-generated arguments, we exploit CNN network to encode the instance with arguments information as follows.

Input. Different from Chen et al. (2015) where input embedding of each word consists of its word embedding, position embedding, and event type embedding, we append the embedding of argument roles into the input embedding for each word by looking up the vector A, which records argument role for each token in S. In A, tokens of previously-predicted arguments are assigned with the generated labels, tokens of the candidate entity e are assigned with a special label "To-Predict", and the other tokens are assigned with label *N/A*.

Convolution. The convolution layer is applied to encode the word sequence into hidden embeddings:

$$(\boldsymbol{h}_1^a, , ..., \boldsymbol{h}_n^a) = \text{CNN}(w_1, ..., t, ..., e, ..., w_n) \tag{3}$$

where the upperscript a denotes argument.

Pooling. Max-pooing operation is then applied to extract the argument feature $\boldsymbol{x}^a$ as follows,

$$\boldsymbol{x}^a = \text{MaxPooling}(\boldsymbol{h}_1^a, ..., \boldsymbol{h}_n^a) \tag{4}$$

We concatenate the instance feature representation $\boldsymbol{x}$ and the argument feature representation $\boldsymbol{x}^a$ as the input feature representation for the argument role classifier, and estimate the role that e plays in the event as follows:

$$\begin{aligned}
\boldsymbol{p} &= f(\boldsymbol{W}[\boldsymbol{x}; \boldsymbol{x}^a] + \boldsymbol{b}) \\
\boldsymbol{o} &= \text{Softmax}(\boldsymbol{p})
\end{aligned} \tag{5}$$

where $\boldsymbol{W}$ and $\boldsymbol{b}$ are weight parameters. $\boldsymbol{o}$ is the probability distribution over the role label space.

For the sake of simplicity, in rest of the paper we use $\text{Unit}(S, t, e, A)$ to represent the calculation of probability distribution $\boldsymbol{o}$ by recurrent unit with S, t, e, A as inputs.

3.2.2 Forward Decoder

Given the sentence S with k candidate arguments $E = \{e_1, ..., e_k\}$, the forward decoder exploits above recurrent unit and generates the argument roles sequence in a left-to-right manner. The conditional probability of the argument roles sequence is calculated as follows:

$$P(R|E, S, t) = \prod_{i=1}^{k} p(y_i|e_i; R_{<i}, S, t) \tag{6}$$

where $R_{<i}$ denotes the role sequence $\{y_1, ..., y_{i-1}\}$ for $\{e_1, ..., e_{i-1}\}$.

For i-th entity e_i, the recurrent unit generates prediction as follows:

$$\overrightarrow{\boldsymbol{y}_i} = \text{Unit}(S, t, e_i, \overrightarrow{A_i}) \tag{7}$$

where $\overrightarrow{\boldsymbol{y}_i}$ denotes the probability distribution over label space for e_i and $\overrightarrow{A_i}$ denotes the contextual argument information of i-th decoding step, which contains previously-predicted argument roles $R_{<i}$. Then we update $\overrightarrow{A_{i+1}}$ by labeling e_i as $g(\overrightarrow{\boldsymbol{y}_i})$ for next step i+1, where $g(\overrightarrow{\boldsymbol{y}_i})$ denotes the label has the highest probability under the distribution $\overrightarrow{\boldsymbol{y}_i}$. The argument feature extracted by recurrent units of forward decoder is denoted as $\overrightarrow{\boldsymbol{x}}_i^a$.

3.2.3 Backward Decoder

The backward decoder is similar to the forward decoder, except that it performs decoding in a right-to-left way as follows:

$$P(R|E, S, t) = \prod_{i=1}^{k} p(y_i|e_i; R_{>i}, S, t) \tag{8}$$

where $R_{>i}$ denotes the role sequence $\{y_{i+1}, ..., y_k\}$ for $\{e_{i+1}, ..., e_k\}$. The probability distribution over

[3] Equation 2 assumes that the entity mention lies after the trigger. If the entity mention lies before the trigger, we switch p_t and p_e in the equation to get a right split.

label space for i-th entity e_i is calculated as follows:

$$\overleftarrow{\boldsymbol{y}_i} = \text{Unit}(S, t, e_i, \overleftarrow{A_i}) \quad (9)$$

where $\overleftarrow{A_i}$ denotes the contextual argument information of i-th decoding step, which contains previously-predicted argument roles $R_{>i}$. We update $\overleftarrow{A_{i-1}}$ by labeling e_i as $g(\overleftarrow{\boldsymbol{y}_i})$ for next step i-1. The argument feature extracted by recurrent units of backward decoder is denoted as $\overleftarrow{\boldsymbol{x}}^a_i$.

3.2.4 Classifier

To utilize both left- and right-side argument information, a classifier is then adopted to combine argument features of both decoders and make final prediction for each entity e_i as follows:

$$\begin{aligned} \boldsymbol{p}_i &= f(\boldsymbol{W_c}[\boldsymbol{x}_i; \overrightarrow{\boldsymbol{x}}^a_i; \overleftarrow{\boldsymbol{x}}^a_i] + \boldsymbol{b_c}) \\ \boldsymbol{y}_i &= \text{Softmax}(\boldsymbol{p}_i) \end{aligned} \quad (10)$$

where $\boldsymbol{y_i}$ denotes the final probability distribution for e_i. $\boldsymbol{W_c}$ and $\boldsymbol{b_c}$ are weight parameters.

3.3 Training and Optimization

As seen, the forward decoder and backward decoder in BERD mainly play two important roles. The first one is to yield intermediate argument features for the final classifier, and the second one is to make the initial predictions fed into the argument feature extractor. Since the initial predictions of the two decoders are crucial to generate accurate argument features, we need to optimize their own classifier in addition to the final classifier.

We use $\overrightarrow{p(y_i|e_i)}$ and $\overleftarrow{p(y_i|e_i)}$ to represent the probability of e_i playing role y_i estimated by forward and backward decoder respectively. $p(y_i|e_i)$ denotes the final estimated probability of e_i playing role y_i by Equation 10. The optimization objective function is defined as follows:

$$\begin{aligned} J(\theta) = -\sum_{S\in D}\sum_{t\in S}\sum_{e_i\in E_S} \alpha \log p(y_i|e_i; R_{\neq i}, S, t) \\ + \beta \log \overrightarrow{p(y_i|e_i)} + \gamma \log \overleftarrow{p(y_i|e_i)} \end{aligned} \quad (11)$$

where D denotes the training set and $t \in S$ denotes the trigger word detected by previous event detection model in sentence S. E_S represents the entity mentions in S. α, β and γ are weights for loss of final classifier, forward decoder and backward decoder respectively.

During training, we apply the teacher forcing mechanism where gold arguments information is fed into BERD's recurrent units, enabling paralleled computation and greatly accelerates the training process. Once the model is trained, we first use the forward decoder with a greedy search to sequentially generate a sequence of argument roles in a left-to-right manner. Then, the backward decoder performs decoding in the same way but a right-to-left manner. Finally, the classifier combines both left- and right-side argument features and make prediction for each entity as Equation 10 shows.

4 Experiments

4.1 Experimental Setup

4.1.1 Dataset

Following most works on EAE (Nguyen et al., 2016; Sha et al., 2018; Yang et al., 2019; Du and Cardie, 2020), we evaluate our models on the most widely-used ACE 2005 dataset, which contains 599 documents annotated with 33 event subtypes and 35 argument roles. We use the same test set containing 40 newswire documents, a development set containing 30 randomly selected documents and training set with the remaining 529 documents.

We notice Wang et al. (2019b) used TAC KBP dataset, which we can not access online or acquire from them due to copyright. We believe experimenting with settings consistent with most related works (e.g., 27 out of 37 top papers used only the ACE 2005 dataset in the last four years) should yield convincing empirical results.

4.1.2 Hyperparameters

We adopt BERT (Devlin et al., 2019) as encoder and the proposed bi-directional entity-level recurrent decoder as decoder for the experiment. The hyperparameters used in the experiment are listed.

BERT. The hyperparameters of BERT are the same as the $\text{BERT}_{\text{BASE}}$ model[4]. We use a dropout probability of 0.1 on all layers.

Argument Feature Extractor. Dimensions of word embedding, position embedding, event type embedding and argument role embedding for each token are 100, 5, 5, 10 respectively. We utilize 300 convolution kernels with size 3. The glove embedding(Pennington et al., 2014) are utilized for initialization of word embedding[5].

Training. Adam with learning rate of 6e-05, $\beta_1 =$

[4]https://github.com/google-research/bert

[5]https://nlp.stanford.edu/projects/glove/

0.9, $\beta_2 = 0.999$, L2 weight decay of 0.01 and learning rate warmup of 0.1 is used for optimization. We set the training epochs and batch size to 40 and 30 respectively. Besides, we exploit a dropout with rate 0.5 on the concatenated feature representations. The loss weights α, β and γ are set to 1.0, 0.5 and 0.5 respectively.

4.2 Baselines

We compare our method against the following four baselines. The first two are state-of-the-art models that separately predicts argument without considering argument interaction. We also implement two variants of DMBERT utilizing the latest inter-event and intra-event argument interaction method, named BERT(Inter) and BERT(Intra) respectively.

1. **DMBERT** which adopts BERT as encoder and generate representation for each entity mention based on dynamic multi-pooling operation(Wang et al., 2019a). The candidate arguments are predicted separately.

2. **HMEAE** which utilizes the concept hierarchy of argument roles and utilizes hierarchical modular attention for event argument extraction (Wang et al., 2019b).

3. **BERT(Inter)** which enhances DMBERT with inter-event argument interaction adopted by Nguyen et al. (2016). The memory matrices are introduced to store dependencies among event triggers and argument roles.

4. **BERT(Intra)** which incorporates intra-event argument interaction adopted by Sha et al. (2018) into DMBERT. The tensor layer and self-matching attention matrix with the same settings are applied in the experiment.

Following previous work (Wang et al., 2019b), we use a pipelined approach for event extraction and implement DMBERT as event detection model. The same event detection model is used for all the baselines to ensure a fair comparison.

Note that Nguyen et al. (2016) uses the last word to represent the entity mention[6], which may lead to insufficient semantic information and inaccurate evaluation considering entity mentions may consist of multiple words and overlap with each other. We sum hidden embedding of all words when collecting lexical features for each entity mention.

[6]Sha et al. (2018) doesn't introduce the details.

Model	P	R	F1
DMBERT	56.9	57.4	57.2
HMEAE	**62.2**	56.6	59.3
BERT(Inter)	58.4	57.1	57.8
BERT(Intra)	56.4	61.2	58.7
BERD	59.1	**61.5**	**60.3**

Table 2: Overall performance on ACE 2005 (%).

4.3 Main Results

The performance of BERD and baselines are shown in Table 2 (statistically significant with $p < 0.05$), from which we have several main observations. (1) Compared with the latest best-performed baseline HMEAE, our method BERD achieves an absolute improvement of 1.0 F1, clearly achieving competitive performance. (2) Incorporation of argument interactions brings significant improvements over vanilla DMBERT. For example, BERT(Intra) gains a 1.5 F1 improvement compared with DMBERT, which has the same architecture except for argument interaction. (3) Intra-event argument interaction brings more benefit than inter-event interaction (57.8 of BERT(Inter) v.s. 58.7 of BERT(Intra) v.s. 60.3 of BERD). (4) Compared with BERT(Inter) and BERT(Intra), our proposed BERD achieves the most significant improvements. We attribute the solid enhancement to BERD's novel seq2seq-like architecture that effectively exploits the argument roles of contextual entities.

4.4 Effect of Entity Numbers

To further investigate how our method improves performance, we conduct comparison and analysis on effect of entity numbers. Specifically, we first divide the event instances of test set into some subsets based on the number of entities in an event. Since events with a specific number of entities may be too few, results on a subset of a range of entity numbers will yield more robust and convincing conclusion. To make the number of events in all subsets as balanced as possible, we finally get a division of four subsets, whose entity numbers are in the range of [1,3], [4,6], [7,9], and [10,] and event quantities account for 28.4%, 28.2%, 25.9%, and 17.5%, respectively.

The performance of all models on the four subsets is shown in Figure 2, from which we can observe a general trend that BERD outperforms other baselines more significantly if more entities appear in an event. More entities usually mean more com-

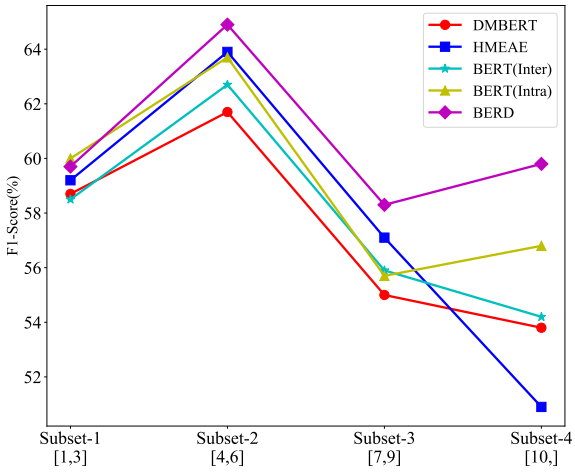

Figure 2: Comparison on four subsets with different range of entity numbers. F1-Score (%) is listed.

plex contextual information for a candidate argument, which will lead to a performance degradation. BERD alleviates degradation better because of its capability of capturing argument role information of contextual entities. We notice that BERT(Intra) also outperforms DMBERT significantly on *Subset-4*, which demonstrates the effectiveness of intra-event argument interaction.

Note that the performance on *Subset-1* is worse than that on *Subset-2*, looking like an outlier. The reason lies in that the performance of the first-stage event detection model on *Subset-1* is much poorer (e.g., 32.8 of F1 score for events with one entity).

4.5 Effect of Overlapping Entity Mentions

Though performance improvement can be easily observed, it is nontrivial to quantitatively verify how BERD captures the distribution pattern of multiple argument roles within an event. In this section, we partly investigate this problem by exploring the effect of overlapping entities. Since there is usually only one entity serving as argument roles in multiple overlapping entities, we believe sophisticated EAE models should identify this pattern. Therefore, we divide the test set into two subsets (*Subset-O* and *Subset-N*) based on whether an event contains overlapping entity mentions and check all models' performance on these two subsets. Table 3 shows the results, from which we can find that all baselines perform worse on *Subset-O*. It is a natural result since multiple overlapping entities usually have similar representations, making the pattern mentioned above challenging to capture. BERD performs well in both *Subset-O* and *Subset-N*, and the superiority on *Subset-O* over baseline is more significant. We attribute it to BERD's capability of distinguishing argument distribution patterns.

Model	*Subset-O*	*Subset-N*
DMBERT	56.4	59.4
HMEAE	58.8	59.6
BERT(Inter)	57.3	58.8
BERT(Intra)	58.5	59.2
BERD	**60.5**	**60.1**

Table 3: Comparison on sentences with and without overlapping entities (*Subset-O* v.s. *Subset-N*). F1-Score (%) is listed.

4.6 Effect of the Bidirectional Decoding

To further investigate the effectiveness of the bidirectional decoding process, we exclude the backward decoder or forward decoder from BERD and obtain two models with only unidirectional decoder, whose performance is shown in the lines of "-w/ Forward Decoder" and "-w/ Backward Decoder" in Table 4. From the results, we can observe that: (1) When decoding with only forward or backward decoder, the performance decreases by 1.6 and 1.3 in terms of F1 respectively. The results clearly demonstrate the superiority of the bidirectional decoding mechanism (2) Though the two model variants have performance degradation, they still outperform DMBERT significantly, once again verifying that exploiting contextual argument information, even in only one direction, is beneficial to EAE.

Model	**P**	**R**	**F1**
DMBERT	56.9	57.4	57.2
BERD	**59.1**	**61.5**	**60.3**
-w/ Forward Decoder	58.0	59.4	58.7
-w/ Backward Decoder	58.3	59.8	59.0
-w/ Forward Decoder x2	56.8	61.1	58.9
-w/ Backward Decoder x2	57.2	61.0	59.1
-w/o Recurrent Mechanism	55.3	60.0	57.4

Table 4: Ablation study on ACE 2005 dataset (%).

Considering number of model parameters will be decreased by excluding the forward/backward decoder, we build another two model variants with two decoders of the same direction (denoted by "-w/ Forward Decoder x2" and "-w/ Backward Decoder x2"), whose parameter numbers are exactly equal to BERD. Table 4 shows that the two enlarged single-direction models have similar performance with their original versions. We can conclude that

the improvement comes from complementation of the two decoders with different directions, rather than increment of model parameters.

Besides, we exclude the recurrent mechanism by preventing argument role predictions of contextual entities from being fed into the decoding module, obtaining another model variant named "-w/o Recurrent Mechanism". The performance degradation clearly shows the value of the recurrent decoding process incorporating argument role information.

4.7 Case Study and Error Analysis

To promote understanding of our method, we demonstrate three concrete examples in Figure 3. Sentence S1 contains a *Transport* event triggered by "sailing". DMBERT and BERT(Intra) assigns *Destination* role to candidate argument "the perilous Strait of Gibraltar ", "the southern mainland" and "the Canary Islands out in the Atlantic", the first two of which are mislabeled. It's an unusual pattern that a *Transport* event contains multiple destinations. DMBERT and BERT(Intra) fail to recognize the information of such patterns, showing that they can not well capture this type of correlation among prediction results. Our BERD, however, leverages previous predictions to generate argument roles entity by entity in a sentence, successfully avoiding the unusual pattern happening.

S2 contains a *Transport* event triggered by "visited", and 4 nested entities exists in the phrase "Ankara police chief Ercument Yilmaz". Since these nested entities share the same sentence context, it is not strange that DMBERT wrongly predicts such entities as the same argument role *Artifact*. Thanks to the bidirectional entity-level recurrent decoder, our method can recognize the distribution pattern of arguments better and hence correctly identifies these nested entities as false instances. In this case, BERD reduces 3 false-positive predictions compared with DMBERT, confirming the results and analysis of Table 3.

As a qualitative error analysis, the last example S3 demonstrates that incorporating previous predictions may also lead to error propagation problem. S3 contains a *Marry* event triggered by "marry". Entity "home" is mislabeled as *Time-Within* role by BERD and this wrong prediction will be used as argument features to identify entity "later in this after", whose role is *Time-Within*. As analyzed in the first case, BERD tends to avoid repetitive roles in a sentence, leading this entity incorrectly being predicted as *N/A*.

S1: *Tens of thousands of destitute Africans* try to enter *Spain* illegally *each year* by crossing *the perilous Strait of Gibraltar* to reach *the southern mainland* or by *sailing* northwest to *the Canary Islands out in the Atlantic*
(the perilous Strait of Gibraltar, N/A, Destination(✗), Destination(✗), N/A) (the southern mainland, N/A, Destination(✗), Destination(✗), N/A) (the Canary Islands out in the Atlantic, Destination, Destination, Destination, Destination)
S2: *Ankara police chief Ercument Yilmaz visited* the *site of the morning blast* but refused to say if *a bomb* had caused the explosion
(Ankara, N/A, Artifact(✗), N/A, N/A) (Ankara police, N/A, Artifact(✗), N/A, N/A) (Ankara police chief, N/A, Artifact(✗), Artifact(✗), N/A) (Ankara police chief Ercument Yilmaz, Artifact, Artifact, Artifact, Artifact)
S3: *Prison authorities* have given the node for *Anwar* to be token *home later in the afternoon* to *marry his eldest daughter*, Nurul Izzah, to *engineer Raja Ahmad Sharir Iskandar* in a traditional *Malay* ceremony, he said
(home, Place, Place, Place, Time-Within(✗)) (later in the afternoon, Time-Within, Time-Within, Time-Within, N/A(✗))

Figure 3: Case study. Entities and triggers are highlighted by green and purple respectively. Each tuple (E,G,P_1,P_2,P_3) denotes the predictions for an entity E with gold label G, where P_1, P_2 and P_3 denotes prediction of DMBERT, BERT(Intra) and BERD respectively. Incorrect predictions are denoted by a red mark.

5 Related Work

We have covered research on EAE in Section 1, related work that inspires our technical design is mainly introduced in the following.

Though our recurrent decoder is entity-level, our bidirectional decoding mechanism is inspired by some bidirectional decoders in token-level Seq2Seq models, e.g., of machine translation (Zhou et al., 2019), speech recognition (Chen et al., 2020) and scene text recognition (Gao et al., 2019).

We formalize the task of EAE as a Seq2Seq-like learning problem instead of a classic classification problem or sequence labeling problem. We have found that there are also some works performing classification or sequence labeling in a Seq2Seq manner in other fields. For example, Yang et al. (2018) formulates the multi-label classification task as a sequence generation problem to capture the correlations between labels. Daza and Frank (2018) explores an encoder-decoder model for semantic role labeling. We are the first to employ a Seq2Seq-like architecture to solve the EAE task.

6 Conclusion

We have presented BERD, a neural architecture with a Bidirectional Entity-level Recurrent Decoder that achieves competitive performance on the task of event argument extraction (EAE). One main characteristic that distinguishes our techniques

from previous works is that we formalize EAE as a Seq2Seq-like learning problem instead of a classic classification or sequence labeling problem. The novel bidirectional decoding mechanism enables our BERD to utilize both the left- and right-side argument predictions effectively to generate a sequence of argument roles that follows overall distribution patterns over a sentence better.

As pioneer research that introduces the Seq2Seq-like architecture into the EAE task, BERD also faces some open questions. For example, since we use gold argument roles as prediction results during training, how to alleviate the exposure bias problem is worth investigating. We are also interested in incorporating our techniques into more sophisticated models that jointly extract triggers and arguments.

Acknowledgements

We thank anonymous reviewers for valuable comments. This research was supported by the National Key Research And Development Program of China (No.2019YFB1405802) and the central government guided local science and technology development fund projects (science and technology innovation base projects) No. 206Z0302G.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Xi Chen, Songyang Zhang, Dandan Song, Peng Ouyang, and Shouyi Yin. 2020. Transformer with bidirectional decoder for speech recognition. pages 1773–1777.

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 ACL and the 7th IJCNLP*, pages 167–176.

Angel Daza and Anette Frank. 2018. A sequence-to-sequence model for semantic role labeling. In *Proceedings of The Third Workshop on Representation Learning for NLP*, pages 207–216.

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, Volume 1 (Long and Short Papers)*, pages 4171–4186.

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

Yunze Gao, Yingying Chen, Jinqiao Wang, and Hanqing Lu. 2019. Gate-based bidirectional interactive decoding network for scene text recognition. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, pages 2273–2276.

Awni Hannun, Ann Lee, Qiantong Xu, and Ronan Collobert. 2019. Sequence-to-sequence speech recognition with time-depth separable convolutions. *Proc. Interspeech 2019*, pages 3785–3789.

Thien Huu Nguyen, Kyunghyun Cho, and Ralph Grishman. 2016. Joint event extraction via recurrent neural networks. In *Proceedings of the 2016 Conference of the NAACL*, pages 300–309.

Thien Huu Nguyen and Ralph Grishman. 2015. Event detection and domain adaptation with convolutional neural networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 365–371.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

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 *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1224–1234.

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 32 AAAI, New Orleans, Louisiana, USA*, pages 5916–5923.

Tian Shi, Yaser Keneshloo, Naren Ramakrishnan, and Chandan K Reddy. 2021. Neural abstractive text summarization with sequence-to-sequence models. *ACM Transactions on Data Science*, 2(1):1–37.

Shengli Song, Haitao Huang, and Tongxiao Ruan. 2019. Abstractive text summarization using lstm-cnn based deep learning. *Multimedia Tools and Applications*, 78(1):857–875.

Zoltán Tüske, Kartik Audhkhasi, and George Saon. 2019. Advancing sequence-to-sequence based speech recognition. In *INTERSPEECH*, pages 3780–3784.

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.

Xiaozhi Wang, Xu Han, Zhiyuan Liu, Maosong Sun, and Peng Li. 2019a. Adversarial training for weakly supervised event detection. In *Proceedings of the 2019 Conference of the NAACL*, pages 998–1008.

Xiaozhi Wang, Ziqi Wang, Xu Han, Zhiyuan Liu, Juanzi Li, Peng Li, Maosong Sun, Jie Zhou, and Xiang Ren. 2019b. 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)*, pages 5781–5787.

Pengcheng Yang, Xu Sun, Wei Li, Shuming Ma, Wei Wu, and Houfeng Wang. 2018. Sgm: Sequence generation model for multi-label classification. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 3915–3926.

S Yang, DW Feng, LB aQiao, ZG Kan, and DS Li. 2019. Exploring pre-trained language models for event extraction and generation. In *Proceedings of the 57th Annual Meeting of the ACL*.

Hong Yu, Zhang Jianfeng, Ma Bin, Yao Jianmin, Zhou Guodong, and Zhu Qiaoming. 2011. Using cross-entity inference to improve event extraction. In *Proceedings of the 49th Annual Meeting of the ACL*, pages 1127–1136. Association for Computational Linguistics.

Ying Zeng, Honghui Yang, Yansong Feng, Zheng Wang, and Dongyan Zhao. 2016. A convolution bilstm neural network model for chinese event extraction. In *NLUIA*, pages 275–287. Springer.

Xiangwen Zhang, Jinsong Su, Yue Qin, Yang Liu, Rongrong Ji, and Hongji Wang. 2018. Asynchronous bidirectional decoding for neural machine translation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32.

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.

Long Zhou, Jiajun Zhang, and Chengqing Zong. 2019. Synchronous bidirectional neural machine translation. *Transactions of the Association for Computational Linguistics*, 7:91–105.