Event Annotation Enhanced Pre-trained Language Model in Event Extraction

Qisen Xi, Yizhi Ren, Guohua Wu, Qiuhua Wang, Lifeng Yuan(✉), and Zhen Zhang(✉)

School of Cyberspace, Hangzhou Dianzi University, Hangzhou 310018, China
{xiqs,renyz,wugh,wangqiuhua,yuanlifeng,zhangzhen}@hdu.edu.cn

Abstract. Event extraction is a crucial task that aims to extract event information in texts. Existing methods usually use pre-trained language models to extract events and have achieved state-of-the-art performance. However, these models do not consider the complexity of the event structure and lack the use of event knowledge. To address these problems, we propose a new framework that integrates event annotation into the pre-trained model explicitly, termd as EABERT. Specifically, event annotations are incorporated into the model input to construct the form "[CLS]sentence[SEP]event annotation[SEP]", which allows the model to encode the semantic relationship between text and event knowledge. To incorporate appropriate event annotations into the model, we further use the bilateral-branch BERT network to train the event type classifier for better accuracy of event annotations. Experiments on the event extraction benchmark dataset (ACE 2005) show that our proposed framework has significantly improved compared to previous methods.

Keywords: Event extraction · Event annotation · Pre-trained language model

1 Introduction

Event extraction (EE) is a challenge extraction task in natural language processing (NLP). It targets to extract structured event information (triggers and arguments) from unstructured text. For example, given the sentence "The EU foreign ministers met hours after U.S. President George W. Bush gave Saddam 48 h to leave Iraq or face invasion.", it contains event detection task to identify event trigger (the word "met") and classify event type (`Meet`). As well as event argument extraction task to identify event arguments ("ministers" and "hours") and classify their argument roles (`Entity` and `Time`). By explicitly capturing the event structure in the text, some AI downstream applications such as financial analysis, public opinion analysis, sentiment analysis can develop.

Many efforts have been devoted to event extraction. Existing methods [1–3] mainly use deep neural networks to follow a supervised learning paradigm for

Y. Chenggang et al. (Eds.): MobiMedia 2022, LNICST 451, pp. 284–297, 2022.
https://doi.org/10.1007/978-3-031-23902-1_22

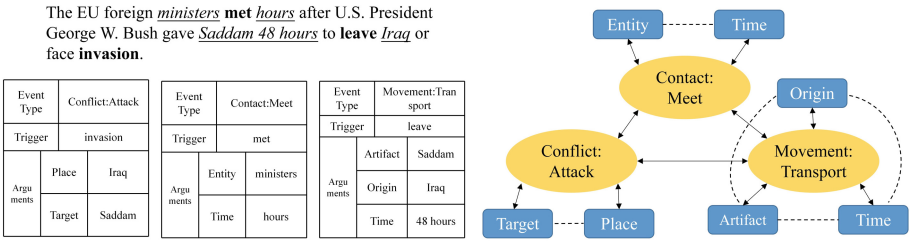

Fig. 1. Event extraction example and event structure

event extraction. Such methods can learn features from raw text and perform well in many publicly available benchmark datasets (ACE 2005, TAC KBP). Inspired by the successful application of pre-trained language models (PLMs) in NLP tasks, many approaches [4–6] have attempted to use PLMs for event extraction and achieved state-of-the-art performance. However, these PLMs-based method focused only on the fine-tuning phase, capturing only the internal pattern of the input text, and did not exploit the event type knowledge.

To alleviate the limitation, several studies have started to integrate event type labels into the PLMs and achieved positive effect. For example, CasEE [7] treats event type encoding as a priori encoding knowledge and uses a cascade decoding strategy for event extraction. GDAP [8] uses prefixed prompt learning to empower the automatic exploitation of event type label semantics on both input and output sides.

Although CasEE and GDAP integrate event type labels into pre-trained models, they do not consider the complex specificity of the event structure itself. The main challenge here is an event structure is far more complicated than a triad (head, relation, tail). As show in Fig. 1, event triggers, event arguments, and event types all have dependencies. Only by enabling the pre-trained model to understand this event structure knowledge better to perform modeling of event features can the event information be better extracted from texts.

To address these issues, we propose EABERT, a new framework with event annotation enhanced BERT for event extraction. Our framework contains two main modules: an event type classifier and an event extraction model. The event type classifier is a multi-label classification model. In particular, to solve the problem of low detection accuracy caused by imbalanced data distribution, we incorporate a bilateral-branch BERT network [9] structure consisting of a conventional branch trained with uniform sampling data and a re-balancing branch trained with reverse sampling according to the number of relevant instances. In event extraction, we use BERT as the basic model. It is worth noting that we add event annotations to BERT. Event annotations are predefined and can be considered as a complete event structure. We construct the token of "[CLS]sentence[SEP]event annotation[SEP]" and feed as the input of BERT. The semantic relationship between the input text and the event annotations is learned using the powerful self-attention mechanism in BERT to obtain the

feature vector. Then, conditional random field (CRF) is used to extract event triggers and arguments.

We tested our model on benchmark dataset (ACE 2005). Our experiments show that our proposed framework achieves excellent performance. The main contributions of the paper can be summarized as follows:

(1) We propose EABERT, a novel framework with event annotation enhanced BERT in event extraction, explicitly integrating event annotation to encode the semantic relationship between text and event knowledge.
(2) We propose an end-to-end bilateral-branch BERT network and design a reverse sampling method according to the number of event samples for exhaustively boosting event type classification accuracy.
(3) We conducted experiments on the event extraction benchmark dataset ACE 2005. The experimental results show that our method achieves significant improvements compared with existing competitive methods.

2 Related Work

Event extraction is a crucial task in NLP. Traditional event extraction methods [10–12] rely on manually extracted features for event extraction, such as lexical and syntactic features. However, because such methods are based on existing NLP tools to do feature extraction, they will be limited by the tools' accuracy. This error propagation will be introduced into subsequent algorithms that cannot be modified. With the development of deep learning, various neural networks have been used for EE, e.g., Chen et al. [1] proposed a dynamic multi-pool convolutional neural networks. Nguyen et al. [2] proposed a joint extraction model JRNN based on RNN. These models can learn text features from the original text. Based on neural networks, researchers (Wadden et al. [3]; Liu et al. [13]; Zhang et al. [14]; Nguyen et al. [15]; Lai et al. [16]; Cui et al. [17]; Zhao et al. [18]) have also mined additional fine-grained information, including entity information, syntactic feature information, and chapter-level information, to enhance the feature representation in neural networks. For example, experiments have demonstrated that these methods achieve better results in public benchmark datasets.

Recently, because of the remarkable success of PLM in the field of NLP, many researchers have proposed PLM-based event extraction models. Yang et al. [4] directly applied BERT to build event extraction models. Wang et al. [5] combined with the dynamic multi-pool operation in DMCNN, BERT is used as the Encoding layer to extract text features for event extraction. Other researchers (Li et al. [6]; Liu et al. [19]; Du et al. [20]; Chen et al. [21]) proposed a BERT-QA-based question-and-answer event extraction method by drawing on the application of the QA approach in named entity recognition. Experimental results show that these event extraction methods based on pre-trained models have become a solution paradigm for event extraction tasks. However, previous work has focused only on the fine-tuning phase to obtain the Embedding form of tokens without

exploiting the event type knowledge. Therefore, researchers have also explored pre-trained model approaches incorporating event type knowledge. For example, CasEE [7] mixes event type encoding into the embedding of BERT, and GDAP [8] adds prefixed event types based on cue learning. However, they still have shortcomings: the complexity of the event structure itself is not taken into account.

3 Method

In this section, we will introduce our EABERT framework in detail. Our framework aims to extract event information in texts. As illustrated in Fig. 2, we divide the event extraction task into two modules: an event type classifier and an event extraction model. In the event type classifier, we input the text to get the event types. In the event extraction module, we first obtain the corresponding event annotations. Then take the sentence and event annotations fed as the BERT input to identify event trigger and arguments and their span (start, end).

Formally: Given a text $\mathcal{D}$,$\boldsymbol{s} = (s_1, s_2, \ldots, s_n)$,$\boldsymbol{s} \in \mathcal{D}$ is a set of sentences. The purpose of our model is to extract the event type $\mathcal{T}$ from $\boldsymbol{s}$, and then construct the input of $(s, event\ annotation)$ to extract the event trigger (Q, S_Q, E_Q) and the event argument (A, S_A, E_A).

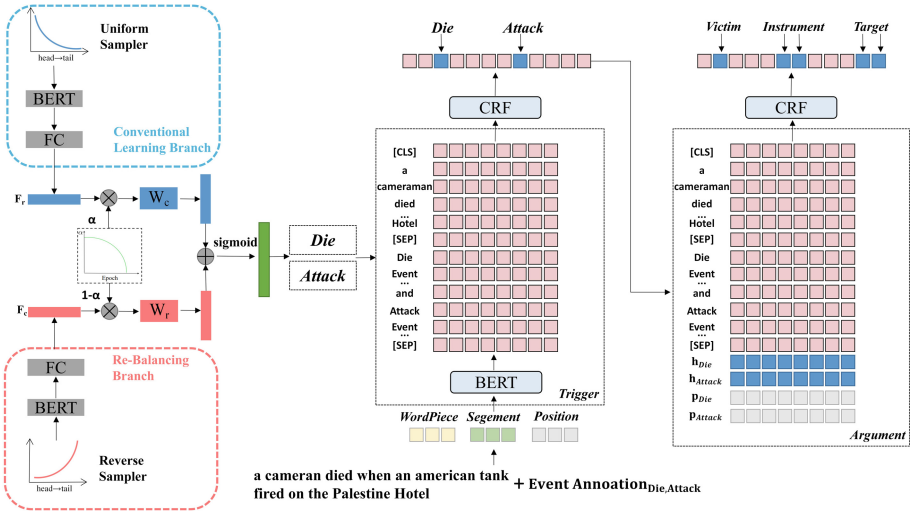

Fig. 2. Our EABERT framewokr for event extraction

3.1 Event Type Classifier

The event type classifier follows the idea of the bilateral-branch network [9]. Our model consists of two branches for representation learning and classifier learning,

called the conventional branch and the re-balancing branch. Both branches use BERT as the backbone network and share all the weights. The input data of the conventional branch comes from a uniform sampler. The re-balancing branch designs a reverse sampling method to sample each class according to the sample size. After that, we use an adaptive learning strategy to balance the bilateral branch learning.

Data Samples. The input data for the conventional branch comes from a uniform sampler, where each sample in the training dataset is sampled only once with equal probability. This sampling method retains the original data distribution, so it facilitates representation learning. While, the re-balancing branch aims to alleviate the long-tailed label distribution. The input of this branch comes from a reverse sampler. For the reversed sampler, the sampling possibility of each class is proportional to the reciprocal of its sample size, i.e., the more instances in a class, the smaller sampling probability that class has. In particular, it can be noted that there are several categories in the event extraction dataset that have only very few instances, which can lead to extremely high sampling probabilities for these categories and thus reduce the classification ability of the model. To solve this problem, we use the average of the sample proportions of all the event categories available as the label proportion threshold to balance the sampling probability of most of the tail labels.

Formally, let N_i denote the number of revelant instances of an event type class i and N_{max} denote the maximum sample number of all event types. The reverse sampler operates in a five-step process: (1)Calculate the proportion of each class to the maximum number of class $w_i = N_{\max}/N_i$; (2)Calculate the average value of class proportion $\overline{w} = \sum_{i=1}^{C} w_i/C$, if $w_i > \overline{w}$, then make the current class proportion $w_i = \overline{w}$; (3)Calculate the sampling probability of class i based on the number of class $P_i = w_i / \sum_{j=1}^{C} w_j$; (4)Randomly select a class according to P_i; (5)Uniformly pick up a sample from class i with replacement. A batch of paired training data can be obtained by repeating this uniform and reverse sampling process.

Adaptive Learning. To avoid overfitting the tail labels and insufficient training on the head labels, we use an adaptive learning strategy to reconcile the bilateral branch learning. In concretely, by setting the adaptive learning parameter α to control the feature weights learned by the conventional and the re-balancing branches. In particular, α is automatically generated based on the training epoch T. The calculation of α can be expressed as follows: $\alpha = 1 - (T/T_{\max})^2$, where T_{max} is the total epoch of training, T is the current epoch. This approach allows to α exhibit a parabolic form of decay as the training period increases. This decay allows the model to focus on learning the conventional branches for most of the period until the final period when it shifts its training attention to the re-balancing branches.

In the training phase, let $\boldsymbol{s}$ denote a training sample and $\boldsymbol{t} \in \{1, 2, 3, 4 \ldots, N\}$ is its corresponding class, where N is the number of classes. For the bilateral

branches, we obtain a pair of samples (s_1^c, t_1^c) and (s_1^r, t_1^r) as input data using uniform sampling and reverse sampling for each branch, respectively. Then, the sampled data are fed into their corresponding branches to obtain the feature vectors f_c and f_r through a fully connected layer. Setting the adaptive learning parameter α to weight the control conventional branch features $f_c \in R^D$ and re-balancing branch features $f_r \in R^D$, i.e., αf_c and $(1-\alpha)f_r$. Then, the two weighted feature vectors are sent into the classifiers $W_c \in R^{n \times D}$ and $W_r \in R^{n \times D}$ and the outputs are integrated by element addition. The final logits output is obtained by sigmoid activation function:

$$\hat{t}_i = \sigma\left(\alpha W_c f_c + (1-\alpha) W_r f_r\right) \tag{1}$$

Finally, we apply a weighted binary cross-entropy loss, denoted by $E(.,.)$, for optimization of the pairwise inputs:

$$L = \alpha E\left(\hat{t}_i, t_i^c\right) + (1-\alpha)\, E\left(\hat{t}_i, t_i^r\right) \tag{2}$$

During inference, we take the event proportion as α:

$$\alpha = \frac{\sum_{i=1}^{C} N_i}{N_{none}} \tag{3}$$

Test samples are fed into two branches with α and $1-\alpha$. Then, we obtained the predicted event type labels.

3.2 Event Extraction Model

Our event extraction model is based on BERT and relies on event annotations. In the following, we will provide details on obtaining the event annotation and details on the event extraction model.

Event Annotation. In the ACE2005 event guideline[1], each event type is given a corresponding event annotation. Examples are shown in Table 1. Event annotations explain well the conditions under when an event occurs, the state and the associated event arguments, etc. Such event annotations can be well regarded as a complete event structure, i.e., an event contains event types, event trigger words and event arguments.

Event Trigger Extraction. Event trigger extraction aims to predict whether a token triggers an event or not and give the span S, i.e., the start index and the end index. The input of the model follows the standard BERT input forms, i.e., WordPiece embedding, positional embedding, and segment embedding. Specifically, to enable our model to learn the event semantic knowledge better, we focus on adding event annotations to the model.

[1] All label annotations are available at: https://www.ldc.upenn.edu/sites/www.ldc.upenn.edu/files/english-events-guidelines-v5.4.3.pdf.

Table 1. Event type and their corresponding event annotation.

Event type	Event annotation
LIFE:BE-BORN	A BE-BORN Event occurs whenever a PERSON Entity is given birth to
LIFE:MARRY	MARRY Events are official Events, where two people are married under the legal definition
MOVEMENT:TRANSPORT	A TRANSPORT Event occurs whenever an ARTIFACT (WEAPON or VEHICLE) or a PERSON is moved from one PLACE (GPE, FACILITY, LOCATION) to another

To achieve this goal, we utilize the SEP separator to construct the input of text and event annotations:

$$[\textbf{CLS}]\langle\textbf{sentence}\rangle[\textbf{SEP}]\langle\textbf{event annotation}\rangle[\textbf{SEP}]$$

where [CLS] is a special classification token for BERT, [SEP] is a special token to denote separation, and ⟨sentence⟩ is the tokenized input sentence. ⟨event annotation⟩ denotes the event annotation corresponding to the event type. It is worth noting that in practical cases, there are situations where there is a single sentence with multiple events. We use "AND" to splice the event annotations of multiple events. Since the input includes both the original sentence and the event annotation, the segment id of the original sentence is set to 0. The segment id of the event annotation is set to 1.

Then we get the contextualized representations of each token for trigger extraction with BERT. Formally, given a sentence $\boldsymbol{x} = w_1, w_2, w_3, \ldots, w_{|x|}$ and corresponding event annotation $\boldsymbol{S_{annotation}} = e_{s1}, e_{s2}, e_{s3}, \ldots, e_{sn}$.

We produce the input for trigger extractor as:

$$\text{Input} = [CLS]w_1, w_2, \ldots w_{|x|}[SEP]e_{s1}, e_{s2}, \ldots, e_{sn}[SEP] \tag{4}$$

Via BERT, we learn the hidden representations:

$$h_{[CLS]}, h_1^w, \ldots, h_{|x|}^w, h_{[SEP]}, h_1^e, \ldots, h_n^e = \text{BERT}(\text{Input}) \tag{5}$$

where $h_{[CLS]}$ and $h_{[SEP]}$ is the hidden state of [CLS] and [SEP].h_i^w is the hidden state of the i-th input token and h_i^e is the hidden state of the corresponding event annotation.

In this paper, we use a linear CRF layer in the prediction phase for trigger labels. The hidden layer representation of the text after BERT encoding is fed to the CRF layer to learn the interdependencies between the output trigger labels and then find the best sequence of trigger labels.

Following Lafferty et al. [22], CRF defines a transition matrix $\mathbf{A}$ and use a score $A_{i,j}$ to model the transition from i^{th} label to the j^{th} label. The scores $H_{i,t}$ of the martix is the score output by the BERT network, for the sentence $[x]_1^N$ and for the i^{th} label, at the t^{th} word. The score of a sequence of trigger label

$[t]_1^N$ for a particular sequence of words $[x]_1^N$ is the sum of the transfer fraction and network fraction efficiently calculated using dynamic programming:

$$S\left([x]_1^N, [t]_1^N\right) = \sum_{i=1}^{N} \left([A]_{[t]_{i-1},[t]_i,} + H_{[t]_i,i}\right) \tag{6}$$

The probability of predicting a sequence of triggers can be obtained by using the softmax function to do global normalization of all possible sequences:

$$P\left(T|X\right) = \frac{e^{S(X,T)}}{\sum_{\tilde{\mathrm{T}}\in T_X} e^{S\left(X,\tilde{\mathrm{T}}\right)}} \tag{7}$$

where T_x denotes the sequence of all possible trigger prediction labels corresponding to X.

Event Argument Extraction. Given the identified trigger for a specific event type, event argument extraction aims to identify the arguments spans(start index and end index) and classify them into their corresponding roles.

The event argument extraction uses the exact hidden layer representation as to the trigger extraction, but it needs to know which token constitutes the event trigger. Therefore, when constructing the input to the argument extraction module, the corresponding token and position encoding of the trigger is embedded and concatenated with the original hidden layer. The format of the input is:

$$h_{[CLS]}, h_1^w, \ldots, h_{|x|}^w, h_{[SEP]}, h_1^e, \ldots, h_n^e, h_1^t, \ldots, h_n^t, h_1^p, \ldots, h_n^p \tag{8}$$

where h_i^t is the corresponding trigger embedding and h_i^p is the corresponding trigger position embedding.

We use the CRF layer proposed in trigger word extraction for the prediction of the argument sequences. The difference is that we use the feature vector mentioned in Eq. 8 as the input to the CRF.

Joint Training. The event extraction process consists of two parts: one is to extract the event trigger, and the other is to extract the event arguments. There is a high correlation between triggers and arguments in the event extraction domain. Inspired by the idea of multi-task learning, this paper uses joint learning for trigger extraction and argument extraction.

Specifically, the event trigger extraction training loss can be expressed as:

$$L_{Tri} = -\sum_{x\in X} \log\left(P\left(T_x|S_x\right)\right) \tag{9}$$

where $\mathcal{X}$ denotes the set of all sentences in the training data, S_x denotes the input sequence corresponding to sentence $\boldsymbol{x}$, T_x and denotes the sequence of trigger prediction corresponding to sentence $\boldsymbol{x}$.

Similarly, the event argument extraction training loss can be expressed as:

$$L_{Arg} = -\sum_{x \in X} \log\left(P\left(A_x | S_x\right)\right) \tag{10}$$

Then, we set the weight of the trigger extraction task loss to 1, and set λ to regulate the weight occupied by the argument extraction task loss, and the joint loss L of the two tasks can be expressed as: $L = L_{Tri} + \lambda L_{Arg}$, where $\lambda \in [0, 1]$.

4 Expriment

4.1 Datasets and Evaluation Metric

We conducted experiments on dataset ACE 2005. ACE 2005 is the most widely used dataset in the field of event extraction. ACE 2005 consists of 8 main events and 33 sub-events. To comply with previous work, we use the same data split as the previous work [1,11,23]. This data split takes the test set with 40 newswire documents, while 30 other documents as the validation set and the remaining 529 documents to be the training set.

Following previous work [1,4,11], we use the following criteria to evaluate the correctness of each predicted event mention: (1) A trigger is correct if its event subtype and offsets match those of a reference trigger. (2) An argument is correctly identified if its event subtype and offsets match those of any of the reference argument mentions. (3) An argument is correctly classified if its event subtype, offsets and argument role match those of any of the reference argument mentions. Finally, we use Precision (P), Recall (R) and F1 scores (F1) as the evaluation metrics.

4.2 Implementation Details

Our implementation is in Pytorch. The BERT base model(uncased) from Hugging Face consists of 12 layers, 768 hidden units and 12 attention heads. In the event type classifier, the MLP consists of two layers with a hidden size of 768 and yields an output of 34 dimensions to predict the probability of the input sentence being assigned to the corresponding 34 classes. In event extraction, We set the sampling probability threshold w is 98.7 and α is 0.77. In event extraction, the trigger MLP consists of two layers with the hidden size being 768 and yields an output of 68 dimensions. The argument MLP yields an output of 58 dimensions. The batch size is 16. The learning rate is set as $2 \times 10^{-}5$. ADAM is the optimizer. All experiments are conducted on an NVIDIA RTX3090 GPU.

4.3 Overall Evaluation Results

We compare our method performance to a number of prior competitive models: **DMCNN** [1] adopts firstly dynamic multi-pooling CNN to extract sentence-level features automatically; **JRNN** [2] proposes a joint framework based on bidirectional RNN for event extraction; **DYGIE++** [3] is a BERT-based framework that models text spans and captures within-sentence and cross-sentence context; **BERT-QA** [6] is a BERT-based model converting event extraction into a QA task; **CasEE** [7] uses a cascade decoding strategy to model the relationship between event types and trigger words and argument elements for event extraction based on the BERT model; **GDAP** [8] takes event types as prefix prompt information to generate events based on T5 model.

Table 2. Overall performance on ACE 2005.

	Trigger identification			Trigger classification			Argument identification			Argument classification		
	P	R	F	P	R	F	P	R	F	P	R	F
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	71.9	66.0	73.0	69.3	61.4	64.2	62.8	54.2	56.7	55.4
BERT-CRF	72.3	80.5	76.2	67.7	75.4	71.3	61.7	45.3	52.2	58.8	45.6	51.3
DYGIE++(ens)	–	–	76.5	–	–	73.6	–	–	55.4	–	–	52.5
BERT-QA	74.3	77.4	75.8	71.1	73.7	72.3	58.0	50.7	54.1	56.9	49.8	53.1
CasEE	78.0	79.8	78.8	74.7	76.8	75.7	62.5	48.7	54.7	59.2	48.4	53.2
GDAP	–	–	–	66.1	75.3	70.4	–	–	–	47.3	59.1	52.6
EABERT(ours)	77.5	82.2	**79.8**	75.4	**80.1**	**77.7**	67.1	51.2	58.1	**64.9**	50.3	**56.7**
EABERT-GOLD	**87.7**	**86.5**	**87.1**	**86.3**	**85.1**	**85.7**	**72.4**	54.7	62.3	**67.8**	51.5	**58.5**

The preformance of all methods on corpus is shown in Table 2. The table reveals that:

(1) In trigger identification and classification, the F1 scores are 79.8% and 77.7%. Our model has significant improvements over the basic neural-based methods (i.e., CNN and RNN). Compared to the same BERT-based classification model DYGIE++, it gains 4.1% on trigger classification F1 score. Compared to the BERT-based QA models BERT-QA, it gains 5.4% on trigger classification. Notably, our model improves 7.3% and 2% on trigger classification compared to the GDAP and CasEE models with the addition of event types. The results demonstrate the effectiveness of EABERT on event detection.

(2) While the improvement in argument extraction is not so obvious. This is probably due to the more rigorous evaluation metric we have taken and the difficulty of the argument extraction task. Compared with the BERT-based models DYGIE++ and BERT-QA, EABERT can achieve better results on the argument extraction task - the F1 score of argument identification is

3.4% higher than that of BERT-QA, and the F1 score of classification is 1.3% and 0.7%. Compared to the GDAP and CasEE, it gains 4.1% and 3.5% on arguments classification.

To further understand our proposed EABERT framework, we conducted experiments on the model EABERT-GOLD using gold annotations. Compared with EABERT, EABERT-GOLD achieved significant progress on the event extraction task, reaching SOTA results: 87.1%, 85.7%, 62.3%, 58.5%, respectively on F1 scores for trigger identification, trigger classification, argument identification and argument classification. It further illustrates that practical event annotation can substantially improve the performance of event extraction models.

4.4 Effectiveness of Event Annotations

To demonstrate the effectiveness of our incorporated event annotations, we constructed two EABERT variants: (1) GloVe event annotation embedding: we initialized the event annotation embedding with GloVe and then spliced it with the input text encoded using BERT. The F1 score dropped by 0.9%,2.3%,0.7%,1.6% on TI, TC, AI, AC, respectively. The results show that the improvement in our EABERT comes from understanding the event annotation. Using the self-attention mechanism, BERT can encode the semantic relationship between text and event annotations well. (2) Event Type Label Name: Replace the event annotations with the name of the event type label. The F1 score drops by 0.5%, 0.6%, 0.7%, 0.7%, respectively, indicating that the label name alone contains less semantic knowledge than the event annotation, which corresponds to a complete knowledge of the event structure.

Table 3. The performance of the EABERT variants. The values in table are F1 scores on test sets.

	TI	TC	AI	AC
EABERT	**79.8**	**77.7**	**58.1**	**56.7**
−GEAE	78.9(↓0.9)	75.4(↓2.3)	57.4(↓0.7)	55.1(↓1.6)
−ETLN	79.3(↓0.5)	77.1(↓0.6)	57.4(↓0.7)	56.0(↓0.7)

4.5 Effect of w and α

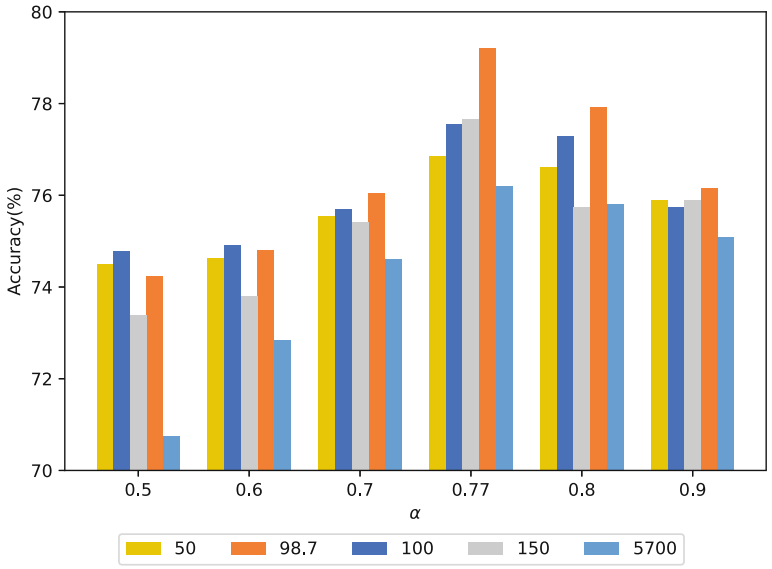

Fig. 3. Event type classifier performance on ACE2005 with different α and w

In this section, we conduct experiments on the ACE 2005 corpus to demonstrate the role of the label proportion threshold w in reverse sampler and the weight parameter α in the inference phase on the performance of the event classifier. To achieve this goal, we design relevant experimental parameters for comparison.

According to Sect. 3.1 and Eq. 3, we can calculate the label proportion threshold w to be 98.7 and the weight parameter α to be 0.77. We selected W from {50,98.7,100,150,5700[2]} and α from {0.5,0.6,0.7,0.77,0.8,0.9}. Figure 3 shows that the event classifier has the highest accuracy of 79.2% when W is selected as 98.7 and α is selected as 0.77, which is an improvement of 4.1% compared to the standard BERT multi-label classifier. This shows that our proposed data enhancement strategy of inverse sampling in bilateral-branch BERT network is very useful for the task of event classification.

5 Conclusion

In this paper, we propose a new framework that integrates event annotation into the BERT explicitly, termd as EABERT. We further use a bilateral-branch network to train the event type classifier to get the appropriate event annotations. Our method allows the model to encode the semantic relationship between text

[2] 5700 is the maximum proportion calculated according to Sect. 3.1.

and event knowledge. To demonstrate the effectiveness of the proposed framework, we systematically conducted a series of experiments on the widely used benchmark dataset ACE 2005. The experimental results show that our proposed method performs better than the previous methods.

In the future, we will: (1) explore how to effectively use the hierarchical information among event types to generate optimal event annotations;(2) explore how to make good use of this event knowledge more effectively.

References

1. Chen, Y., Xu, L., Liu, K., Zeng, D., Zhao, J.: Event extraction via dynamic multi-pooling convolutional neural networks. In: Proc. of ACL, pp. 167–176 (2015)
2. Nguyen, T.H., Cho, K., Grishman, R.: Joint event extraction via recurrent neural networks. In: Proc. of NAACL, pp. 300–309 (2016)
3. Wadden, D., Wennberg, U., Luan, Y., Hajishirzi, H.: Entity, relation, and event extraction with contextualized span representations. In: Proc. of EMNLP (2019)
4. Yang, S., Feng, D., Qiao, L., Kan, Z., Li, D.: Exploring pre-trained language models for event extraction and generation. In: Proc. of ACL, pp. 5284–5294 (2019)
5. Wang, X., Han, X., Liu, Z., Sun, M., Li, P.: Adversarial training for weakly supervised event detection. In: Proc. of NAACL, pp. 998–1008 (2019)
6. Li, F., et al.: Event extraction as multi-turn question answering. In: Proc. of EMNLP Findings, pp. 829–838 (2020)
7. Sheng, J., et al.: CasEE: a joint learning framework with cascade decoding for overlapping event extraction. In: Proc. of ACL Findings (2021)
8. Si, J., Peng, X., Li, C., Xu, H., Li, J.: Generating disentangled arguments with prompts: a simple event extraction framework that works. arXiv preprint arXiv:2110.04525 (2021)
9. Zhou, B., Cui, Q., Wei, X., Chen, Z.: BBN: bilateral-branch network with cumulative learning for long-tailed visual recognition. In: Proc. of CVPR, pp. 9719–9728 (2020)
10. Hong, Y., Zhang, J., Ma, B., Yao, J., Zhou, G., Zhu, Q.: Using cross-entity inference to improve event extraction. In: Proc. of ACL, pp. 1127–1136 (2011)
11. Li, Q., Ji, H., Huang, L.: Joint event extraction via structured prediction with global features. In: Proc. of ACL, pp. 73–82 (2013)
12. Li, Q., Ji, H., Hong, Y., Li, S.: Constructing information networks using one single model. In: Proc. of EMNLP, pp. 1846–1851 (2014)
13. Liu, S., Chen, Y., Liu, K., Zhao, J.: Exploiting argument information to improve event detection via supervised attention mechanisms. In: Proc. of ACL, pp. 1789–1798 (2017)
14. Zhang, Y., Xu, G., Wang, Y., Liang, X., Wang, L., Huang, T.: Empower event detection with bi-directional neural language model. Knowl. Based Syst. **167**, 87–97 (2019)
15. Nguyen, T.M., Nguyen, T.H.: One for all: Neural joint modeling of entities and events. In: Proc. of AAAI, pp. 6851–6858 (2019)
16. Lai, V.D., Nguyen, T.N., Nguyen, T.H.: Event detection: gate diversity and syntactic importance scores for graph convolution neural networks. In: Proc. of EMNLP (2020)

17. Cui, S., Yu, B., Liu, T., Zhang, Z., Wang, X., Shi, J.: Edge-enhanced graph convolution networks for event detection with syntactic relation. In: Proc. of EMNLP Findings (2020)
18. Zhao, Y., Jin, X., Wang, Y., Cheng, X.: Document embedding enhanced event detection with hierarchical and supervised attention. In: Proc. of ACL, pp. 414–419 (2018)
19. Liu, J., Chen, Y., Liu, K., Bi, W., Liu, X.: Event extraction as machine reading comprehension. In: Proc. of EMNLP, pp. 1641–1651 (2020)
20. Du, X., Cardie, C.: Event extraction by answering (almost) natural questions. In: Proc. of EMNLP (2020)
21. Chen, Y., Chen, T., Ebner, S., White, A.S., Van Durme, B.: Reading the manual: event extraction as definition comprehension. In: Proceedings of the Fourth Workshop on Structured Prediction for NLP (2020)
22. Lafferty, J.D., McCallum, A., Pereira, F.C.N.: Conditional random fields: probabilistic models for segmenting and labeling sequence data. In: Proc. of ICML (2001)
23. Liu, X., Luo, Z., Huang, H.: Jointly multiple events extraction via attention-based graph information aggregation. In: Proc. of EMNLP (2018)