Event Extraction via Extracting Triggers and Arguments Simultaneously and Matching

Qianlong Wang and Jiangtao Ren*
School of Data and Computer Science
Sun Yat-sen University
Guangzhou, China
wangqlong3@mail2.sysu.edu.cn, issrjt@mail.sysu.edu.cn

***Abstract*—In recent years, researchers have proposed various methods to achieve better results on the event extraction task. Among them, the pipelined-based methods first perform event trigger prediction and then identify arguments in separate stages. Therefore, the errors from the trigger identification propagate the argument identification. The joint-based methods generally consider the argument role task as a multi-class classification problem, which undoubtedly reduces efficiency. To address these two shortcomings, in this paper, we propose an effective neural model for event extraction. Specifically, we use a Named Entity Recognition method to label arguments while extracting event triggers. As a result, the labels contain arguments and their role information. If the arguments are assigned to event triggers, we assume that there are inter-dependencies between them. Inspired by their inter-dependencies, we then assign arguments to event triggers by matching. This matching operation greatly improves efficiency as its essence is the binary classification. The experimental results on Chinese Emergency Corpus show that our model is more effective and competitive than baselines.**

***Index Terms*—Event Extraction, Named Entity Recognition, Neural Networks**

I. Introduction

Automatic event extraction is an important yet challenging task in information extraction field. Event extraction aims to extract structured event information (e.g., event triggers and arguments *"when and where did the event happen"*, or *"who or what is involved"*) from unstructured text [1]. According to the definition of Automatic Content Extraction[1] (ACE), event extraction consists of two sub-tasks. One is to extract event triggers (trigger identification) and classify them into predefined types (trigger classification). The other is argument identification and argument role classification. So an event description includes both trigger and arguments. For example, as shown in Table I, 炸弹袭击(bomb attack) is an event trigger, and 2月28日(February 28^{th}) is an argument whose role in this event is *Time*.

There are two main types of methods of event extraction: (i) the pipelined methods [2], [3], and (ii) the joint methods [4], [5]. For the pipelined methods, event triggers are first extracted, and then these extracted event triggers are used as inputs for extracting events and their arguments. For example, Chen et al. [2] used a dynamic multi-pooling convolutional neural network (DMCNN) to classify each word in a sentence to identify event triggers, and applied a similar DMCNN to assign arguments to event triggers and align the roles of the arguments. Due to the nature of the pipelined methods, they made the error propagation from the upstream component (trigger identification) to the downstream classifier (argument identification). This seriously affects the performance of the methods.

TABLE I
BOLD DENOTES EVENT TRIGGERS, AND RED DENOTES ARGUMENTS.

Source text	2月28日,希拉市发生一起自杀式汽车**炸弹袭击**事件造成125名平民**死亡**200人**受伤**。 On February 28^{th}, a suicide car **bomb attack** in Shira City **killed** 125 civilians and **injured** 200 people.
Category	恐怖袭击 (Terrorist attack)
Event description	(炸弹袭击, 2月28日, 希拉市) (死亡, 125名平民) (受伤, 200人) (bomb attack, February 28^{th}, Shira City) (killed, 125 civilians) (injured, 200 people)

For the joint methods, the event triggers and arguments are predicted simultaneously, and the event extraction is regarded as a structured prediction problem. For instance, Li et al. [4] proposed a structured prediction based on the joint framework, which extracts event triggers and arguments with a large set of local features and global ones. Besides, Nguyen et al. [5] proposed to do event extraction in a joint framework, which uses the dense representations to perform event trigger and argument role identification simultaneously. Compared with the pipelined methods, the joint methods are capable of mitigating the error propagation problem. However, the joint methods generally treat the argument role task as a multi-class classification problem, which undoubtedly reduces efficiency.

Named Entity Recognition (NER) is a sub-task of information extraction. NER seeks to locate and classify entities in the text into predefined categories. In event extraction task, the arguments are entities participating in events. There are inter-dependencies between them if arguments are assigned to event triggers. Therefore, we assume that: (i) NER methods can be used to label arguments while extracting event triggers, and (ii) argument role task can be downgraded to a binary classification task with the help of labels and inter-dependencies.

Based on the above assumptions, in this paper, we propose

* Corresponding author.
[1]https://catalog.ldc.upenn.edu/ldc2006t06

978-1-7281-6926-2/20/$31.00 ©2020 IEEE

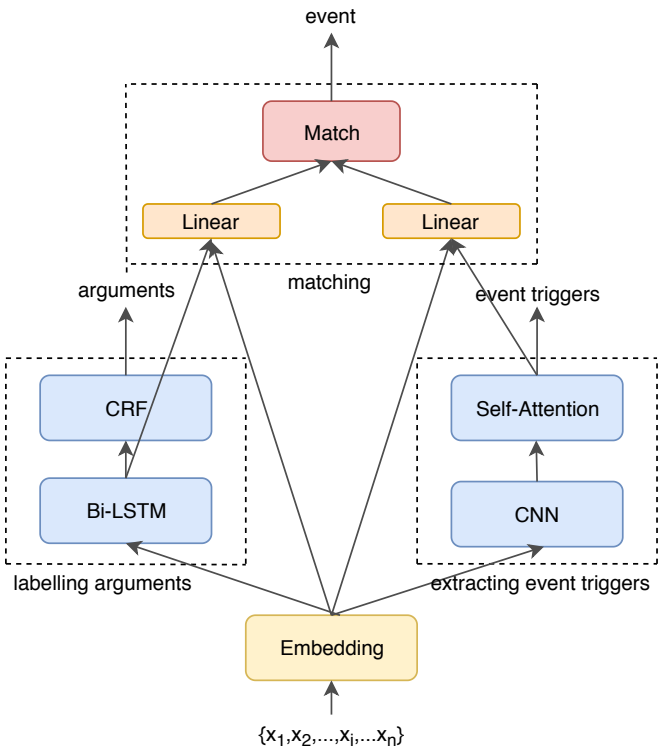

Fig. 1. Structure overview of our model.

an effective neural model[2] for event extraction. In our model, we first extract event triggers and arguments simultaneously. Here, we treat the arguments extraction as a NER task, as both arguments and their role can be labeled. We then employ matching operation to assign arguments to event triggers based on their inter-dependencies. As shown in Figure 1, our model consists of three components: 1) extracting event triggers; 2) labelling arguments; 3) matching between event triggers and arguments. Specifically, first of all, we use a convolutional neural network (CNN) [6] with self-attention mechanism to extract event triggers. Here, CNN and self-attention mechanism are capable of capturing the local and global features respectively. These features are intuitively helpful to identify event triggers. Simultaneously, we utilize a bidirectional long short-term memory (Bi-LSTM) [7] with conditional random field (CRF) [8] to label arguments. In this work, we use a relatively simple annotation mode (BIO + role), where B, I and O denote the beginning of argument, the inside of argument and the no-argument, respectively. For example, the label B-Time denotes the word is at the beginning of argument and its role is Time[3]. After extracting event triggers and labelling arguments, we then employ matching operation to assign arguments to event triggers. In other words, given an event trigger, this matching operation determines which arguments should be assigned to it. Thus it can be seen that the essence of matching operation is the binary classification. If an argument is assigned to an event trigger, it plays a specific role, obtained by its label, in the event. For example, as shown in Table I, the argument 2月28日(February 28^{th}) is assigned to event trigger 炸弹袭击(bomb attack), and its role is Time, which is obtained by label B-Time.

Compared with the pipelined methods [2], [3], our model alleviates the error propagation from the trigger identification to the argument identification as a consequence of extracting event triggers and arguments simultaneously. In addition, compared with the joint methods [4], [5], our model improves efficiency, as we downgrade the argument role task to a binary classification task.

Our contributions could be summarized as follows:

- We propose an effective neural model to solve event extraction problem, which mitigates the error propagation without relying on features.
- Our model downgrades the argument role task to a binary classification task by skillfully utilizing labels of NER, which greatly increase the efficiency of the model. To the best of our knowledge, this is an innovative work to employ a binary classification method to accomplish the argument role task.
- Experimental results on Chinese Emergency Corpus show that our model is more effective and competitive than baselines.

The rest of paper is organized as follows: Section II illustrates related work; Section III introduces the proposed model in detail; Section IV presents experimental results; Section V concludes this paper.

II. Related Work

Automatic event extraction is an important and challenging task in information extraction field.

Early work on event extraction primarily focused on using the local sentence-level representations in the pipelined methods. For instance, Grishman et al. [9] took advantage of the combination of logical grammatical structure and predicate-argument structure in event recognition. Ahn et al. [10] used the local sentence-level representations in modular approach. After that, the higher level features are investigated to improve the performance of event extraction systems. Specifically, Ji and Grishman [3] designed a simple scheme to conduct cross-document inference on event extraction based on the global features. Li et al. [4] proposed the structured perceptron algorithm with a large set of local features and global ones, such as the constituent tags. In addition, there are many methods [11]–[13] utilizing the higher level features to solve event extraction.

Recent research proposed the joint models for event extraction, including the models based on markov logic networks [14]–[16], structured perceptron [4], [17], and dual decomposition [18].

Recently, researchers applied the neural networks to event extraction task. In particular, Nguyen and Grishman [19] studied event detection problem using CNN, which overcomes

[2]Source code is available athttps://github.com/qlwang25/event_extraction

[3]Chinese Emergency Corpus specifies that an argument only play a specific role in an event.

the fundamental limitations of the feature-based approaches. Chen et al. [2] applied two DMCNNs on event extraction in a pipelined method.

Named Entity Recognition (NER) is an important sub-task of information extraction. Researchers used different statistical models to perform NER task, such as hidden markov model [20] and conditional random fields [21]. Recently, several neural network models [22], [23] are successfully applied to NER, in which NER is regarded as a sequence labeling problem.

III. Model

Section A provides the problem definition; Section B gives the overview of our model; Two components (extract event triggers and label arguments) are introduced from Section C to Section D respectively; Section E explains the matching operation between event triggers and arguments; Section F presents the overall loss.

A. Problem Definition

ACE defines an event as something that happens or leads to some change of state. In the following section, we introduce some terminologies to help better understand event extraction task.

- **Event trigger**: a word or phrase that clearly expresses an event occurrence.
- **Event argument**: an entity mention, temporal expression or value that serves as a participant or attribute in an event.
- **Argument role**: the relationship between arguments to the event in which it participates.

Following the previous work [24], an event is formally defined as a four-tuple:

$$event = (A, O, T, V)$$

where A is an event trigger; O denotes objects participating in events; T and V respectively denote time and location environment. Due to the differences between Chinese Emergency Corpus and ACE 2005 corpus, in this work, we do not consider the trigger classification task. Thus, in this work, event extraction includes event trigger A identification, and argument O/T/V identification and argument role classification.

B. Overview

An overview of our model is shown in Figure 1. Our model takes a word sequence $x = \{x_1, x_2, ..., x_i, ..., x_n\}$ as input. Embedding first maps each word x_i to a real number vector e_i. Our model then uses CNN and self-attention to extract event triggers, and employs Bi-LSTM and CRF to label arguments, simultaneously. Finally, our model assigns arguments to event triggers by matching, and the role of arguments is obtained by corresponding labels.

C. Extract Event Triggers

In this section, we introduce in detail how to extract event triggers (i.e., the right-dashed box in Figure 1).

a) CNN: Since the local-range semantic information is useful for extracting event triggers [2], in this work, we use CNN to capture the local-range semantic information and to overcome the limitations of feature-based methods. Besides, we assume that the local-range semantic information fixes Chinese lexical ambiguity errors.

We use CNN to encode the embedding sequence. Given the input embedding sequence $e = \{e_1, e_2, ..., e_i, ..., e_n\}$, the formula of convolutional operation is written as:

$$c_i^k = \text{Conv}(e_{i-\lfloor \frac{k \cdot d}{2} \rfloor}, ..., e_i, ..., e_{i+\lfloor \frac{k \cdot d}{2} \rfloor}) \tag{1}$$

where k and d denote the kernel size and the dilation coefficient, respectively. In this work, we employ the multiple convolutional filters with different kernel size to produce the local-range semantic information. The multiple convolutional filters are capable of capturing the local-range semantic information of various granularities. Following the previous work [25], we use two convolutional filters with widths of 2 and 3 to encode the semantic information of bigrams and trigrams. Hence the output of CNN is $c = \{c_1, c_2, ..., c_i, ..., c_n\}$, and $c_i = [c_i^2 \oplus c_i^3]$, where $\oplus$ denotes the concatenation.

b) Self-Attention: It is inadequate to identify event triggers only using the local-range semantic information. Since self-attention [26] mechanism offers the ability to capture the long-range dependencies information, in this work, we use it to produce the richer semantic information.

Given $c = \{c_1, c_2, ..., c_i, ..., c_n\}$, we use the following formulas to further produce the richer semantic information:

$$\alpha_i = \text{softmax}(W_{c1} \cdot \tanh(W_{c2} \cdot c_i)) \tag{2}$$

$$o_{trigger,i} = \sum_{j=1}^{n} \alpha_{i,j} \cdot c_j \tag{3}$$

where W_{c1} and W_{c2} are the trainable weights. The outputs are collected and treated as $o_{trigger}$.

c) Softmax: A Chinese event trigger may include one or more words. For example, 恐怖袭击(terrorist attack) is an event trigger, which is a phrase consisting of 恐怖(terror) and 袭击(attack). So, it is difficult to accurately extract event triggers in Chinese corpus. Inspired by the work [25], we label each word in the input text with tag 0 or 1 to extract event triggers. The tag 1 denotes the word is an event trigger, and the tag 0 is the opposite. Therefore, we transform the extraction of event triggers into a 0 or 1 tag problem.

Given the input text x, we predict a tag sequence t^p:

$$p_{\theta_1}(t|x) = \text{softmax}(W_t \cdot o_{trigger} + b_t) \tag{4}$$

$$t^p = \text{argmax}_t(p_{\theta_1}(t|x)) \tag{5}$$

where W_t is a trainable weight, and b_t is a trainable bias, $p_{\theta_1}(t|x)$ is the probability distribution of tags, and θ_1 denotes all trainable parameters of event triggers module.

We use the negative log-likelihood function to calculate the loss $l_{trigger}$:

$$l_{trigger} = -\sum_{i=1}^{n} \log(p_{\theta_1}(t_i^t|x)) \tag{6}$$

where t^t is the ground truth tag sequence of the input text x, and n is the length of x.

D. Label Arguments

In this section, we introduce in detail how to label arguments (the left-dashed box in Figure 1).

a) Bi-LSTM: In this work, we feed the embedding sequence e to forward direction LSTM, which considers the contextual information in the positive direction; in the similar way, we feed the reverse order e to backward direction LSTM, which considers the contextual information in the negative direction. Therefore, we can use the information from the past and future of current time by processing the embedding sequence in two directions.

Given the input text x, the input of Bi-LSTM is e_t, and its output is h_t at time step t:

$$\overrightarrow{h}_t = \text{LSTM}(e_t, \overrightarrow{h}_{t-1}) \tag{7}$$

$$\overleftarrow{h}_t = \text{LSTM}(e_t, \overleftarrow{h}_{t+1}) \tag{8}$$

$$h_t = [\overrightarrow{h}_t \oplus \overleftarrow{h}_t] \tag{9}$$

After this operation, we obtain the hidden states sequence $h = \{h_1, h_2, ..., h_i, ..., h_n\}$.

b) CRF: Bi-LSTM does not capture the dependency information between neighboring labels, thus it fails to produce a optimized label sequence. In this work, we use CRF [8] to overcome this shortcoming.

The conditional probability of the input text x with a label sequence $y = \{y_1, y_2, ..., y_i, ..., y_n\}$ is:

$$\text{Score}(x, y) = \sum_{i}^{n} h_{i,[y_i]} + A_{y_i, y_{i-1}} \tag{10}$$

$$p_{\theta_2}(y|x) = \frac{\exp(\text{Score}(x, y))}{\sum_{y'} \exp(\text{Score}(x, y'))} \tag{11}$$

where h and A are the emission matrix and transition matrix, respectively; h is the hidden states, and A is a trainable weight; $A_{y_i, y_{i-1}}$ is a score of transitioning to label y_i from label y_{i-1}; $y^{'}$ denotes all possible label sequences, and θ_2 is all trainable parameters of this module. During testing, given the input text x, we use the viterbi algorithm [27] to find the label sequence with maximum score.

To train parameters θ_2, we employ the following equation to calculate the loss $l_{argument}$:

$$l_{argument} = \log \sum_{y'} \exp(\text{Score}(x, y^{'})) - \text{Score}(x, y^t) \tag{12}$$

where y^t is the ground truth label sequence of the input text x.

E. Match

After extracting event triggers and labelling arguments, we use a simple matching operation to assign arguments to event triggers. As for the role of arguments, it is directly obtained by their label. In this paper, we downgrade the argument role task to a binary classification task[4], which undoubtedly improves efficiency.

Firstly, we use two different linear transformations after the residual connection:

$$r_{trigger} = \text{Linear}(e \oplus o_{trigger}) \tag{13}$$

$$r_{argument} = \text{Linear}(e \oplus h) \tag{14}$$

where the linear transformation is to transfer two inputs to the same semantic space, and its computational formula is $y = W \cdot x + b$ (W is a trainable weight, b is a trainable bias, x is the input and the output is y.)

In this work, we assume that the arguments are assigned to the event triggers, if there are inter-dependencies between them. Based on this assumption, we then employ the dot results between them to represent the strength of their inter-dependencies:

$$u = \text{sigmoid}(r_{trigger} \cdot r_{argument}^T) \tag{15}$$

Here, we set a *threshold* value. If values in u are greater than *threshold*, it means that there are inter-dependencies and the arguments should be assigned to corresponding event triggers. For example, if $u_{i,j} >$ *threshold*, the argument w_j should be assigned to the event trigger w_i, and the premise is that w_i and w_j are an event trigger and an argument, respectively.

Finally, we measure the binary cross entropy between targets and predictions. We calculate the loss l_{match}, which is the training object of the matching module:

$$l_{match} = -\sum_{i}^{n}\sum_{j}^{n}[g_{i,j} \cdot \log u_{i,j} + (1 - g_{i,j}) \cdot \log(1 - u_{i,j})] \tag{16}$$

where $g_{i,j}$ is the ground truth matching result, and its values are in the set $\{0, 1\}$.

F. Train

As described above, we have three losses: $l_{trigger}$, $l_{argument}$ and l_{match}. We train our networks by minimizing the loss l with stochastic gradient descent [28] algorithm:

$$l = l_{match} + \lambda_1 \cdot l_{trigger} + \lambda_2 \cdot l_{argument} \tag{17}$$

where λ_1 and λ_2 are two hyper-parameters to balance three losses.

IV. Experiments

In this Section, Chinese Emergency Corpus and experimental results are mainly introduced in detail. Firstly, we provide the details of Chinese Emergency Corpus and experiment settings; Secondly, we introduce several popular state-of-the-art models; Finally, we report the comparison results and present a brief analysis.

[4]Note that, i) the previous work used the multi-class classification based approach to solve the argument role task, and ii) we just downgrade this task to a binary classification task, instead of the whole event extraction task.

TABLE II
THE STATISTICS OF CHINESE EMERGENCY CORPUS.

Training set	346
Validation set	43
Test set	43
Average number of sentences in news	8.42
Average length of event trigger	3.04
Average length of argument	4.63
Number of argument role	5

TABLE III
THE NUMBER OF EVENT DATA STRUCTURES IN CHINESE EMERGENCY CORPUS.

Data set	Event	Denoter	Time	Location	Participant	Object
Training	5597	5597	1314	1427	2663	1980
Validation	809	809	177	202	381	279
Test	770	770	168	227	358	295

A. Dataset

In this work, we use Chinese Emergency Corpus[5] (CEC) which is built by Shanghai University. In this corpus, news on six categories of emergencies (earthquake, fire, traffic accident, terrorist attack, intoxication of food and pollution) are collected. Note that the original CEC only contains the first five categories, and news on pollution category[6] come from Chinese Environment Emergency Corpus[7] (CEEC). To increase the scale of corpus, we merge CEEC into CEC. Therefore, the total number of news is 432.

CEC includes six data structures: *Event, Denoter, Time, Location, Participant* and *Object*. *Event* and *Denoter* respectively denote the event and the event trigger. *Time, Location, Participant* and *Object* represent the argument. The main difference between ACE 2005 corpus and CEC lies in the definition of the arguments. The ACE 2005 corpus defines the different arguments for different type events. For example, Born event has three arguments: *Person*, *Time* and *Place*, and Phone-Write event has two arguments: *Entity* and *Time*. By contrast, CEC defines no more than four arguments (*Time*, *Location*, *Participant* and *Object*) for all events. In addition, ACE 2005 corpus defines one type for each event trigger while CEC defines one category for each news text. Thus, in this work, we do not conduct the trigger classification task.

In CEC, each news includes several paragraphs, each paragraph consists of several sentences, each sentence contains one or more events, and each event consists of one event trigger (*Denoter*) and multiple arguments. For example, the sentence in Table I contains three events whose event triggers are 炸弹袭击(bomb attack), 死亡(killed) and 受伤(injured), respectively. According to the ratio of $8:1:1$, we split CEC into the training set, the validation set and the test set. Table II shows the detailed statistics of CEC, and Table III gives the statistics of event data structures in each data set. From Table III, we can observe the number of *Event* and *Argument role* included in each data set. For example, the test set includes 770 *Event* and 168 *Time* argument roles. It is clear that an event must include an event trigger (i.e., *Event* == *Denoter* in Table III).

B. Experimental Details

We initialize all trainable parameters with values drawn from $\mathcal{N}(0,1)$, and tune the following hyper-parameters based on the performance of the validation set. We set *threshold* to 0.6; Both λ_1 and λ_2 are set to 1; In addition, we set the hidden size of Bi-LSTM and CNN to 256; Embedding size is 300; The learning rate is set to 0.001; Batch size is 32. During training, we use the dropout [29] to avoid the over-fitting; We apply rectified linear unit [30] to enable better training our networks; To reduce the difficulty of training, we first train the event triggers module and the arguments module separately, then load their trained parameters, finally perform the jointly training; In particular, we train each module with its respective loss function ((6) for the event triggers module; (12) for the arguments module), before using loss function (17); We use Precision (P), Recall (R) and F1 measure as the evaluation metrics.

C. Baselines

We compare our model with the following baselines:

- **HNN** [25] is a hybrid neural network, which captures both sequence and chunk information for event detection task.
- **DMCNN** [2] automatically extracts the lexical-level and the sentence-level features, and formulates event extraction as two-stages: (i) the event trigger classification; (ii) the argument classification.
- **JRNN** [5] is a joint framework, and consists of two steps: (i) the encoding; (ii) the prediction of event triggers and arguments role.

D. Results

Event extraction task includes trigger identification, trigger classification, argument identification and role classification. As described in Section IV-A, CEC does not define the types for event triggers. So, in this work, we compare our model with baselines on three tasks (i.e., trigger identification, argument identification and argument role). Table IV shows the overall experimental results on CEC. Note that, due to differences between ACE 2005 corpus and CEC, the event-type features are not used in DMCNN, and the entity type embeddings and depdendecy tree relations are not employed in JRNN.

We first compare our model and HNN on trigger identification. Both models label each word in the input text with tag 0 or 1 to extract event triggers. Surprisingly, our model is more competitive than HNN where the corresponding improvement is 13.78%. This dvances may benefit from the design of event triggers module and the joint training.

Then, compared with the pipelined baseline (DMCNN), Table IV shows that our model achieves significant higher

[5]https://github.com/shijiebei2009/CEC-Corpus
[6]The pollution category is divided into six sub-categories.
[7]https://github.com/shijiebei2009/CEEC-Corpus

TABLE IV
OVERALL PERFORMANCE ON CHINESE EMERGENCY CORPUS. THE MODELS WITH MARKER"*" ARE RE-IMPLEMENTED BASED ON THEIR RESPECTIVE DESCRIPTIONS, AND THE MARKER "-" INDICATES THAT RESULTS ARE NOT AVAILABLE.

	Trigger Identification			Argument Identification			Argument Role		
Metric	P	R	F	P	R	F	P	R	F
HNN*	77.17	56.81	65.44	-	-	-	-	-	-
DMCNN*	74.12	77.18	75.61	65.23	55.88	60.19	58.86	46.87	52.18
JRNN*	80.14	75.63	77.82	69.57	54.52	61.13	61.89	48.83	54.58
Our Model	80.57	77.93	**79.22**	70.19	58.98	**62.33**	61.01	56.05	**58.45**

scores. In particular, our model is 3.61% better than DMCNN on trigger identification while the corresponding improvement on argument identification is 2.14%. Here, DMCNN extracts arguments based on the extracted event triggers. So the errors from the trigger identification propagate to the argument identification. However, our model extracts event triggers and arguments simultaneously, which mitigates the error propagation. This comparison displays that alleviating the error propagation can improve the performance of the model. Besides, the comparison between DMCNN (the pipelined method) and JRNN (the joint method) also confirms our viewpoint.

Finally, as shown in Table IV, our model outperforms the joint baseline (JRNN) by a large margin on role classification. Specifically, compared with JRNN, our model achieves a gain of 1.4% on trigger identification, 1.2% on argument identification and 3.87% on role classification. Here, our model and JRNN are the joint methods. For the role classification, JRNN considers it as a multi-class classification problem. However, our model use a simple matching operation to assign arguments to event triggers, where their role is obtained by NER label. The essence of matching operation is the binary classification. This comparison on role classification shows that our model greatly increases efficiency. This phenomenon coincides with the fact that the binary classification method is more efficient than the multi-class classification method.

In addition, our model has a distinct advantage that features are not required. In work [2], [5], these baselines inevitably use features. Since the scale of CEC is small and the extraction of Chinese encounters mismatching problem, all results in Table IV are relatively low.

V. CONCLUSIONS

In this paper, we propose an effective neural model for event extraction. Specifically, we use a Named Entity Recognition method to label arguments while extracting event triggers. We then assign arguments to event triggers by matching whose essence is the binary classification. Compared with the pipelined methods, our model is capable of mitigating the error propagation. Moreover, compared with the joint methods, our model greatly improves efficiency since we downgrade the argument role task to a binary classification task. The experimental results on CEC show that our model is more effective and competitive than baselines.

ACKNOWLEDGMENT

We would like to thank Xiaoqing Tian (Sun Yat-sen University) for her careful inspection and revision on an earlier version of the manuscript, and anonymous reviewers for their valuable comments and suggestions. This research is partially supported by National Natural Science Foundation of China (No. U1811462 and U1711263).

REFERENCES

[1] T. Zhang and H. Ji, "Event extraction with generative adversarial imitation learning," *arXiv preprint arXiv:1804.07881*, 2018.

[2] Y. Chen, L. Xu, K. Liu, D. Zeng, and J. Zhao, "Event extraction via dynamic multi-pooling convolutional neural networks," in *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics*, 2015, pp. 167–176.

[3] H. Ji and R. Grishman, "Refining event extraction through cross-document inference," in *Proceedings of ACL*, 2008, pp. 254–262.

[4] Q. Li, H. Ji, and L. Huang, "Joint event extraction via structured prediction with global features," in *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, 2013, pp. 73–82.

[5] T. H. Nguyen, K. Cho, and R. Grishman, "Joint event extraction via recurrent neural networks." in *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics*, 2016, pp. 300–309.

[6] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[7] M. Schuster and K. K. Paliwal, "Bidirectional recurrent neural networks," *IEEE Transactions on Signal Processing*, vol. 45, no. 11, pp. 2673–2681, 1997.

[8] J. Lafferty, A. McCallum, F. Pereira, and K. Duh, "Probabilistic models for segmenting and labeling sequence data," in *International Conference on Machine Learning*, 2002.

[9] R. Grishman, D. Westbrook, and A. Meyers, "Nyu's english ace 2005 system description," *ACE*, vol. 5, 2005.

[10] D. Ahn, "The stages of event extraction," in *Proceedings of the Workshop on Annotating and Reasoning about Time and Events*, 2006, pp. 1–8.

[11] P. Gupta and H. Ji, "Predicting unknown time arguments based on cross-event propagation," in *Proceedings of the ACL-IJCNLP 2009 Conference Short Papers*, 2009, pp. 369–372.

[12] R. Huang and E. Riloff, "Modeling textual cohesion for event extraction," in *Twenty-Sixth AAAI Conference on Artificial Intelligence*, 2012.

[13] S. Liao and R. Grishman, "Using document level cross-event inference to improve event extraction," in *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, 2010, pp. 789–797.

[14] H. Poon and L. Vanderwende, "Joint inference for knowledge extraction from biomedical literature," in *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, 2010, pp. 813–821.

[15] S. Riedel, H.-W. Chun, T. Takagi, and J. Tsujii, "A markov logic approach to bio-molecular event extraction," in *Proceedings of the Workshop on Current Trends in Biomedical Natural Language Processing: Shared Task*, 2009, pp. 41–49.

[16] D. Venugopal, C. Chen, V. Gogate, and V. Ng, "Relieving the computational bottleneck: Joint inference for event extraction with high-dimensional features," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, 2014, pp. 831–843.

[17] Q. Li, H. Ji, H. Yu, and S. Li, "Constructing information networks using one single model," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, 2014, pp. 1846–1851.

[18] S. Riedel and A. McCallum, "Fast and robust joint models for biomedical event extraction," in *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 2011, pp. 1–12.

[19] T. H. Nguyen and R. Grishman, "Event detection and domain adaptation with convolutional neural networks," in *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics*, vol. 2, 2015, pp. 365–371.

[20] S. Morwal, N. Jahan, and D. Chopra, "Named entity recognition using hidden markov model (hmm)," *International Journal on Natural Language Computing*, vol. 1, no. 4, pp. 15–23, 2012.

[21] G. Luo, X. Huang, C.-Y. Lin, and Z. Nie, "Joint entity recognition and disambiguation," in *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, 2015, pp. 879–888.

[22] J. P. C. Chiu and E. Nichols, "Named entity recognition with bidirectional lstm-cnns," *Transactions of the Association for Computational Linguistics*, vol. 4, pp. 357–370, 2016.

[23] Z. Huang, W. Xu, and K. Yu, "Bidirectional lstm-crf models for sequence tagging." *arXiv preprint arXiv:1508.01991*, 2015.

[24] T. Liao, Z. Liu, and X. Wang, "Research and implementation on event-based method for automatic summarization," in *Proceedings of The Eighth International Conference on Bio-Inspired Computing: Theories and Applications*. Springer, 2013, pp. 103–111.

[25] X. Feng, L. Huang, D. Tang, H. Ji, B. Qin, and T. Liu, "A language-independent neural network for event detection." in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, 2016, pp. 66–71.

[26] Z. Lin, M. Feng, C. N. dos Santos, M. Yu, B. Xiang, B. Zhou, and Y. Bengio, "A structured self-attentive sentence embedding," *International Conference on Learning Representations*, 2017.

[27] L. R. Rabiner, "A tutorial on hidden markov models and selected applications in speech recognition," *Proceedings of the IEEE*, vol. 77, no. 2, pp. 257–286, 1989.

[28] J. Kiefer, J. Wolfowitz *et al.*, "Stochastic estimation of the maximum of a regression function," *The Annals of Mathematical Statistics*, vol. 23, no. 3, pp. 462–466, 1952.

[29] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov, "Improving neural networks by preventing co-adaptation of feature detectors," *arXiv preprint arXiv:1207.0580*, 2012.

[30] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *Proceedings of the 27th International Conference on Machine Learning*, 2010, pp. 807–814.