A Joint Learning Framework for the CCKS-2020 Financial Event Extraction Task

Jiawei Sheng[1,2], Qian Li[3], Yiming Hei[3], Shu Guo[4] ⋆, Bowen Yu[1,2], Lihong Wang[4] ⋆⋆, Min He[4], Tingwen Liu[1,2], and Hongbo Xu[1,2]

[1] Institute of Information Engineering, Chinese Academy of Sciences
[2] School of Cyber Security, University of Chinese Academy of Sciences
[3] Beihang University
[4] National Computer Network Emergency Response Technical Team / Coordination Center of China
[5] shengjiawei@iie.ac.cn, guoshu@cert.org.cn, wlh@isc.org.cn

Abstract. This paper presents a winning solution for the CCKS-2020 financial event extraction task, where the goal is to detect event types, triggers and arguments in sentences across multiple event types. In this task, we focus on resolving two challenging problems (i.e. low resources and element overlapping) by proposing a joint learning framework, named `SaltyFishes`. We first formulate the event extraction task as a joint probability model. By sharing parameters in the model across different types, we can learn to adapt to low-resource events based on high-resource events. Then we further address the element overlapping problems by a mechanism of Conditional Layer Normalization, achieving even better extraction accuracy. The overall approach achieves a F1-score of 87.8% which ranks the first place in the task.

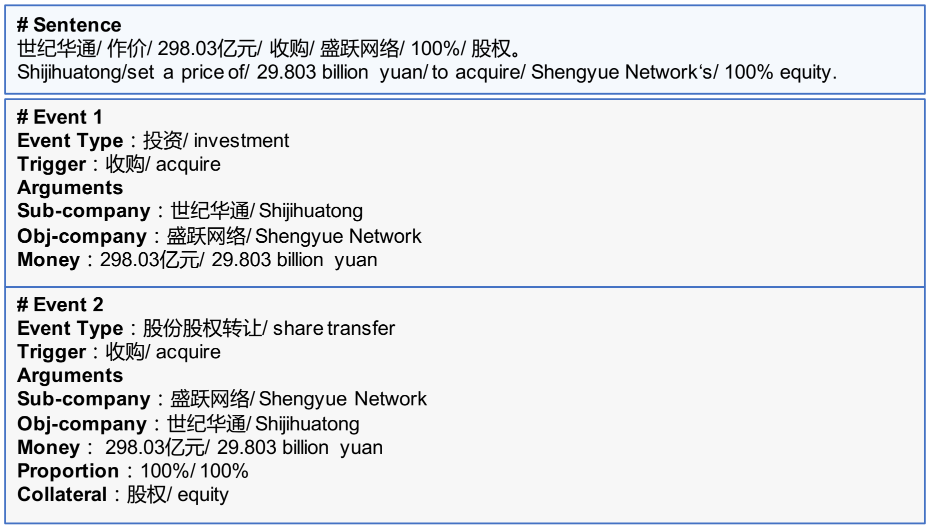
Keywords: Event Detection · Event Extraction · Joint Learning.

1 Introduction

The CCKS-2020 financial event extraction task[1] aims at extracting structural events by detecting event types, triggers and arguments in sentences across multiple types. Figure (1) gives an example of event extraction for a financial news sentence. One structural event belongs to the type of 投资, along with the trigger 收购 and its arguments providing more complementary details. Note that, this sentence contains more than one event, and the trigger and arguments overlap across the events.

The CCKS-2020 task provides two kinds of such event sentences. The first one contains 5 types of events associated with abundant sentence corpus, called source events. The second one contains another 5 types of events associated with low-resource sentence corpus, called target events. Each type of event sentences

⋆ Corresponding author.
⋆⋆ Corresponding author.
[1] `https://www.biendata.xyz/competition/ccks_2020_3/`



# Sentence
世纪华通/ 作价/ 298.03亿元/ 收购/ 盛跃网络/ 100%/ 股权。
Shijihuatong/set a price of/ 29.803 billion yuan/ to acquire/ Shengyue Network‘s/ 100% equity.

# Event 1
Event Type：投资/ investment
Trigger：收购/ acquire
Arguments
Sub-company：世纪华通/ Shijihuatong
Obj-company：盛跃网络/ Shengyue Network
Money：298.03亿元/ 29.803 billion yuan

# Event 2
Event Type：股份股权转让/ share transfer
Trigger：收购/ acquire
Arguments
Sub-company：盛跃网络/ Shengyue Network
Obj-company：世纪华通/ Shijihuatong
Money： 298.03亿元/ 29.803 billion yuan
Proportion：100%/ 100%
Collateral：股权/ equity

Fig. 1. Example of event extraction with element overlaping problem.

is split into training (labeled data) and testing (unlabeled data) parts. Our goal is to test the performance of event extraction on the test set of target events. This poses two main challenges compared to traditional event extraction tasks [2–4, 6, 7]:

– The target events contain only 179 training sentences on average for each type. This limited supervision information cannot provide sufficient contextual information for the event extraction.
– Elements can be overlapping with each other, i.e., the same trigger or argument may belong to different events. As shown in the example, the trigger 收购 and the argument 世纪华通 belong to both event types of 投资 and 股份股权转让. Performing event extraction by a simple sequence labeling method will cause label conflicts.

To address these challenges, we devise a joint learning method. In our approach, the overall framework is formulated as a joint probability model, which is decomposed into submodels, i.e., the joint distribution is decomposed into a product of three conditional distributions. Each subtask will be a specifical use of this distribution, including event type detection, trigger extraction and argument extraction. For the first subtask, given a financial news sentence, we first classify the sentence into a correct event type by using a multi-class multi-label text classification paradigm. For the other two subtasks, we successively extract triggers and arguments with a pre-training/fine-tuning framework. The pre-training module is implemented by a pre-trained language model RoBERTa [5] on all financial news sentences, and we further fine-tune the pre-trained model with respect to the trigger/argument extraction module. To deal with the element

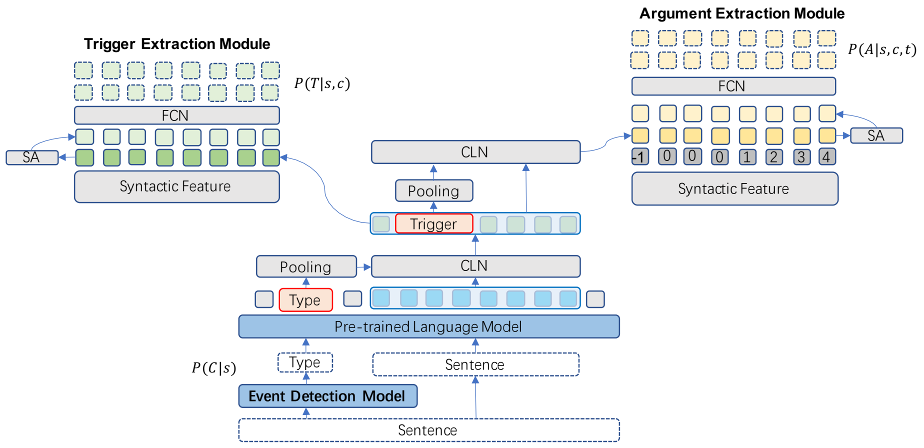


Fig. 2. The overall framework of the financial event extraction approach.

overlaping issue, we introduce Conditional Layer Normalization, only extracting triggers according to the specific event type, and extracting arguments according to the specific trigger. This way can extract elements separately in different conditions, avoiding overlapping. In addition, by sharing parameters across different types in such a unified model, we can learn to adapt to low-resource events based on high-resource events. Our approach achieves a F1-score of 87.8% which ranks the first in the CCKS-2020 financial event extraction competition.

2 Our Approach

This section introduces our approach in details. We will present the overview, the design of each component, and some strategies for improvements.

2.1 Overview

Given a sentence denoted as s, we propose a joint learning approach to detect its event types C, event triggers T and event arguments A. The approach is formulated as a joint probability model, which is decomposed into three submodels with respect to the event type detection, the event trigger extraction and the event argument extraction:

$$P(C,T,A|s;\Theta) \propto P(C|s;\Theta_1)P(T|s,C;\Theta_2,\Theta_3)P(A|s,C,T;\Theta_2,\Theta_4). \quad (1)$$

The event type detection is modeled by a multi-class multi-label text classification paradigm, where Θ_1 is the set of model parameters. Other two extraction parts are modeled by a pre-training/fine-tuning framework, where Θ_2 contains model parameters shared by both modules, while Θ_3 and Θ_4 are respective private model parameters. All parameters in $\Theta = \{\Theta_1, \Theta_2, \Theta_3, \Theta_4\}$ are used across different event types (either high-resource or low-resource), which promotes rich interactions between source events and target events. Figure (2) sketches the overall framework.

2.2 Event Type Detection Model

In order to discover the event types occuring in the sentence, we adopt codes provided by official competition[6] as our Event type Detection Model (EDM). This model utilizes a pre-trained language model [7] to derive sentence representations, formulated as a multi-label multiclass text classification. Specifically, given the sentence s, the probability of s belonging to a specific type c is calculated as follows:

$$p(c|s;\Theta_1) = sigmoid(\mathbf{w}_c \cdot \boldsymbol{z}_{sent}), \tag{2}$$

where $\boldsymbol{z}_{sent}$ is the hidden state corresponding to the input token `<CLS>` in PLM, which encodes the entire sentence representation of s; Θ_1 includes all parameters used in PLM.

Then, we can update and obtain the desired sentence representation $\boldsymbol{z}_{sent}$ by minimizing following binary cross entropy loss function:

$$Loss(\Theta_1) = \sum_{m=1}^{M} y_m * log(p_m) + (1 - y_m) * log(1 - p_m) \tag{3}$$

where M is the number of the training sentences; y_m is the true type label. During prediction, we simply set a threshold δ and select the resultant event types C where each type c such that $p(c|s) > \delta$.

2.3 Extraction Model

This section introduces our Event Extraction Model (EEM), achieving two subtasks by a pre-training/fine-tuning framework: trigger extraction and argument extraction. The pre-training part encodes sentence tokens as contextualized representations with the pre-trained language model (PLM) RoBERTa [5], which contains rich language knowledge widely used for NLP tasks. The fine-tuning part is divided into three modules, including a shared module to encode condition information based on Conditional Layer Normalization, and two private modules to extract triggers and arguments. Note that both extraction modules have similar model structure.

Shared Module This section introduces a sentence representation layer shared by both trigger extraction and argument extraction, which will derive a conditional sentence representation H_{s-typ} for the specific event type c, and a syntactic feature representation H_{syn}.

[6] `https://github.com/xyionwu/ccks-2020-finance-transfer-ee-baseline`
[7] `https://github.com/ymcui/Chinese-BERT-wwm`

Since we have obtained event types C occuring in the given sentence s, we are going to derive sentence representations conditioned on each specific event type $c \in C$, so as to avoid element overlapping issues. To this end, we introduce a general module, named Conditional Layer Normalization (CLN) [8], to integrate such conditional information into sentence representation. CLN is mostly based on the well-known layer normalization [1], but can dynamically generate gain $\boldsymbol{\gamma}$ and bias $\boldsymbol{\beta}$ based on the condition information. Given a condition representation $\boldsymbol{c}$ and a sentence representation $\boldsymbol{x}$, CLN is formulated as:

$$CLN(\boldsymbol{x}, \boldsymbol{c}) = \boldsymbol{\gamma}_c \odot (\frac{\boldsymbol{x} - \mu}{\sigma}) + \boldsymbol{\beta}_c, \tag{4}$$

$$\mu = \frac{1}{d}\sum_{i=1}^{d} \boldsymbol{x}_i, \sigma = \sqrt{\frac{1}{d}\sum_{i=1}^{d}(\boldsymbol{x}_i - \mu)^2}, \tag{5}$$

$$\boldsymbol{\gamma}_c = W_\gamma \boldsymbol{c} + \boldsymbol{b}_\gamma, \boldsymbol{\beta}_c = W_\beta \boldsymbol{c} + \boldsymbol{b}_\beta, \tag{6}$$

where $\boldsymbol{x}_i$ is the i-th token representation in $\boldsymbol{x}$, $\boldsymbol{\gamma}_c \in \mathbb{R}^d$ and $\boldsymbol{\beta}_c \in \mathbb{R}^d$ are the conditional gain and bias, respectively. In this way, the given condition representation is encoded into the gain and bias, and then integrated into the contextual representations.

Then we utilize CLN to integrate event type information into the sentence. Specifically, we first transform the event type's name into textual tokens, such as the type 投资 is transformed into tokens 投 and 资. Then we concatenate these type tokens together with the word tokens in the sentence s, forming a sequence as X : `<CLS>`+ `type tokens` + `<SEP>` + `word tokens` + `<SEP>`. The sequence is input into the PLM to derive their contextualized representations, and we term the representations corresponding to `type tokens` as H_c and `word tokens` as H_s. Then, we fuse H_c with mean pooling and H_s together, to derive the conditional token representations for s:

$$H_{s-typ} = \mathrm{CLN}\left(H_s, MeanPooling(H_c)\right), \tag{7}$$

where H_{s-typ} is the token representations conditioned on the event type c. Such process actually generates type-aware token representations adaptive to various event types. As such, we can perform trigger extraction and argument extraction in the independent context of each type.

Private Module The private module contains the following two sub-modules.

(1) **Trigger Extraction Module (TEM)** This module extracts event triggers given the event type $c \in C$. In order to improve textual representations for trigger extraction, we adopt a Self-Attention (SA) layer. Thus the type-aware token representations can be enhanced as follows:

$$H_{sa-typ} = SA(H_{s-typ}). \tag{8}$$

$$H_{tri} = H_{s-typ} \oplus H_{sa-typ} \oplus H_{syn}. \tag{9}$$

where $\oplus$ is the concatenation operation. H_{syn} corresponds to the representation of syntactic features, obtained by NLP tool LTP[8].

In order to strength interactions among triggers of different event types, we predict triggers with the same trigger extractor. For each token, we predict whether it is a begin or end position of a trigger as:

$$p(t^{(b)}|x_i, c; \Theta_2, \Theta_3) = sigmoid(\boldsymbol{w}_{t^{(b)}} * \boldsymbol{h}_{tri,i}), \tag{10}$$

$$p(t^{(e)}|x_i, c; \Theta_2, \Theta_3) = sigmoid(\boldsymbol{w}_{t^{(e)}} * \boldsymbol{h}_{tri,i}) \tag{11}$$

where $\boldsymbol{h}_{tri,i}$ is the i-th element of H_{tri}. $\boldsymbol{w}_{t^{(b)}}$ and $\boldsymbol{w}_{t^{(e)}}$ are learnable parameters. Θ_3 includes $\boldsymbol{w}_{t^{(b)}}$, $\boldsymbol{w}_{t^{(e)}}$, and parameters in SA.

Then, a binary cross entropy loss function is used for begin position prediction and end position prediction, denoted as $Loss_{t^{(b)}}$ and $Loss_{t^{(e)}}$. The final loss is defined as:

$$Loss_{tri}(\Theta_2, \Theta_3) = w_t * Loss_{t^{(b)}}(\Theta_2, \Theta_3) + (1 - w_t) * Loss_{t^{(e)}}(\Theta_2, \Theta_3), \tag{12}$$

where $w_t \in (0, 1)$ is a trade-off factor. For prediction, we simply set a threshold ϕ_{tri}, and select positions such that their prediction scores are higher than ϕ_{tri}. We just match the start position with the nearest end position to obtain a complete trigger. The final trigger extraction results form the trigger set T.

(2) **Argument Extraction Module (AEM)** This module is to extract arguments conditioned on one of the triggers T extracted from the TEM. Given a specific trigger $t \in T$ in the sentence s, we obtain trigger-aware sentence representation H_{s-tri} conditioned on t, where the process is the same as Eq (7). We also utilize self-attention layer to enhance the sentence representation, termed as H_{sa-tri}. To discern the position of trigger t, we further add its relative position embedding R, which measures the distance from current position to the trigger position. The syntactic feature H_{syn} is also taken into consideration. Thus, the enhanced sentence overall representation is:

$$H_{arg} = H_{s-tri} \oplus H_{sa-tri} \oplus R \oplus H_{syn}, \tag{13}$$

As for trigger extraction, we also extract all arguments with the same extractor and devise it as follows:

$$p(a_k^{(b)}|x_i, c, t; \Theta_2, \Theta_4) = sigmoid(\boldsymbol{w}_{a_k^{(b)}} * \boldsymbol{h}_{arg,i}), \tag{14}$$

$$p(a_k^{(e)}|x_i, c, t; \Theta_2, \Theta_4) = sigmoid(\boldsymbol{w}_{a_k^{(e)}} * \boldsymbol{h}_{arg,i}), \tag{15}$$

where $\boldsymbol{h}_{arg,i}$ is the i-th element of H_{arg}. $\boldsymbol{w}_{a_k^{(b)}}$ and $\boldsymbol{w}_{a_k^{(e)}}$ are learnable parameters for the k-th argument role. Θ_4 includes $\boldsymbol{w}_{a_k^{(b)}}$, $\boldsymbol{w}_{a_k^{(e)}}$, and the parameters in SA and CLN.

[8] `http://ltp.ai/`

The loss function is also binary cross entropy for both begin and end position prediction for each argument role:

$$Loss_{arg}(\Theta_2, \Theta_4) = \sum_{k=1}^{K} w_a * Loss_{a_k^{(b)}}(\Theta_2, \Theta_4) + (1 - w_a) * Loss_{a_k^{(e)}}(\Theta_2, \Theta_4), \quad (16)$$

where $w_t \in (0, 1)$ is a tradeoff factor. For prediction, we simply set a threshold ϕ_{arg}, and select positions that prediction score higher than ϕ_{arg}. We just match the start position with the nearest end position to obtain a complete argument. We remove the redundant argument types for each event type based on the event schema constrain. The final results form the argument set A.

Training and Prediction To jointly learn the TEM and AEM, we combine both losses from the two modules as:

$$Loss(\Theta_2, \Theta_3, \Theta_4) = w_j * Loss_{tri}(\Theta_2, \Theta_3) + (1 - w_j) * Loss_{arg}(\Theta_2, \Theta_4) \quad (17)$$

where $w_j \in (0, 1)$ is a weight hyperparameter to balance the two modules.

We utilize groundtruth labels to train the overall model. For prediction, we first obtain trigger extraction results, and then input them into the argument extraction module. The results obtained from the two modules are returned as the final predictions.

2.4 Additional Strategy

Actually, despite the training datasets, the unlabeled data in the testing datasets also contains rich information. In order to exploit all the data to improve performance, we also employ the following strategies:

Continuing Pre-training on PLM: PLMs are usually pre-trained on the common corpus, which may causes semantic bias on the financial corpus. Therefore, we continue pre-training the PLM on all the financial data, including training data and testing data. This strategy is applied to both EDM and EEM.

Model Ensemble on Variant Data Splits: To fully exploit labeled data, we adopt K-fold validation on the labeled data, which also leads to K models trained on different data splits. Then, we ensemble K model predictions by the voting strategy. This model ensemble strategy is applied to EDM and EEM, separately.

Utilizing Pseudo-Labels on Unlabeled Data: To fully exploit unlabeled data, we employ a novel strategy to label testing data with pseudo labels. Specifically, we train models on the groundtruth data, and then predict labels on those unlabeled data, which is called pseudo-label data. By integrating pseudo-labeled data into groundtruth data, we obtain a mixed event dataset. We train new models on this mixed dataset. Note that this strategy is only used for EEM, where we achieve better performance.

source Types	质押 pledge	投资 investment	股份股权转让 share transfer	高管减持 reduction	起诉 prosecution
Data Size	815	1083	1581	670	533
target Types	收购 acquisition	判决 judgment	中标 win bid	签署合同 sign contract	担保 guarantee
Data Size	200	200	200	132	163

Table 1. Statistics of each event type in the dataset.

	training	validation	testing
source Types	2,459	273	163,763
target Types	738	82	93,610

Table 2. Data partition for training, validation and testing.

3 Experiment

This section introduces the dataset provided in the competition, and conducts experiments to evaluate the model.

3.1 Dataset

The dataset provided in the competition contains source event data and target event data, including labeled data and testing data for each event type. The statistics of each event type of labeled data is shown in Table (1). The competition only evaluates on the tesing target event data. For validation, we separate a part of labeled data as validation data. The details of data partition is shown in Table (2). Since the groundtruth of testing data not is available, all experiments below is conducted on the validation data.

3.2 Implementation

We utilize RoBERTa continue pre-trained on this financial data as PLM. For EDM, we set learning rate to 2e-5. The batch size is 16. For EEM, we apply learning rate of 2e-5 to PLM layer and 1e-4 to other layers. The batch size is 8. The tradeoff weight w_t, w_a, w_j is set to 0.5, 0.5, 0.2, respectively. Each kind of syntactic embedding dimension is set to 40. The relative position embedding dimension is set to 64. We apply dropout to SA layer and all input embeddings with the rate set to 0.3. With the model ensemble strategy, we train 5 EDMs for a better event type prediction. For EEM, we train 5 models, and ensemble the 5 results to obatin pseudo label on the testing data. Then, we train 10 EEMs on the mixed training data, and obtain 10 predictions on the testing data. We finally ensemble all 15 EEM results as the final submission.

3.3 Main Result

Since the groundtruth of testing data is not available, we conduct experiments on the validation data. The F1-score of event detection, trigger extraction and argument extraction on validation data is 0.921, 0.970, 0.889, respectively. The best result of our approach on official testing data is 0.8781 which is the highest score in the competition.

3.4 Ablation Study

	Trigger Extraction			Argument Extraction			
	P	R	F1	P	R	F1	F1-mean
1. complete model	**.969**	**.979**	**.970**	.844	**.969**	**.889**	**.930**
2. w/o pseudo-label data	.940	.952	.939	.845	.863	.838	.888
3. w/o source data	.941	.945	.934	.818	.865	.823	.878
4. repl PLM: BERT	.901	.924	.904	.789	.825	.789	.846
5. repl PLM: RoBERTa	.931	.938	.929	.837	.886	.828	.879
6. repl CLN: concat	.940	.952	.939	.845	.863	.838	.888
7. w/o layer lr	.946	.945	.940	.799	.869	.816	.878
8. w/o syntactic feature	.921	.924	.917	**.863**	.874	.856	.887

Table 3. Results on validation data.

We conduct ablation study on the event extraction model, where the results are shown in Table (3). Specifically, Line.1 shows the complete model, which is trained on both groundtruth data and pseudo-label data with all components. Line.2 removes the pseudo-label data, and the results show that utilizing pseudo-label data improves performance significantly. The following experiments is ablated based on Line.2. Line.3 removes source data in training, and the result indicates that learning target events with source events is effective. Line.4 and Line.5 replace the continue pre-trained PLM by BERT and standard RoBERTa, which indicates the effectiveness of continuing pre-training for PLM. Line.6 replaces CLN by a simple concatenate operation, which indicates CLN can utilize condition information more effectively. Line.7 applies the same learning rate to all layers, which indicates utilizing different learning rate on model layers benefits the learning process. Line.8 removes syntactic features, which indicates syntactic features improve the extraction performance. All results demonstrate that each component is effective in the event extraction task.

4 Conclusion

In this paper, we propose a financial event extraction approach based on a joint learning framework, which fully utilizes all the data to improve the performance of low-resource event types, and effectively solves the overlapping problem of events. The experimental results show that the approach achieves significantly performance, and it ranks the first place in the CCKS-2020 financial event extraction competition.

Acknowledgments

This work is supported by the National Natural Science Foundation of China (No.61772151) and the National Key Research and Development Program of China (No.2017YFB0803305). The computing device of this work is supported by Beijing Advanced Innovation Center of Big Data and Brain Computing, Beihang University.

References

1. Ba, L.J., Kiros, J.R., Hinton, G.E.: Layer normalization. CoRR **abs/1607.06450** (2016)
2. Chen, Y., Xu, L., Liu, K., Zeng, D., Zhao, J.: Event extraction via dynamic multi-pooling 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 of the Asian Federation of Natural Language Processing, ACL 2015. pp. 167–176 (2015)
3. Deng, S., Zhang, N., Kang, J., Zhang, Y., Zhang, W., Chen, H.: Meta-learning with dynamic-memory-based prototypical network for few-shot event detection. In: WSDM '20: The Thirteenth ACM International Conference on Web Search and Data Mining. pp. 151–159 (2020)
4. Huang, L., Ji, H., Cho, K., Dagan, I., Riedel, S., Voss, C.R.: Zero-shot transfer learning for event extraction. In: Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018. pp. 2160–2170 (2018)
5. Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., Stoyanov, V.: Roberta: A robustly optimized BERT pretraining approach. CoRR **abs/1907.11692** (2019)
6. Nguyen, T.H., Cho, K., Grishman, R.: Joint event extraction via recurrent neural networks. In: NAACL HLT 2016, The 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies. pp. 300–309 (2016)
7. Nguyen, T.M., Nguyen, T.H.: One for all: Neural joint modeling of entities and events. In: The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019. pp. 6851–6858 (2019)
8. Yu, B., Zhang, Z., Shu, X., Liu, T., Wang, Y., Wang, B., Li, S.: Joint extraction of entities and relations based on a novel decomposition strategy. In: ECAI 2020 - 24th European Conference on Artificial Intelligence. vol. 325, pp. 2282–2289 (2020)