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

Authors: Jiawei, Sheng, Qian, Li, Yiming, Hei, Shu, Guo, Yu Bowen, Lihong Wang, Min, He, Tingwen, Liu, Hongbo, Xu, Shu, Guo, Lihong Wang

Date: 2022-11-27T00:00:00+00:00

Abstract

This paper presents a winning solution for the CCKS-2020 financial event extraction task, where the goal is to identify 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. We further address the element overlapping problems by a mechanism of Conditional Layer Normalization, achieving even better extraction accuracy. The overall approach achieves an F1-score of 87.8%, which ranks first in the competition.

Full Text

Preamble

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

Jiawei Sheng[1],[2], Qian Li[3], Yiming Hei[4], Shu Guo[5]†, Bowen Yu[1],[2], Lihong Wang[5]†, Min He[5], Tingwen Liu[1],[2] & Hongbo Xu[1],[2]

[1]Institute of Information Engineering, Chinese Academy of Sciences, Beijing 100093, China
[2]School of Cyber Security, University of Chinese Academy of Sciences, Beijing 100049, China
[3]School of Computer Science, Beihang University, Beijing 100191, China
[4]School of Cyber Science and Technology, Beihang University, Beijing 100191, China

[5]National Computer Network Emergency Response Technical Team/Coordination Center of China, Beijing 100029, China

Keywords: Event detection; Event extraction; Joint learning; Financial event

Citation: Sheng, J.W., et al: A joint learning framework for the CCKS-2020 financial event extraction task. Data Intelligence 3(3), 444-459 (2021). doi: 10.1162/dint_a_{00098}

Received: February 13, 2021; **Revised:** April 22, 2021; **Accepted:** April 30, 2021

ABSTRACT

This paper presents a winning solution for the CCKS-2020 financial event extraction task, where the goal is to identify 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 across different event types in the model, we can learn to adapt to low-resource events based on high-resource events. We further address the element overlapping problem through a mechanism of Conditional Layer Normalization, achieving even better extraction accuracy. The overall approach achieves an F1-score of 87.8%, which ranks first place in the competition.

1. INTRODUCTION

The CCKS-2020 financial event extraction task aims to extract structural events by identifying event types, triggers, and arguments in sentences across multiple types. Figure 1 [Figure 1: see original paper] provides an example of event extraction for a financial news sentence. One structural event belongs to the type of 投资/investment, along with the trigger 收购/acquire and its arguments providing complementary details. Note that this sentence contains more than one event, and the triggers and arguments have overlaps across events.

The CCKS-2020 task provides two kinds of event sentences. The first contains five types of events associated with an abundant sentence corpus, called source events. The second contains another five types of events associated with a low-resource sentence corpus, called target events. Each type of event sentence is split into training (labeled data) and testing (unlabeled data) parts. Our goal is to evaluate event extraction performance on the test set of target events. This poses two main challenges compared to traditional event extraction tasks [1,2,3,4,5]:

First, the target events contain only 179 training sentences on average for each type. This limited supervision cannot provide sufficient contextual information

for event extraction. Second, elements can overlap with each other; that is, the same trigger or argument may belong to different events. As shown in the example, the trigger 收购/acquire and the argument 世纪华通/Shijihuatong belong to both event types of 投资/investment and 股份股权转让/share transfer. Performing event extraction with a simple sequence labeling method will cause label conflicts.

To address these challenges, we devised a joint learning method. In our approach, the overall framework is formulated as a joint probability model, which is decomposed into submodels; that is, the joint distribution is decomposed into a product of three conditional distributions. Each subtask represents a specific use of this distribution, including event type detection, trigger extraction, and argument extraction.

For the first subtask, given a financial news sentence, we classify the sentence into correct event types 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 BERT [6] 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 overlapping element issue, we introduce conditional layer normalization, extracting triggers only according to the specific event type and extracting arguments only according to the specific trigger. This method can extract elements separately under different conditions, avoiding overlap. 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 achieved an F1-score of 87.8%, which ranked first in the CCKS-2020 financial event extraction competition.

2. RELATED WORK

Traditional event extraction research is typically conducted in high-resource settings and assumes events appear in sentences without overlapped elements. These studies can be roughly categorized into two groups:

Traditional joint methods [4,5,7,8,9] perform trigger extraction and argument extraction simultaneously. They solve the task in a sequence labeling manner, extracting triggers or arguments by tagging the sentence only once. However, these methods fail to extract overlapped elements since overlapped elements cause label conflicts when forced to have more than one label.

Pipeline methods [1,10,11,12,13,14] perform trigger extraction and argument extraction in separate stages. Though pipeline methods enable extraction of overlapped elements in separate stages [14], they usually lack explicit dependencies between triggers and arguments and suffer from error propagation. All the above methods require sufficient training data to learn model parameters for each event type, and few methods can extract complex overlapped elements in event extraction.

Recently, several methods have been proposed to solve event extraction in various low-resource settings, such as few-shot learning [2], zero-shot learning [3,12], and incremental learning [15]. However, these methods cannot be directly transferred to this CCKS-2020 financial event extraction task because this task aims to extract low-resource target events with the help of rich-resource source events —a completely different setting compared to the above low-resource event extraction scenarios.

3. OUR APPROACH

We present the overview, the design of each component, and several improvement strategies.

3.1 Overview

Given a sentence denoted as s, we propose a joint learning approach to identify 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 event type detection, event trigger extraction, and event argument extraction:

$$P(C, T, A|s; \Theta) \propto P(C|s; \Theta_1) \cdot P(T|s, C; \Theta_2) \cdot P(A|s, C, T; \Theta_3)$$

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

3.2 Event Detection Model

To discover the event types occurring in the sentence, we adopted codes provided by the official competition as our event detection model (EDM). This model utilizes a pre-trained language model (PLM) to derive sentence representations, formulated as a multi-label multi-class text classification task. Specifically, given the sentence s, the probability of s belonging to a specific type c_m is calculated as in Equation (1):

$$p(c_m|s; \Theta_1) = \text{sigmoid}(w_m \cdot z_{\text{sent}})$$

where z_{sent} is the hidden state corresponding to the input token <CLS> in the PLM, which encodes the entire sentence representation of s; Θ_1 includes all parameters used in the PLM. For simplicity, we denote $p(c_m|s;\Theta_1)$ as p_m.

Then, we update and obtain the desired sentence representation z_{sent} by minimizing the following binary cross-entropy loss function with Equation (2):

$$\mathcal{L}_{\text{EDM}}(\Theta_1) = -\frac{1}{N}\sum_{n=1}^{N}\sum_{m=1}^{M}[y_{nm}\log p_m + (1-y_{nm})\log(1-p_m)]$$

where N is the number of training sentences; M is the number of pre-defined event types; y_{nm} is the true type label, which is either 0 or 1. During prediction, we simply set a threshold d and select the resultant event types C where each type c_m satisfies $p(c_m|s;\Theta_1) > d$.

3.3 Event Extraction Model

This section introduces our event extraction model (EEM), which achieves two subtasks through a pre-training/fine-tuning framework: trigger extraction and argument extraction. The pre-training part encodes sentence tokens as contextualized representations using the pre-trained language model BERT [6], which contains rich language knowledge widely used for natural language processing (NLP) tasks. The fine-tuning part is divided into three modules: a shared module to encode condition information based on conditional layer normalization, and two private modules to extract triggers and arguments. Both extraction modules have a similar model structure.

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

Since we have obtained event types C occurring in the given sentence s, we derive sentence representations conditioned on each specific event type $c \in C$ to avoid element overlapping issues.

To this end, we introduce a general module named conditional layer normalization (CLN) [16,17] to integrate such conditional information into sentence representation. CLN is mostly based on the well-known layer normalization [18] but can dynamically generate gain γ and bias β based on the condition information. Given a condition representation c and a sentence representation x, CLN is formulated in Equations (3) to (5):

$$\mu = \frac{1}{d}\sum_{i=1}^{d}x_i, \quad \sigma = \sqrt{\frac{1}{d}\sum_{i=1}^{d}(x_i-\mu)^2}$$

$$\gamma = W_\gamma c + b_\gamma, \quad \beta = W_\beta c + b_\beta$$

$$\text{CLN}(x, c) = \gamma \frac{x - \mu}{\sigma} + \beta$$

where x_i is the i-th dimensional element in x, $\gamma \in \mathbb{R}^d$ and $\beta \in \mathbb{R}^d$ 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.

We then 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/investment which is transformed into tokens 投 and 资. Then we concatenate these type tokens 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 its 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 with Equation (6):

$$H_{s\text{-typ}} = \text{CLN}(H_s, \text{MeanPooling}(H_c))$$

where $H_{s\text{-typ}}$ is the token representations conditioned on the event type c. This process 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.

3.3.2 Private Module The private module contains two sub-modules.

(1) Trigger Extraction Module (TEM)

This module extracts event triggers given the event type $c \in C$. To improve textual representations for trigger extraction, we adopt a self-attention (SA) layer. Thus the type-aware token representations can be enhanced as in Equations (7) and (8):

$$H_{\text{sa-typ}} = \text{SA}(H_{s\text{-typ}})$$

$$H_{\text{tri}} = H_{\text{sa-typ}} \oplus H_{\text{syn}}$$

where $\oplus$ is the concatenation operation. H_{syn} corresponds to the representation of syntactic features, obtained by the NLP tool LTP. The syntactic features include B/I/O labels of word segmentation, part-of-speech tagging, named entity

recognition, and dependency parsing, which are initialized randomly as learnable embeddings in our model.

To strengthen interactions among triggers of different event types, we predict triggers with the same trigger extractor. For each token, we adopt a pair of fully connected networks (FCN) to predict whether it is a "begin" or "end" position of a trigger, as summarized in Equations (9) and (10):

$$p_{t,i}^{(b)} = \text{sigmoid}(w_t^{(b)} \cdot h_{\text{tri},i}), \quad p_{t,i}^{(e)} = \text{sigmoid}(w_t^{(e)} \cdot h_{\text{tri},i})$$

where $h_{\text{tri},i}$ is the i-th element of H_{tri}, $w_t^{(b)}$ and $w_t^{(e)}$ are learnable parameters, and Θ_3 includes $w_t^{(b)}$, $w_t^{(e)}$, and all parameters in SA.

Then, a binary cross-entropy loss function is used for begin position prediction and end position prediction, denoted as $\mathcal{L}_t^{(b)}$ and $\mathcal{L}_t^{(e)}$. The final loss is defined as in Equation (11):

$$\mathcal{L}_{\text{TEM}}(\Theta_3) = w_t \cdot \mathcal{L}_t^{(b)}(\Theta_3) + (1 - w_t) \cdot \mathcal{L}_t^{(e)}(\Theta_3)$$

where $w_t \in (0, 1)$ is a trade-off factor. For prediction, we simply set a threshold ϕ_{tri} and select positions where prediction scores are higher than ϕ_{tri}. We match the begin 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 extracts arguments conditioned on one of the triggers T extracted from TEM. Given a specific trigger $t \in T$ in sentence s, we obtain trigger-aware sentence representation $H_{s\text{-tri}}$ conditioned on t, where the process follows Equation (7). We also utilize a self-attention layer to enhance the sentence representation, termed as $H_{\text{sa-tri}}$. To discern the position of trigger t, we further add its relative position embedding R, which measures the distance from the current position to the trigger position. The syntactic feature H_{syn} is also taken into consideration. Thus, the enhanced overall sentence representation is defined as in Equation (12):

$$H_{\text{arg}} = H_{\text{sa-tri}} \oplus R \oplus H_{\text{syn}}$$

For argument extraction, we extract all arguments with pairs of FCNs and devise them as in Equations (13) and (14):

$$p_{a,i}^{(b,k)} = \text{sigmoid}(w_a^{(b,k)} \cdot h_{\text{arg},i}), \quad p_{a,i}^{(e,k)} = \text{sigmoid}(w_a^{(e,k)} \cdot h_{\text{arg},i})$$

where $h_{\text{arg},i}$ is the i-th element of H_{arg}, $w_a^{(b,k)}$ and $w_a^{(e,k)}$ are learnable parameters for the k-th argument role, and Θ_4 includes $w_a^{(b,k)}$, $w_a^{(e,k)}$, and all parameters in

SA and CLN. Note that the number of pre-defined argument roles is K, and there are $2K$ prediction sequences for all argument roles.

The loss function is also binary cross-entropy for both begin and end position prediction for each argument role, calculated with Equation (15):

$$\mathcal{L}_{\text{AEM}}(\Theta_4) = w_a \cdot \mathcal{L}_a^{(b)}(\Theta_4) + (1 - w_a) \cdot \mathcal{L}_a^{(e)}(\Theta_4)$$

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

3.3.3 Training and Prediction To jointly learn TEM and AEM, we combine both losses from the two modules as in Equation (16):

$$\mathcal{L}_{\text{EEM}}(\Theta_3, \Theta_4) = w_j \cdot \mathcal{L}_{\text{TEM}}(\Theta_3) + (1 - w_j) \cdot \mathcal{L}_{\text{AEM}}(\Theta_4)$$

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

We utilize ground-truth 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 from both modules are returned as the final predictions.

3.4 Additional Improvement Strategy

Despite the training datasets, the unlabeled data in the testing datasets also contain rich information. To exploit all data to improve performance, we employed the following strategies:

(1) Continuing Pre-training on PLM

PLMs are typically pre-trained on common corpora, which may cause semantic bias on financial corpora. Therefore, we continued pre-training the PLM on all financial data, including training and testing data. This strategy was applied to both EDM and EEM.

(2) Model Ensemble on Variant Data Splits

To fully exploit labeled data, we adopted K-fold validation on the labeled data, leading to K models trained on different data splits. We then ensembled K model predictions through voting. This ensemble strategy was applied to EDM and EEM separately.

(3) Utilizing Pseudo-Labels on Unlabeled Data

To fully exploit unlabeled data, we employed a novel strategy to label testing data with pseudo-labels. Specifically, we trained models on ground-truth data and then predicted labels on unlabeled data, called pseudo-labeled data. By integrating pseudo-labeled data with ground-truth data, we obtained a mixed training dataset. We trained new models on this mixed dataset. Note that this strategy was only used for EEM, where we achieved better performance.

4. EXPERIMENT

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

4.1 Data Set

The dataset provided in the competition contains source event data and target event data, including labeled and testing data for each event type. Statistics for each event type in the labeled data are shown in Table 1 . The competition only evaluated on the testing target event data. For validation, we separated part of the labeled data as validation data. Details of the data partition are shown in Table 2 . Since the ground truth of the testing data was not available, all experiments below were conducted on the validation data.

4.2 Implementation

We utilized an extended BERT model as our PLM, which was pre-trained on a mixed large Chinese corpus (termed as BERT-ext) from the model zoo. We then continued pre-training it on this financial data.

For EDM, we set the learning rate to 2e-5 with a batch size of 16. For EEM, we applied a learning rate of 2e-5 to the PLM layer and 1e-4 to other layers, with a batch size of 8. The trade-off weights w_t, w_a, and w_j were set to 0.5, 0.5, and 0.2, respectively. Each syntactic embedding dimension was set to 40. The relative position embedding dimension was set to 64. We applied dropout to the SA layer and all input embeddings with a rate of 0.3. With the model ensemble strategy, we trained five EDMs for better event type prediction. For EEM, we trained 5 models and ensembled their results to obtain pseudo-labels on the testing data. We then trained 10 EEMs on the mixed training data and obtained 10 predictions on the testing data. We ensembled all 15 EEM results as the final submissions.

4.3 Main Result

Since the ground truth of testing data was not available, we conducted experiments on the validation data. The F1-scores of event detection, trigger extraction, and argument extraction on validation data were 0.921, 0.970, and 0.889, respectively.

The best result (F1-score) of our approach on official testing data was 0.8781, which was the highest score in the competition.

4.4 Ablation Study

We conducted an ablation study on the event extraction model, with results shown in Table 3 . Since the entire decoding process follows a pipelined paradigm, the performance of each submodel is affected by previously predicted results. To avoid this impact, we adopted ground-truth results as input for each submodel. Specifically, Line 1 in Table 3 shows the complete model trained on both ground-truth and pseudo-label data with all components. Line 2 removed the pseudo-label data, and results show that utilizing pseudo-label data significantly improved performance. Subsequent experiments were ablated based on Line 2. Line 3 removed source data in training, and results indicated that learning target events with source events was effective. Lines 4 and 5 replaced the continued pre-trained PLM with standard BERT and BERT-ext, respectively, suggesting the effectiveness of continued pre-training for PLM. Line 6 replaced CLN with a simple concatenation operation, indicating CLN can utilize condition information more effectively. Line 7 applied the same learning rate to all layers, indicating that utilizing different learning rates on model layers benefits the learning process. Line 8 removed syntactic features, showing that syntactic features improved extraction performance. All results demonstrated the effectiveness of each component in the task.

4.5 Case Study

To demonstrate model predictions, we conducted a case study. Figure 3 [Figure 3: see original paper] depicts an example of model results. Given an input sentence, the model sequentially performed event detection, trigger extraction, and argument extraction. The model first predicted all event types occurring in the sentence. Then, it extracted triggers according to each given event type. Next, the model extracted all arguments according to the given event type and trigger. This process extracted overlapped elements separately. Furthermore, by sharing parameters across different types in such a unified model, the model learned to extract low-resource events based on high-resource events.

Although the model attempted to solve the low-resource and element overlapping issues, two main error patterns existed: (1) error propagation: since subtasks were conducted in a cascading manner, errors in earlier predictions led to errors in subsequent predictions; (2) argument prediction errors: since arguments are complexly associated with their roles, the model tended to predict fewer arguments or predict arguments with wrong roles. We will attempt further improvements in the future.

5. CONCLUSION

In this paper, we proposed a financial event extraction approach based on a joint learning framework that fully utilizes all data to improve performance on low-resource event types and effectively solves the event overlapping problem. Experimental results show that the approach achieved significant performance and ranked first place in the CCKS-2020 financial event extraction competition.

ACKNOWLEDGEMENTS

This work is supported by the National Key Research and Development Program of China (No. 2016YFB1000105) and the National Natural Science Foundation of China (No. 61772151). This work' s computing device is also supported by Beijing Advanced Innovation Center of Big Data and Brain Computing, Beihang University. The author Shu Guo is supported by the "Zhizi Program."

AUTHOR CONTRIBUTIONS

This work was a collaboration of all authors. J.W. Sheng (shengjiawei@iie.ac.cn) proposed the joint learning framework idea and served as team leader in the CCKS-2020 competition. Q. Li (liqian@act.buaa.edu.cn) and Y.M. Hei (black@buaa.edu.cn) devised and implemented model details. S. Guo (guoshu@cert.org.cn) revised and proofread the manuscript. B.W. Yu (yubowen@iie.ac.cn) provided constructive solutions for model components. L.H. Wang (wlh@isc.org.cn) guided the competition team. M. He (heminsmile@163.com), T.W. Liu (liutingwen@iie.ac.cn), and H.B. Xu (hbxu@iie.ac.cn) provided insightful and constructive comments on the manuscript. All authors made meaningful and valuable contributions.

DATA AVAILABILITY STATEMENT

The datasets generated and/or analyzed during this study are not publicly available because they were produced by expert consultants of China Merchants Bank based on their own experience. The publicly released version requires consent from all expert consultants and is available from the corresponding author upon reasonable request.

REFERENCES

[1] Chen, Y., et al.: Event extraction via dynamic multi-pooling convolutional neural networks. In: Proceedings of the Annual Meeting of the Association for Computational Linguistics, pp. 167–176 (2015)

[2] Deng, S., et al.: Meta-learning with dynamic-memory-based prototypical network for few-shot event detection. In: WSDM ' 20: Proceedings of the 13th International Conference on Web Search and Data Mining, pp. 151–159 (2020)

[3] Huang, L., et al.: Zero-shot transfer learning for event extraction. In: Proceedings of the Annual Meeting of the Association for Computational Linguistics, pp. 2160–2170 (2018)

[4] Nguyen, T.H., Cho, K., Grishman, R.: Joint event extraction via recurrent neural networks. In: Proceedings of the Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT), pp. 300–309 (2016)

[5] Nguyen, T.M., Nguyen, T.H.: One for all: Neural joint modeling of entities and events. In: The 33rd AAAI Conference on Artificial Intelligence (AAAI-19), pp. 6851–6858 (2019)

[6] Devlin, J., et al.: BERT: Pre-training of deep bidirectional transformers for language understanding. In: Proceedings of the Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT), pp. 4171–4186 (2019)

[7] Li, Q., Ji, H., Huang, L.: Joint event extraction via structured prediction with global features. In: Proceedings of the Annual Meeting of the Association for Computational Linguistics, pp. 73–82 (2013)

[8] Liu, X., Luo, Z., Huang, H.: Jointly multiple events extraction via attention-based graph information aggregation. In: Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP), pp. 1247–1256 (2018)

[9] Sha, L., et al.: Jointly extracting event triggers and arguments by dependency-bridge RNN and tensor-based argument interaction. In: The 32nd AAAI Conference on Artificial Intelligence (AAAI-18), pp. 5916–5923 (2018)

[10] Du, X., Cardie, C.: Event extraction by answering (almost) natural questions. In: Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP), pp. 671–683 (2020)

[11] Li, F., et al.: Event extraction as multi-turn question answering. In: Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP), pp. 829–838 (2020)

[12] Liu, J., et al.: Event extraction as machine reading comprehension. In: Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP), pp. 1641–1651 (2020)

[13] Wadden, D., et al.: Entity, relation, and event extraction with contextualized span representations. In: Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing (EMNLP), pp. 5783–5788 (2019)

[14] Yang, S., et al.: Exploring pre-trained language models for event extraction and generation. In: Proceedings of the Annual Meeting of the Association for Computational Linguistics, pp. 5284–5294 (2019)

[15] Cao, P., et al.: Incremental event detection via knowledge consolidation networks. In: Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP), pp. 707–717 (2020)

[16] Su, J.L.: Conditional text generation based on conditional layer normalization (in Chinese). Available at: https://spaces.ac.cn/archives/7124. Accessed 21 May 2021

[17] Yu, B., et al.: Semi-open information extraction. In: Proceedings of WWW (2021). Available at: https://www2021.thewebconf.org/program/papers/. Accessed 21 May 2021

[18] Ba, L.J., Kiros, J.R., Hinton, G.E.: Layer normalization. arXiv preprint arXiv:1607.06450 (2016)

AUTHOR BIOGRAPHY

Jiawei Sheng is currently pursuing his PhD degree at the Institute of Information Engineering, Chinese Academy of Sciences. His research interests include information extraction, knowledge graph embedding, and knowledge acquisition. ORCID: 0000-0002-4865-982X

Qian Li is currently pursuing her PhD degree at the School of Computer Science, Beihang University. Her research interests include knowledge graph embedding and information extraction. ORCID: 0000-0002-1612-4644

Yiming Hei is currently pursuing his PhD degree at the School of Cyber Science and Technology, Beihang University. His research interests include knowledge graph embedding and information extraction. ORCID: 0000-0003-0794-9932

Shu Guo received her PhD degree from the Institute of Information Engineering, Chinese Academy of Sciences. She is currently working at the National Computer Network Emergency Response Technical Team/Coordination Center of China. Her research interests include knowledge graph embedding, knowledge acquisition, and Web mining. ORCID: 0000-0002-9660-0291

Bowen Yu is currently pursuing his PhD degree at the Institute of Information Engineering, Chinese Academy of Sciences. His research interests include information extraction, metric learning, and unsupervised learning. ORCID: 0000-0002-6804-1859

Lihong Wang is a Professor at the National Computer Network Emergency Response Technical Team/Coordination Center of China. Her research interests include big data mining and analytics, information retrieval, and natural language processing. ORCID: 0000-0003-0179-2364

Min He is currently working at the National Computer Network Emergency Response Technical Team/Coordination Center of China. Her research interests include natural language processing, Web mining, and information security.

Tingwen Liu is an Associate Researcher at the Institute of Information Engineering, Chinese Academy of Sciences. His research interests include information extraction and knowledge fusion, text understanding, semantic computing, and heterogeneous network representation.

Hongbo Xu is a researcher at the Institute of Information Engineering, Chinese Academy of Sciences. His research interests include knowledge graph and Web mining.

Note: Figure translations are in progress. See original paper for figures.