

KeyEE: Enhancing Low-Resource Generative Event Extraction with Auxiliary Keyword Sub-Prompt

Junwen Duan, Xincheng Liao, Ying An, and Jianxin Wang*

Abstract: Event Extraction (EE) is a key task in information extraction, which requires high-quality annotated data that are often costly to obtain. Traditional classification-based methods suffer from low-resource scenarios due to the lack of label semantics and fine-grained annotations. While recent approaches have endeavored to address EE through a more data-efficient generative process, they often overlook event keywords, which are vital for EE. To tackle these challenges, we introduce KeyEE, a multi-prompt learning strategy that improves low-resource event extraction by Event Keywords Extraction(EKE). We suggest employing an auxiliary EKE sub-prompt and concurrently training both EE and EKE with a shared pre-trained language model. With the auxiliary sub-prompt, KeyEE learns event keywords knowledge implicitly, thereby reducing the dependence on annotated data. Furthermore, we investigate and analyze various EKE sub-prompt strategies to encourage further research in this area. Our experiments on benchmark datasets ACE2005 and ERE show that KeyEE achieves significant improvement in low-resource settings and sets new state-of-the-art results.

Key words: natural language processing; Event Extraction (EE); Multi-Prompt Learning (MPL); low-resource

1 Introduction

Event Extraction (EE) presents a significant challenge in natural language understanding. Its objective is to extract triggers, arguments, and their respective role types in a structured format from unstructured text. For instance, Fig. 1 demonstrates an event extraction example featuring two event types: Conflict:Attack and Life:Die, with corresponding details in the passage. The Conflict:Attack event is triggered by the term “detonated” and it involves three argument roles:

- Junwen Duan, Xincheng Liao, and Jianxin Wang are with the Hunan Key Laboratory of Bioinformatics, School of Computer Science and Engineering, Central South University, Changsha 410083, China. E-mail: jwduan@csu.edu.cn; ostars@csu.edu.cn; jxwang@mail.csu.edu.cn.
- Ying An is with the Big Data Institute, Central South University, Changsha 410083, China. E-mail: anying@csu.edu.cn.

* To whom correspondence should be addressed.

Manuscript received: 2023-07-26; revised: 2023-11-06;
accepted: 2023-11-28

“man” as the “Attacker”, who attacks “soldiers” as the “Target”, using “explosives” and “car” as the “Instrument”. Owing to the monumental success of neural networks and deep learning across a vast array of domains^[1–5], contemporary researchers predominantly employ neural networks for event extraction, which can be divided into classification-based method and generation-based method.

To address the complexity of event structure, classification-based method divides EE into various classification subtasks, such as entity recognition, event detection, and argument classification. Subsequently, the results from these subtasks are combined to generate the final predicted event record. Some methods follow a pipeline approach^[6–9], while others adopt a joint manner^[10–13]. Although these methods perform well with ample annotated data, their performances significantly decline when faced with limited training data. Unfortunately, obtaining high-quality event annotations is expensive and not always feasible.

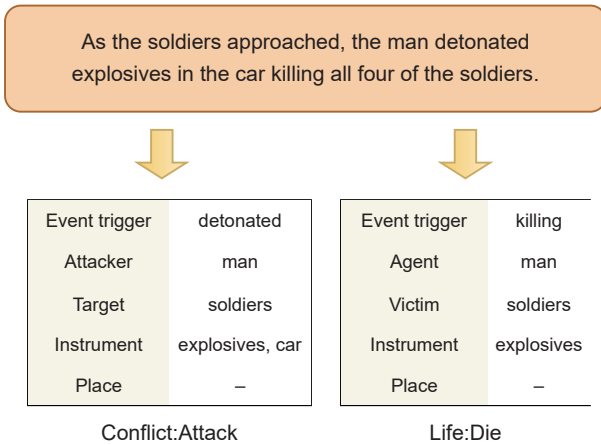


Fig. 1 Two event records (Conflict:Attack and Life:Die) extracted from the given passage. Each event consists of triggers and arguments, and each argument plays a specific role in the event.

Over the past two years, generative models have gradually emerged and achieved commendable performance in a multitude of tasks^[14–16]. Inspired by these works, recent researchers have turned to solve EE using a conditional generation approach. Generation-based methods involve incorporating additional knowledge into prompts^[17, 18] or capturing dependencies within a unified information extraction framework^[19, 20]. These approaches prove to be more data-efficient and require fewer annotated data to achieve comparable performance. Notably, event keywords have been found to benefit EE in DEGREE^[17], where the authors used three static triggers as event keywords, improving the event trigger extraction performance. However, they only considered triggers as keywords, neglecting the importance of arguments, and static keywords provide limited help when they are absent from the context.

To tackle the aforementioned challenges, we propose KeyEE, a multi-prompt learning strategy that enhances low-resource event extraction through Event Keyword Extraction (EKE). Unlike DEGREE^[17], which uses only three static triggers as event keywords, KeyEE considers both triggers and arguments. Moreover, instead of directly inserting event keywords into the EE prompt^[17], KeyEE employs an auxiliary EKE sub-prompt, which dynamically extracts event keywords from the input context. Consequently, we train KeyEE with both the EE prompt and the auxiliary EKE sub-prompt, implicitly incorporating event keyword knowledge into the model. This approach offers the

following advantages: (1) it incorporates event description and label semantics into the input prompt, reducing the reliance on annotated data compared to classification-based methods; (2) it provides dynamic event keyword knowledge for the model, thereby further enhancing event extraction performance in low-resource scenarios when compared to other generation-based methods. Our main contributions are as follows:

(1) We propose KeyEE, a simple yet effective multi-prompt learning strategy for low-resource generation-based event extraction, which implicitly provides additional event keywords knowledge for low-resource EE.

(2) We explore various EKE sub-prompt strategies, and conduct comprehensive experiments to compare and analyze their impact and limitations on event extraction performance, hoping to inspire further research on this direction.

(3) Experiments on ACE2005^[21] and ERE^[22] demonstrate the effectiveness of our method in both high-resource and low-resource scenarios, where we achieve a new state-of-the-art result.

Our code and models will be available at <https://github.com/OStars/KeyEE>.

2 Related Work

Our work is related to event extraction and multi-prompt learning, where event extraction can be further divided into classification-based methods and generation-based methods.

2.1 Classification-based method

Most classification-based methods split EE into multiple classification subtasks. Wang et al.^[8] and Yang et al.^[9] solved each subtask one by one in a pipeline which can not avoid error propagation. As a result, researchers attempted to design a dedicated joint framework to solve subtasks simultaneously^[11–13, 23], which also can capture dependencies between different subtasks. OneIE^[10] is an outstanding joint model incorporated with global features which achieved state-of-the-art performance. Moreover, recently some researchers^[6, 7] convert event extraction to Question Answering (QA) problem, which could make use of the label semantics but get a high time cost. The above classification-based methods need abundant fine-grained annotations for different subtasks which cause poor performance in low-resource scenarios.

2.2 Generation-based method

Generation-based methods are more data-efficient and can be divided into three categories. (1) Event structure generation: TEXT2EVENT^[24] presents events in a linearized format, and then generates the linearized structure end-to-end using the constrained decoding algorithm. (2) Unified information extraction framework: TanL^[19] and UIE^[20] propose a set of schema that enables the generative model to handle many other information extraction tasks, including event extraction tasks. (3) Template-based generation: Liu et al.^[18] and Li et al.^[25] formulated the EE as conditional generation following event templates, where GTEE-DYNPREF^[18] extra uses prefix-tuning^[26] and multi-head attention^[27] to capture dependencies between different events. Our work falls under template-based generation. The most similar work to ours is DEGREE^[17], a template-based model using static event keywords. In contrast, our dynamic EKE sub-prompt strategy is more adept at assisting the model in comprehending event keywords and effectively responding to varying contexts.

2.3 Multi-prompt learning

Multi-Prompt Learning (MPL) is a research area that uses multiple prompts to enhance the effectiveness of prompting methods^[28]. There are four main strategies for expanding a single prompt into multiple prompts: (1) Prompt ensemble, which combines several unanswered prompts at inference time. The responses to all prompts are then aggregated to form the final prediction^[29, 30]. (2) Prompt augmentation, which provides some prompt examples before the input to illustrate how the language model should respond to the actual prompt^[31, 32]. (3) Prompt composition, which employs multiple sub-prompts that are combined to create a composite prompt that serves as the input^[33]. (4) Prompt decomposition, which divides the holistic prompt into various sub-prompts, each of which is answered separately^[34]. We extend the above ideas for classification tasks to generation-based EE. Moreover, instead of using multiple prompts to achieve an answer, we propose an auxiliary EKE sub-prompt to enhance major EE prompt learning.

3 Keywords-Enhanced Generative Event Extraction

In this section, we revisit how to cast event extraction

into a template-based conditional generation task and introduce our MPL strategy based on dynamic EKE to learn implicit keyword knowledge. Finally, we give details on keywords-enhanced training and inference.

3.1 Problem statement

We focus on sentence-level event extraction and model event extraction as a template-based conditional generation task. Generally, given an event type e , we use an encoder-decoder based Pre-trained Language Model (PLM), such as BART^[35] and T5^[36], to generate corresponding events end-to-end. The input consists of an event-type-specific prompt P and context C . The output is a filled-in event-type-specific template T that can be parsed to event records with a deterministic algorithm. Formally, the model calculates the conditional probability p of choosing a new token y_t in t -th time step, considering the preceding tokens $y_{<t}$ and the encoder input X ,

$$P(Y|X) = \prod_{t=1}^{|Y|} p(y_t | y_{<t}, X) \quad (1)$$

where $X = [C; P]$ is the concatenation of context and the event-type-specific prompt, and $Y = T$ is the output template filled with golden labels.

3.2 Template-based generative baseline

The architecture of template-based conditional generation is illustrated in Fig. 2a, which shows the following components from left to right: context, type-specific input prompt, model, and type-specific filled-in output template. The type-specific input prompt is designed to provide ample knowledge to the model, while the type-specific output template governs the generation of valid output that complies with event ontology. We construct an input prompt for each event type. To reduce the need for annotated data and capture the dependencies among entities, triggers and arguments, the input prompt consists of three parts:

- Event type description gives a definition for a given event type. For instance, the Life:Die event description is “The event is related to life and someone died”.
- Event keywords point to the key components of the event directly. DEGREE^[17] takes three preset triggers as event keywords where the event keywords for Life:Die are “Similar triggers such as kill, death, assassination”. Different from the static event keywords, we generate event keywords according to

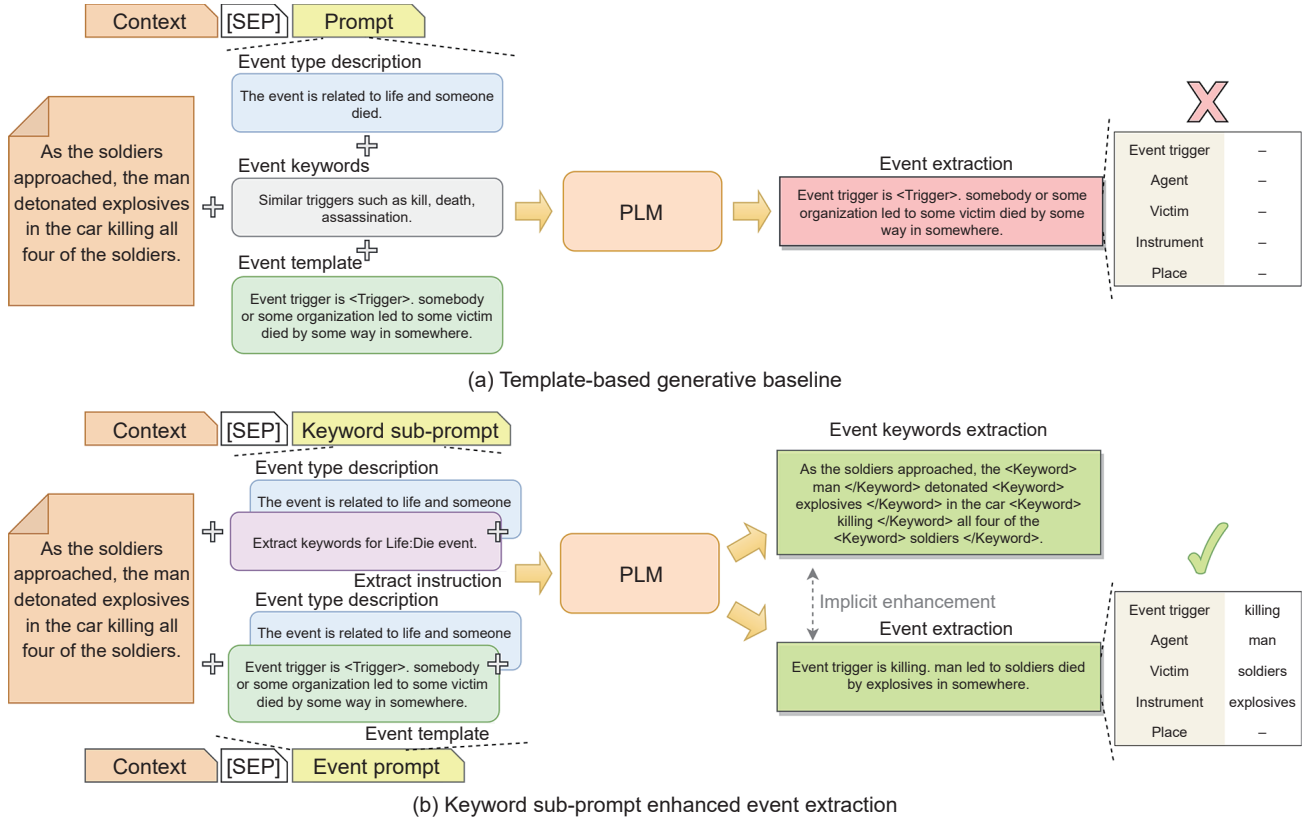


Fig. 2 Comparison of generation-based method with (a) static event keywords and (b) KeyEE (ours) with dynamic event keywords. The “+” in the figure indicates concatenation operation. The figure shows an example of extracting Life:Die event, which is triggered by “killing”, “man” plays the role of “Agent”, “soldiers” plays the role of “Victim”, and “explosives” plays the role of “Instrument”, while there is no argument plays the role of “Place” in the context. As a result, the right output template is “Event trigger is killing. man led to soldiers died by explosives in somewhere”. Static keywords not appearing in the context give the model limited knowledge, resulting in the wrong extraction. KeyEE extracts event keywords dynamically according to the context, which provides sufficient knowledge for the model, resulting in the right extraction.

context dynamically, which further improves the event extraction performance (see Section 3.3).

- Event template is designed following DEGREE^[17]. The template comprises two main components: the trigger template and the argument template. The former remains consistent across all event types, taking the form of “Event trigger is <Trigger>”, where <Trigger> acts as a special token representing the golden triggers placeholder. The latter varies for each event type, offering a detailed description of the processes in different events using natural language. This approach effectively leverages the intrinsic knowledge of PLM, enhancing the model’s comprehension of events. For instance, the Life:Die event argument template reads as follows: “somebody or some organization led to some victim died by some ways in somewhere”. Here, the tokens with the “some” prefix act as golden arguments placeholders, indicating the argument role type.

The golden type-specific output template employs the same event template as the input prompt, but it replaces the placeholders with golden labels. Following DEGREE^[17], if multiple arguments exist for the same role, we concatenate them using “and”. If a role lacks a predicted argument, the model retains the corresponding placeholder. In instances where multiple triggers of the same event type exist, the model generates separate output templates for each trigger and its associated arguments.

3.3 Event keywords extraction

Leveraging the supplementary keywords knowledge within the input prompt, the template-based generative baseline demonstrates superior performance over classification-based approaches in low-resource scenarios. However, there are two primary limitations that should be addressed: firstly, it solely takes event triggers into account as keywords, overlooking the

significance of event arguments; secondly, the keywords remain static for all input contexts, thereby providing limited knowledge when the input context lacks keywords (see Fig. 2a).

To address the two limitations, we propose an auxiliary EKE sub-prompt to dynamically generate type-specific event keywords based on the input context. It considers both triggers and arguments as event keywords, ensuring the model pays more attention to the keywords rather than other tokens. Specifically, we use a pair of special tokens, i.e., $\langle \text{Keyword} \rangle$ and $\langle / \text{Keyword} \rangle$ to label the event keywords corresponding to the given event type in the context (see Fig. 3).

Similar to the model input Section 3.2, KeyEE takes the concatenation of the context and the auxiliary keyword sub-prompt as input, and outputs the keywords-enhanced context (see Fig. 2b). While in EKE, the auxiliary keyword sub-prompt consists of two parts:

- Event type description is the same as that in EE.
- Extraction instruction guides the model to generate event keywords rather than events like “Extract keywords for $\langle \text{event_type} \rangle$ event”, where $\langle \text{event_type} \rangle$ is a placeholder of the event type.

Based on this strategy, KeyEE possesses the following advantages in low-resource scenarios: (1) KeyEE furnishes the model with event definition and label semantics through the input prompt, leading to a better understanding of events by the model. (2) KeyEE considers both triggers and arguments as event

keywords, providing the model with comprehensive knowledge. (3) Our dynamic event keywords are more precise and closely tied to the events. Moreover, the MPL strategy shields EE from the direct impact of inappropriate event keywords. In this MPL set, we calculate two losses to optimize KeyEE during the training stage. During inference, we only perform event extraction.

3.4 Keywords-enhanced training, inference, and parsing

(1) Training

All trainable parameters in KeyEE are the parameters θ of PLM^[35]. There are two kinds of prompts in KeyEE, the main prompt is for event extraction with training samples $D_E = (x_e^1, y_e^1), (x_e^2, y_e^2), \dots, (x_e^{|D_E|}, y_e^{|D_E|})$, where x_e is the concatenation of input context C and event prompt P_e , y_e is the filled event template, and $|D_E|$ is the number of training examples. The auxiliary prompt is for event keywords extraction with training samples $D_K = (x_k^1, y_k^1), (x_k^2, y_k^2), \dots, (x_k^{|D_K|}, y_k^{|D_K|})$, where x_k is the concatenation of input context C and keyword sub-prompt P_k , y_k is the context with $\langle \text{Keyword} \rangle$ labels, and $|D_K|$ is the number of training examples. The whole learning objective is the negative log-likelihood function,

$$\mathcal{L} = - \sum_{(x_e, y_e) \in D_E} \log P(y_e | x_e; \theta) - \sum_{(x_k, y_k) \in D_K} \log P(y_k | x_k; \theta) \quad (2)$$

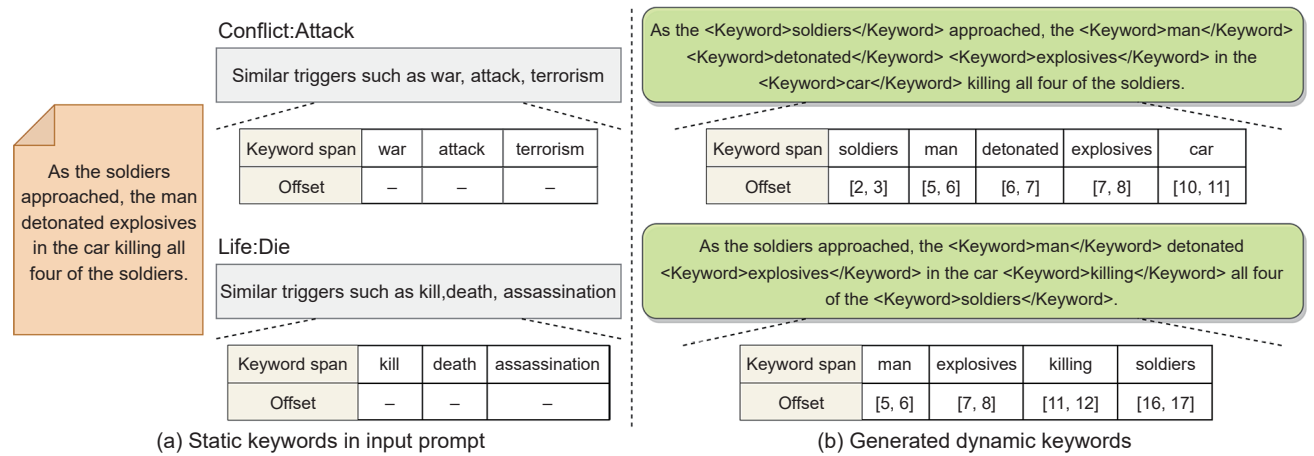


Fig. 3 Comparison of static keywords (grey rectangle) and dynamic keywords (green rounded rectangle). The keyword span and offset indicate the extracted keywords and their corresponding position in the original context, respectively, where “—” in the offset indicates the keyword does not appear in the context. The static keywords are all the same for different contexts. While our dynamic keywords are adjusted as the context.

(2) Inference

During the inference stage, we opt for probability maximization-based decode instead of beam search. This decision is based on previous research^[37], which indicates that beam search does not consistently yield the best results, and the scores of generated candidates may not serve as a reliable ranking indicator for event triggers^[38].

(3) Parsing

First, we obtain the predicted trigger words and their parameters by parsing the model output according to a predefined output template. After that, we conduct string matching to obtain the offsets of the spans in the previous sentence. If the context contains multiple spans of the same predicted trigger, we default to the first occurrence span as the final event trigger. If there are multiple spans with the same prediction argument in the context, we select the span closest to its corresponding trigger as the final event argument.

4 Experiment

4.1 Experimental setting

(1) Datasets

We conduct experiments on two widely used event extraction benchmarks, ACE2005^[21] and ERE^[22]. ACE2005 encompasses 599 annotated documents and 33 event types, while ERE comprises 458 English documents and 38 event types. We follow the pre-process method^[10, 13, 17], resulting in ACE2005-EN, ACE2005-EN+, and ERE-EN. Compared to ACE2005-EN, both ACE2005-EN+ and ERE-EN consider the order of relation arguments, pronouns, and multi-token event triggers, which have been largely ignored in

previous work^[10].

(2) Low-resource setting

Our low-resource setting is the same as DEGREE^[17]. According to the proportion of the number of documents, 1%, 3%, 5%, 10%, 20%, and 30% of the original training set are selected as the low-resource training set. Our model is trained on each of the six low-resource training sets described above, and performance is evaluated on the original complete validation set and test set. Statistics for both the full data set and the low-resource data set are shown in Table 1.

(3) Evaluation metrics

We use the same evaluation criteria in previous work^[10, 17, 20, 24]. Trigger F1-score is classified (Tri-Cls) if and only if its offset and type are both predicted correctly. Argument F1-score is classified (Arg-Cls) if and only if its offset and argument role type are both predicted correctly. Moreover, in order to evaluate the performance of EKE, we propose keyword F1-score, which is classified (Key-Cls) if and only if its offset is predicted correctly. Given the high variance in low-resource training, we conduct each experiment with three different random seeds and report the average metric.

(4) Implementation details

For all experiments, we fine-tune the pre-trained BART-large^[35] with Huggingface^[39]. The whole trainable parameters are around 4.0×10^8 . We consider AdamW optimizer^[40] to optimize KeyEE. The hyperparameters are shown in Table 2. We train KeyEE on an NVIDIA A100 Tensor Core GPU 40GB.

(5) Compared models

We consider both classification-based models and

Table 1 Dataset statistics for all splits used in our experiments, which are conducted on sentences extracted from documents, where “#Docs” represents the number of documents, “#Sents” denotes the number of sentences, “#Events” signifies the number of events, and “#Args” indicates the total number of arguments.

Split	ACE05-EN				ACE05-EN+				ERE-EN			
	#Docs	#Sents	#Events	#Args	#Docs	#Sents	#Events	#Args	#Docs	#Sents	#Events	#Args
All training sets	529	17 172	4202	4859	529	19 216	4419	6607	396	14 736	6208	8924
1% of training set	5	103	47	65	5	92	49	75	4	109	61	78
3% of training set	15	451	119	153	15	434	124	203	12	419	179	272
5% of training set	25	649	212	228	25	628	219	297	20	701	437	640
10% of training set	50	1688	412	461	50	1915	428	629	40	1536	618	908
20% of training set	110	3467	823	936	110	3834	878	1284	80	2848	1231	1656
30% of training set	160	5429	1368	1621	160	6159	1445	2212	120	4382	1843	2632
Validation	28	923	450	605	28	901	468	759	31	1209	525	730
Test	40	832	403	576	40	676	424	689	31	1163	551	822

Table 2 Hyperparameter settings used in KeyEE.

Dataset	Parameter						
	Learning rate	Train batch size	Number of train epochs	Weight decay	Gradient clip	Number of warm-up epochs	Max output length
ACE2005-EN	10^{-5}	16	45	10^{-5}	5	5	100
ACE2005-EN+	10^{-5}	24	45	10^{-5}	5	5	100
ERE-EN	10^{-5}	16	45	10^{-5}	5	5	110

generation-based models as our comparison objects. The classification-based models include: (a) OneIE^[10]: It is a joint information extraction model, decodes with global features to capture cross-task and cross-instance dependencies, which achieves state-of-the-art results in the classification-based model. (b) MQAEE^[6]: It is a model, converts event extraction to a multi-turn question-answering problem, which is easy to model label semantics. (c) Joint3EE^[11]: It is a joint model, shares hidden representation for entities, triggers, and arguments, capturing dependencies between different tasks to enhance EE. (d) TBDEE^[8]: It is a pipeline model, refines event extraction as type-oriented binary decoding, which can make use of label semantics and extend to new event types.

The generation-based models include: (a) TANL^[19]: A trigger-argument pipeline model converting event extraction to augmented natural languages translation problem. (b) Text2Event^[24]: A sequence-to-structure model converting event records into a linear tree-like

format and extracts them end-to-end. (c) UIE^[20]: A unified information extraction model combining many different information extraction tasks into one extraction schema, demonstrating strong performance in multiple tasks. (d) GTEE-DYNPREF^[18]: A dynamic prefix-tuning based model replacing manual prompt with the trainable prefix to learn the optimal prompt and captures dependencies among different type events. (e) DEGREE^[17]: A template-based generative model for low-resource EE incorporating static event keywords information into the model. It is the most similar model to ours, which

4.2 Low-resource result

Tables 3 and 4 present all baselines and KeyEE low-resource Arg-Cls and Tri-Cls on ACE2005-EN, ACE2005-EN+, and ERE-EN. KeyEE achieves state-of-the-art results 18 times and 10 times in argument classification and in trigger classification over 18 splits, respectively, demonstrating its effectiveness in

Table 3 Event argument classification F1-score in low-resource scenarios. Bold indicates the highest scores, “Cls” and “Gen” represent the classification-based methods and the generation-based methods, respectively. The baseline results are from DEGREE^[17].

(%)										
Model	Type	ACE2005-EN			ACE2005-EN+			ERE-EN		
		1% of training set	3% of training set	5% of training set	1% of training set	3% of training set	5% of training set	1% of training set	3% of training set	5% of training set
OneIE ^[10]	Cls	9.4	22.0	26.8	10.4	20.6	29.7	2.6	20.3	29.7
Text2Event ^[24]	Gen	3.9	12.2	19.1	5.7	16.5	21.3	2.3	15.2	23.6
TanL ^[19]	Gen	8.5	17.2	24.7	8.6	22.3	30.4	1.4	20.2	29.5
DEGREE ^[17]	Gen	21.7	30.1	35.5	18.7	34.0	35.7	14.5	28.9	33.4
Ours	Gen	21.9	34.6	38.9	23.1	38.3	40.3	15.7	33.3	36.2

Model	Type	ACE2005-EN			ACE2005-EN+			ERE-EN		
		10% of training set	20% of training set	30% of training set	10% of training set	20% of training set	30% of training set	10% of training set	20% of training set	30% of training set
OneIE ^[10]	Cls	26.8	42.7	47.8	35.5	46.7	48.0	35.1	40.7	43.0
Text2Event ^[24]	Gen	24.9	32.3	39.2	26.4	35.2	42.1	28.7	35.7	38.7
TanL ^[19]	Gen	29.0	34.0	39.2	29.2	34.6	39.0	30.1	35.6	36.9
DEGREE ^[17]	Gen	41.6	46.2	48.7	43.6	48.9	51.2	41.7	42.9	45.5
Ours	Gen	45.7	48.5	49.9	48.0	49.8	51.3	42.0	43.9	47.3

Table 4 Event trigger classification F1-score in low-resource scenarios. Bold indicates the highest scores, “Cls” and “Gen” represent the classification-based methods and the generation-based methods, respectively. The baseline results are from DEGREE^[17].

(%)										
Model	Type	ACE2005-EN			ACE2005-EN+			ERE-EN		
		1% of training set	3% of training set	5% of training set	1% of training set	3% of training set	5% of training set	1% of training set	3% of training set	5% of training set
OneIE ^[10]	Cls	38.5	52.4	59.3	39.0	52.5	60.6	11.0	36.9	46.7
Text2Event ^[24]	Gen	14.2	35.2	46.4	15.7	38.4	43.9	6.3	25.6	33.5
TanL ^[19]	Gen	34.1	48.1	53.4	30.3	50.9	53.1	5.7	30.8	43.4
DEGREE ^[17]	Gen	55.4	62.1	65.8	49.5	63.5	62.3	27.9	45.5	47.0
Ours	Gen	50.9	61.3	63.0	53.5	62.2	61.5	31.2	46.8	48.1

Model	Type	ACE2005-EN			ACE2005-EN+			ERE-EN		
		10% of training set	20% of training set	30% of training set	10% of training set	20% of training set	30% of training set	10% of training set	20% of training set	30% of training set
OneIE ^[10]	Cls	61.5	67.6	67.4	58.1	66.5	66.4	48.8	51.8	53.5
Text2Event ^[24]	Gen	47.0	55.6	60.7	46.3	56.5	62.0	42.4	46.7	50.1
TanL ^[19]	Gen	54.8	61.8	61.6	55.7	60.8	61.7	45.9	49.0	49.3
DEGREE ^[17]	Gen	65.8	68.3	68.2	68.5	67.6	66.9	53.0	51.7	53.5
Ours	Gen	68.7	68.0	68.4	68.2	67.6	68.0	53.1	51.2	55.8

low-resource scenarios. Compared to the classification-based method OneIE^[10], KeyEE achieves significant improvement (over 10%) with the extra knowledge in the input prompt and EKE when there is less than 10% training data. While those generative models (Text2Event^[24] and TanL^[19]) that provide no additional knowledge to the model perform even worse in the above scenarios.

Compared to DEGREE^[17] which uses static keywords, KeyEE outperforms it in most splits because our dynamic EKE captures more accurate event keywords and helps KeyEE understand the context better. We found that the improvement in Arg-Cls is greater than that in Tri-clis, as KeyEE considers both triggers and arguments as event keywords, while DEGREE^[17] only considers triggers. Our MPL and dynamic EKE enhance both Tri-Cls and Arg-Cls. However, we notice that KeyEE Tri-Cls performs worse than DEGREE^[17] when the training data are less than 10%. This inferior performance primarily occurs in the ACE2005 dataset, mainly due to DEGREE’s selection of the three most frequent triggers from the dataset as static event keywords. In situations with extremely limited data, these static keywords carry even greater significance, providing direct assistance to the model. Conversely, compared to ACE2005, the ERE-EN dataset contains more complex sentences and noise. Consequently, KeyEE, which employs dynamic

event keywords, surpasses DEGREE Tri-Cls under conditions of extremely limited data in ERE-EN dataset.

The most significant Arg-Cls gains are evident when training data is less than 10%. For instance, KeyEE achieves 4.5% and 4.6% Arg-Cls improvements in 3% of the original training set on ACE2005-EN and 5% of the original training set on ACE2005-EN+, respectively. This highlights the importance of EKE which provides sufficient event knowledge for the model when training data is insufficient. However, the improvement in both Tri-Cls and Arg-Cls within the ERE dataset is less significant compared to ACE2005. The longer sentences in ERE-EN pose a greater challenge for the model to label event keywords accurately within the original context, thus making it harder for KeyEE to furnish precise knowledge in ERE-EN.

4.3 High-resource result

To gain a deeper understanding of KeyEE’s capabilities, we conduct experiments in high-resource scenarios. Table 5 shows that KeyEE’s performance is comparable to the baselines. Notably, in ACE2005-EN and ACE2005-EN+, KeyEE establishes two new SOTA Arg-Cls results, with 0.7% and 1.3% improvements, respectively. It demonstrates our MPL strategy still works well and provides sufficient

Table 5 Event trigger and argument classification F1-score in high-resource scenarios. Bold indicates the highest scores, “Cls” and “Gen” represent the classification-based methods and the generation-based methods, respectively. DEGREE Δ is the result from the original paper, while DEGREE $\diamond$ is the result of running the default configuration based on the open source code. The other baseline results are from DEGREE^[17] and the original paper.

Model	Type	ACE2005-EN		ACE2005-EN+		ERE-EN	
		Tri-Cls	Arg-Cls	Tri-Cls	Arg-Cls	Tri-Cls	Arg-Cls
Joint3EE ^[11]	Cls	69.8	52.1	–	–	–	–
MQAEE ^[7]	Cls	71.7	53.4	–	–	–	–
OneIE ^[10]	Cls	74.7	56.8	72.8	54.8	57.0	46.5
TBDEE ^[8]	Cls	–	–	73.1	55.1	60.4	50.4
TANL ^[19]	Gen	68.4	47.6	68.6	46.0	54.7	43.2
UIE ^[20]	Gen	73.4	54.8	–	–	–	–
Text2Event ^[24]	Gen	71.9	53.8	71.8	54.4	59.4	48.3
DEGREE Δ ^[17]	Gen	73.3	55.8	70.9	56.3	57.1	49.6
DEGREE $\diamond$ ^[17]	Gen	73.9	55.7	69.5	54.9	55.0	47.7
GTEE-DYNPREF ^[18]	Gen	72.6	55.8	74.3	54.7	66.9	55.1
Ours	Gen	73.2	57.5	71.3	57.6	58.0	49.1

knowledge for the model in high-resource scenarios. Similar to low-resource scenarios, KeyEE achieves a better improvement in Arg-Cls compared to Tri-Cls, which shows the effectiveness of adding arguments as event keywords. These findings are consistent across both low and high-resource scenarios: KeyEE outperforms in Arg-Cls over Tri-Cls, and exhibits greater proficiency with ACE2005 than ERE-EN due to sentence length.

4.4 Impact of EKE

KeyEE uses the auxiliary keyword sub-prompt to learn implicit event keyword knowledge from EKE, we study the impact of the performance of EKE on EE in this section.

Figure 4 shows the changes in EKE performance and its relationship with EE. Both EE and EKE exhibit a similar trend of improvement with the expansion of training data. Notably, there is a significant enhancement in both EE and EKE between 1% and 3% of the training data, indicating our model’s ability to achieve considerable gains even with extremely limited training data.

Furthermore, Key-Cls and Arg-Cls maintain a similar performance in all data splits, indicating that event argument extraction is challenging for both EE and EKE. This highlights the importance of treating event arguments as event keywords in event extraction, as event arguments play an important role in the overall performance. However, Key-Cls performs poorer than Arg-Cls in ACE2005-EN, which can be attributed to

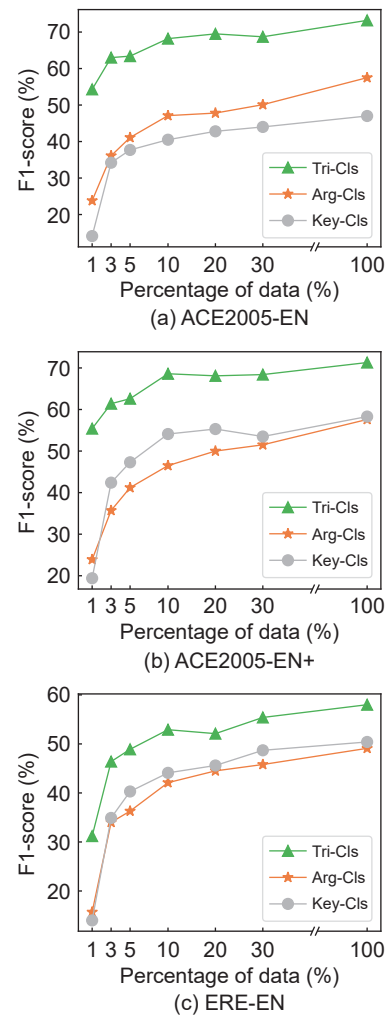


Fig. 4 Comparison of Tri-Cls, Arg-Cls, and Key-Cls in different proportion training sets.

the absence of pronouns in the dataset. Our EKE labels event keywords in the original context, improving the understanding of context and co-reference semantics. As a result, our model achieves better performance in Key-CIs on the ACE2005-EN+ and ERE-EN.

4.5 Case study

We present four cases in Fig. 5 to demonstrate the ability of KeyEE and the impact of different EKE results. In Case A, the baseline incorrectly assigns “people” as the “Agent” for the Life:Die event it extracts. With the aid of the auxiliary EKE sub-prompt, KeyEE accurately extracts the Life:Die event keywords in Case B, avoiding redundant or incorrect arguments. The last two cases show the impact of incorrect EKE results on EE. In Case C, our model overlooks the “Crichton” argument in the Life:Die event during keyword extraction. Despite this, the auxiliary sub-

prompt of EKE implicitly enhances EE, enabling our model to extract the complete event correctly. Conversely, in Case D, EKE incorrectly identifies “vote” as the Personnel:Elect event keyword, leading our model to erroneously infer a Personnel:Elect event triggered by “vote” in the context.

4.6 Impact of different sub-prompt strategies

In order to obtain the best auxiliary sub-prompt strategies, we have made many different attempts. Table 6 shows the differences between them.

4.6.1 Sub-prompt strategies

- Baseline is the same as Section 3.2, which models EE as a template-based generation problem.
- NER-Key models EKE as an independent Named Entity Recognition (NER) task, extracting both triggers and arguments dynamically according to the context and event type. Specifically, similar to Section 3.3, we

Comparison of KeyEE and baseline	Case A: Police have arrested four people in connection with the killings. Baseline: <Justice:Arrest-Jail, arrested>: (Agent, Police), (Person, people) <Life:Die, killings>: <u>(Agent, people)</u> KeyEE: <Justice:Arrest-Jail, arrested>: (Agent, Police), (Person, people) <Life:Die, killings>
	Case B: Police have arrested four people in connection with the killings. Event Type: Life:Die EKE result: Police have arrested four people in connection with the <Keyword> killings </Keyword>. EE result: <Life:Die, killings>
Impact of different EKE results	Case C: Anne-Marie sued Crichton for divorce in September after their marriage broke down. Event Type: Life:Die EKE result: <Keyword> Anne-Marie </Keyword> <Keyword> sued </Keyword> <u>Crichton</u> for divorce in September after their marriage broke down. EE result: <Justice:Sue, sued>, (Plaintiff, Anne-Marie), (Defendant, Crichton)
	Case D: We are proceeding with all the plans for the vote. Event Type: <u>Personnel:Elect</u> EKE result: We are proceeding with all the plans for the <Keyword> vote </Keyword>. EE result: <u><Personnel:Elect, vote></u>

Fig. 5 Four event extraction cases where three with event keywords extraction results. In EE results, event type and corresponding trigger are paired with “< >”, argument role type and argument are paired with “()”, different events are separated by “|”, where the wrong result is highlighted by underlined and red words. In EKE results, the right results are in green and the wrong results are in red. In the last case, the red event type indicates there is no ‘Personnel:Elect’ event in the context.

Table 6 Comparison of different event keywords format. <Trigger> is the placeholder of triggers and <Keyword> is the placeholder of keywords, i.e., both triggers and arguments.

Model	Manner	Train	Format
Baseline	Static	–	Similar triggers such as <Trigger>
NER-Key	Dynamic	Pipeline	Event keywords such as <Keyword>
MPL-Context	–	MPL	Original context
MPL-NaturalKey	Dynamic	MPL	Event keywords such as <Keyword>
MPL-LabelKey	Dynamic	MPL	Context labeled with <Keyword> and </Keyword>

construct the input by concatenating the context and the extract instruction. Then we train another BERT-CRF model to label the inputs with Begin-Inside-Outside (BIO) tags. After that, we replace the placeholder $\langle \text{Keyword} \rangle$ in “Event keywords such as $\langle \text{Keyword} \rangle$ ” with the keywords extracted by the NER model, which is called NER-Key. Finally, we run EE as described in Section 3.2, while the static keywords are replaced by NER-Key. In the above process, EE depends on the NER result, which is a pipeline manner.

- MPL-Context bears a resemblance to the conventional language modeling task. In order to delve into whether it is the fine-tuning of the language model or the event keywords that contribute to the enhancement of model performance, we establish this experiment using the keywords sub-prompt to guide the model outputting the original context.

- MPL-NaturalKey models EE and EKE in a multi-prompt learning strategy and enhances EE implicitly. Instead of labelling the context with special tokens, MPL-NaturalKey outputs event keywords in a natural language format (NaturalKey) like “Event keywords are $\langle \text{event_keywords} \rangle$ ”, where $\langle \text{event_keywords} \rangle$ is golden label placeholder. Except for the keywords extraction format, the other training details are the same with Section 3.3.

- MPL-LabelKey is the strategy described in Section 3.3, which is used in KeyEE in practice. MPL-

LabelKey labels the event keywords (LabelKey) in the context with a pair of special tokens and enhances EE implicitly with the MPL strategy.

We evaluate different keywords-enhanced strategies on three datasets with both high-resource and low-resource settings, and the results are shown in Table 7. For low-resource settings, we follow DEGREE^[17] using the 10% training split in the experiments and report the average performance over 3 runs.

4.6.2 Experimental result

Overall, MPL-LabelKey proves to be the most effective strategy, demonstrating notable performance in both Tri-Cls and Arg-Cls. Conversely, NER-Key is the worst and does not yield any improvement in all experiments. MPL-NaturalKey is similar to MPL-LabelKey and achieves improvement in most experiments, but the improvement is limited compared to MPL-LabelKey. In most cases, MPL-Context exhibits a decline in performance. This signifies that the improvement is not attributed to the fine-tuning of the language model, validating the effectiveness and significance of our EKE setting. Our MPL-LabelKey successfully directs the model’s attention to event keywords, leading to a notable improvement in the event extraction performance.

NER-Key works in a pipeline manner and cannot avoid error propagation. Compared with static event keywords, the wrong keywords are harmful. They not

Table 7 Different prompt strategies results in both high-resource scenarios and low-resource scenarios. The baseline is the result of running the default configuration based on DEGREE^[17] open source code. Bold indicates the highest scores. The numbers in parentheses are the results compared to the baseline, where “+” indicates improvement and “-” indicates a decrease.

Model	High-resource					
	ACE2005-EN		ACE2005-EN+		ERE-EN	
	Tri-Cls	Arg-Cls	Tri-Cls	Arg-Cls	Tri-Cls	Arg-Cls
Baseline	73.9	55.7	69.5	54.9	55.0	47.7
Baseline with NER-Key	71.0 (−2.9)	50.1 (−5.6)	71.6 (+2.1)	50.9 (−4.0)	55.9 (+0.9)	46.7 (−1.0)
Baseline with MPL-Context	69.8 (−4.1)	52.3 (−3.4)	73.0 (+3.5)	55.7 (+0.8)	54.6 (−0.4)	46.6 (−1.1)
Baseline with MPL-NaturalKey	73.6 (−0.3)	54.2 (−1.5)	73.3 (+3.8)	56.6 (+1.7)	57.8 (+2.8)	48.7 (+1.0)
Baseline with MPL-LabelKey	73.2 (−0.7)	57.5 (+1.9)	71.3 (+1.8)	57.6 (+2.7)	58.0 (+3.0)	49.1 (+1.4)
Model	Low-resource (10% of training set)					
	ACE2005-EN		ACE2005-EN+		ERE-EN	
	Tri-Cls	Arg-Cls	Tri-Cls	Arg-Cls	Tri-Cls	Arg-Cls
Baseline	68.0	42.9	69.4	44.4	53.1	41.2
Baseline with NER-Key	59.9 (−8.1)	31.9 (−11.0)	59.7 (−9.7)	39.4 (−5.0)	43.6 (−9.5)	27.4 (−13.8)
Baseline with MPL-Context	66.6 (−1.4)	42.3 (−0.6)	61.3 (−8.1)	43.7 (−0.7)	53.4 (+0.3)	39.3 (−1.9)
Baseline with MPL-NaturalKey	67.8 (−0.2)	43.9 (+1.0)	68.5 (−0.9)	44.7 (+0.3)	54.8 (+1.7)	42.0 (+0.8)
Baseline with MPL-LabelKey	68.7 (+0.7)	45.7 (+2.8)	68.2 (−1.2)	48.0 (+3.6)	53.1 (+0.0)	42.0 (+0.8)

only fail to supply the model with sufficient knowledge, but also misguide it, directly impacting its performance and leading to subpar results. This problem is more serious in low-resource scenarios. It is more difficult for the named entity model to extract the right event keywords, resulting in a great performance degradation (Arg-Cls decreases 11.0% and 13.8% in 10% of the original set on ACE2005-EN and ERE-EN, respectively).

In comparison to NER-Key, both MPL-NaturalKey and MPL-LabelKey acquire event keyword knowledge implicitly. This means that even if they extract incorrect event keywords, it does not directly impact event extraction. The experimental results substantiate this claim, as both methods outperform NER-Key significantly. However, with the same multi-prompt strategy, MPL-LabelKey performs better than MPL-NaturalKey. First, MPL-LabelKey is more stable: in high-resource scenarios, MPL-NaturalKey experiences a performance decrease on ACE2005-EN but gains an abnormal Tri-Cls improvement on ACE2005-EN+. Second, MPL-LabelKey achieves a better Arg-Cls improvement in all experiments. There are three main reasons for these phenomena: (1) MPL-LabelKey outputs the origin sentence labelled with special tokens, which helps the model understand the input context better. (2) Compared to outputting event keywords directly, labeling event keywords in the original context provides position information of the event keywords for the model. (3) Labeling event keywords with special tokens can avoid ambiguity when the context contains multiple identical text spans.

MPL-LabelKey also harbors some limitations. Firstly, our method needs to manually construct event templates for each event type, and can only extract one type at a time. For each input sentence, we need concatenate it with templates of all event types in turn, thus the extraction efficiency of a single sentence is inversely proportional to the number of event types. Secondly, the introduction of a set of auxiliary prompts necessitates two forward computations of approximately equal magnitude during training. This results in a more severe computation increase when the context is long. Thirdly, as stated in Section 4.2, the performance of EKE gradually declines with an increase in context length, leading to a limited improvement in event extraction performance when the context is long.

5 Conclusion

In this paper, we propose KeyEE, a simple yet effective MPL strategy for low-resource generation-based event extraction, which implicitly provides additional event keywords knowledge for the model to enhance event extraction. We conduct both low-resource and high-resource experiments on ACE2005 and ERE-EN. Experimental results show that our model outperforms or achieves at least competitive results with the previous strong competitors. In the future, we will delve deeper into the role of event keywords in event extraction. Furthermore, we believe that expanding event keyword extraction to document-level event extraction and accomplishing event keyword extraction in a more computationally efficient manner are two avenues worth exploring.

Acknowledgment

This work was supported by the National Key Research and Development Program of China (No. 2021YFF1201200), the Science and Technology Major Project of Changsha (No. kh2202004), and the Natural Science Foundation of China (No. 62006251). This work was carried out in part using computing resources at the High-Performance Computing Center of Central South University, China.

References

- [1] J. Yao, B. Zhang, C. Li, D. Hong, and J. Chanussot, Extended vision transformer (ExViT) for land use and land cover classification: A multimodal deep learning framework, *IEEE Trans. Geosci. Remote. Sens.*, vol. 61, p. 5514415, 2023.
- [2] C. Li, B. Zhang, D. Hong, J. Yao, and J. Chanussot, LRR-net: An interpretable deep unfolding network for hyperspectral anomaly detection, *IEEE Trans. Geosci. Remote Sensing*, vol. 61, pp. 1–12, 2023.
- [3] X. Wu, D. Hong, and J. Chanussot, UIU-net: U-net in U-net for infrared small object detection, *IEEE Trans. Image Process.*, vol. 32, pp. 364–376, 1809.
- [4] X. Wu, D. Hong, and J. Chanussot, Convolutional neural networks for multimodal remote sensing data classification, *IEEE Trans. Geosci. Remote. Sens.*, vol. 60, p. 5517010, 2021.
- [5] D. Hong, N. Yokoya, J. Chanussot, and X. X. Zhu, An augmented linear mixing model to address spectral variability for hyperspectral unmixing, *IEEE Trans. Image Process.*, vol. 28, no. 4, pp. 1923–1938, 2019.
- [6] X. Du and C. Cardie, Event extraction by answering (almost) natural questions, in *Proc. 2020 Conf. Empirical Methods in Natural Language Processing (EMNLP)*, <https://doi.org/10.48550/arXiv.2004.13625>, 2023.

- [7] F. Li, W. Peng, Y. Chen, Q. Wang, L. Pan, Y. Lyu, and Y. Zhu, Event extraction as multi-turn question answering, in *Proc. Findings of the Association for Computational Linguistics (EMNLP 2020)*, Virtual Event, pp. 829–838.
- [8] S. Wang, M. Yu, S. Chang, L. Sun, and L. Huang, Query and extract: Refining event extraction as type-oriented binary decoding, in *Proc. Findings of the Association for Computational Linguistics (ACL 2022)*, Dublin, Ireland, 2022, pp. 169–182.
- [9] S. Yang, D. Feng, L. Qiao, Z. Kan, and D. Li, Exploring pre-trained language models for event extraction and generation, in *Proc. 57th Annual Meeting of the Association for Computational Linguistics*, Florence, Italy, 2019, pp. 5284–5294.
- [10] Y. Lin, H. Ji, F. Huang, and L. Wu, A joint neural model for information extraction with global features, in *Proc. 58th Annual Meeting of the Association for Computational Linguistics*, Virtual Event, 2020, pp. 7999–8009.
- [11] T. M. Nguyen and T. H. Nguyen, One for all: Neural joint modeling of entities and events, *Proc. AAAI Conf. Artif. Intell.*, vol. 33, no. 1, pp. 6851–6858, 2019.
- [12] A. Ramponi, R. van der Goot, R. Lombardo, and B. Plank, Biomedical event extraction as sequence labeling, in *Proc. 2020 Conf. Empirical Methods in Natural Language Processing (EMNLP)*, Virtual Event, 2020, pp. 5357–5367.
- [13] D. Wadden, U. Wennberg, Y. Luan, and H. Hajishirzi, Entity, relation, and event extraction with contextualized span representations, in *Proc. 2019 Conf. Empirical Methods in Natural Language Processing and the 9th Int. Joint Conf. Natural Language Processing (EMNLP-IJCNLP)*, Hong Kong, China, 2019, pp. 5784–5789.
- [14] Z. Zhou, X. Dong, R. Meng, M. Wang, H. Yan, K. Yu, and K.-K. R. Choo, Generative steganography via auto-generation of semantic object contours, *IEEE Trans. Inform. Forensic Secur.*, vol. 18, pp. 2751–2765, 2023.
- [15] Y. Cao, Z. Zhou, C. Chakraborty, M. Wang, Q. M. Jonathan Wu, X. Sun, and K. Yu, Generative steganography based on long readable text generation, *IEEE Trans. Comput. Soc. Syst.*, doi:10.1109/TCSS.2022.3174013.
- [16] Z. Zhou, X. Dong, Z. Li, K. Yu, C. Ding, and Y. Yang, Spatio-temporal feature encoding for traffic accident detection in VANET environment, *IEEE Trans. Transp. Syst.*, vol. 23, no. 10, pp. 19772–19781, 2022.
- [17] I.-H. Hsu, K.-H. Huang, E. Boschee, S. Miller, P. Natarajan, K.-W. Chang, and N. Peng, DEGREE: A data-efficient generation-based event extraction model, in *Proc. 2022 Conf. North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, Seattle, WA, USA, 2022, pp. 1890–1908.
- [18] X. Liu, H. Huang, G. Shi, and B. Wang, Dynamic prefix-tuning for generative template-based event extraction, in *Proc. 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Dublin, Ireland, 2022, pp. 5216–5228.
- [19] G. Paolini, B. Athiwaratkun, J. Krone, J. Ma, A. Achille, R. Anubhai, C. N. D. Santos, B. Xiang, and S. Soatto, Structured prediction as translation between augmented natural languages, in *Proc. International Conference on Learning Representations*, Virtual Event, 2021, pp. 1–26.
- [20] Y. Lu, Q. Liu, D. Dai, X. Xiao, H. Lin, X. Han, L. Sun, and H. Wu, Unified structure generation for universal information extraction, in *Proc. 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Dublin, Ireland, 2022, pp. 5755–5772.
- [21] G. Doddington, A. Mitchell, M. Przybocci, L. Ramshaw, S. Strassel, and R. Weischedel, The automatic content extraction (ACE) program—tasks, data, and evaluation, in *Proceedings of the Fourth International Conference on Language Resources and Evaluation*, Lisbon, Portugal, 2004, pp. 1–4.
- [22] Z. Song, A. Bies, S. Strassel, T. Riese, J. Mott, J. Ellis, J. Wright, S. Kulick, N. Ryant, and X. Ma, From light to rich ERE: Annotation of entities, relations, and events, in *Proc. the 3rd Workshop on EVENTS: Definition, Detection, Coreference, and Representation*, Denver, CO, USA, 2015, pp. 89–98.
- [23] H. Cao, J. Li, F. Su, F. Li, H. Fei, S. Wu, B. Li, L. Zhao, and D. Ji, OneEE: A one-stage framework for fast overlapping and nested event extraction, in *Proceedings of the 29th International Conference on Computational Linguistics*, Gyeongju, Republic of Korea, 2022, pp. 1953–1964.
- [24] Y. Lu, H. Lin, J. Xu, X. Han, J. Tang, A. Li, L. Sun, M. Liao, and S. Chen, Text2Event: Controllable sequence-to-structure generation for end-to-end event extraction, in *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing*, Virtual Event, 2021, pp. 2795–2806.
- [25] S. Li, H. Ji, and J. Han, Document-level event argument extraction by conditional generation, in *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, Virtual Event, 2021, pp. 894–908.
- [26] X. L. Li and P. Liang, Prefix-tuning: Optimizing continuous prompts for generation, in *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing*, Virtual Event, 2021, pp. 4582–4597.
- [27] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, Attention is all you need, arXiv preprint arXiv: 1706.03762, 2017.
- [28] P. Liu, W. Yuan, J. Fu, Z. Jiang, H. Hayashi, and G. Neubig, Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing, *ACM Comput. Surv.*, vol. 55, no. 9, pp. 1–35, 2023.
- [29] Z. Jiang, F. F. Xu, J. Araki, and G. Neubig, How can we know what language models know, *Trans. Assoc. Comput. Linguist.*, vol. 8, pp. 423–438, 2020.
- [30] T. Schick and H. Schütze, Exploiting cloze-questions for

- few-shot text classification and natural language inference, in *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, Virtual Event, 2021, pp. 255–269.
- [31] T. Gao, A. Fisch, and D. Chen, Making pre-trained language models better few-shot learners, in *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing*, Virtual Event, 2021, pp. 3816–3830.
- [32] S. Kumar and P. Talukdar, Reordering examples helps during priming-based few-shot learning, in *Proc. Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, Virtual Event, 2021, pp. 4507–4518.
- [33] X. Han, W. Zhao, N. Ding, Z. Liu, and M. Sun, PTR: Prompt tuning with rules for text classification, *AI Open*, vol. 3, pp. 182–192, 2022.
- [34] L. Cui, Y. Wu, J. Liu, S. Yang, and Y. Zhang, Template-based named entity recognition using BART, in *Proc. Findings of the Association for Computational Linguistics: ACLIJCNLP 2021*, Virtual Event, 2021, pp. 1835–1845.
- [35] M. Lewis, Y. Liu, N. Goyal, M. Ghazvininejad, A. Mohamed, O. Levy, V. Stoyanov, and L. Zettlemoyer, BART: denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension, in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, Virtual Event, 2020, pp. 7871–7880.
- [36] M. Matena, Y. Zhou, W. Li, and P. J. Liu, Exploring the limits of transfer learning with a unified text-to-text transformer, *Journal of Machine Learning Research*, vol. 21, no. 1, pp. 140:1–140:67, 2020.
- [37] E. Cohen and C. Beck, Empirical analysis of beam search performance degradation in neural sequence models, in *Proceedings of the 36th International Conference on Machine Learning*, Long Beach, CA, USA, 2019, pp. 1290–1299.
- [38] M. Zhang, Y. Su, Z. Meng, Z. Fu, and N. Collier, COFFEE: A contrastive oracle-free framework for event extraction, arXiv preprint arXiv: 2303.14452, 2023.
- [39] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, et al., Transformers: State-of-the-art natural language processing, in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, Virtual Event, 2020, pp. 38–45.
- [40] I. Loshchilov and F. Hutter, Decoupled weight decay regularization, in *Proc. 7th International Conference on Learning Representations*, New Orleans, LA, USA, 2019, pp. 1–18.



Junwen Duan received the BEng degree in information security and the PhD degree in computer science from Harbin Institute of Technology, China in 2013 and 2020, respectively. He currently is a lecturer at School of Computer Science and Engineering, Central South University, China. His main research interests include

natural language processing and text mining.



Jianxin Wang received the BEng and MEng degrees in computer engineering from Central South University, China in 1992 and 1996, respectively, and the PhD degree in computer science from Central South University, China in 2001. He is the chair and a professor at School of Computer Science and Engineering,

Central South University, China. His current research interests include algorithm analysis and optimization, parameterized algorithm, bioinformatics, and computer network. He is a senior member of the IEEE.



Xincheng Liao received the BEng degree in computer science from Central South University, China in 2022. He currently is a master student at School of Computer Science and Engineering, Central South University, China. His research interests include natural language processing and information extraction.



Ying An received the BEng degree in automatic control and the PhD degree in computer science from Central South University, China in 2009 and 2013, respectively. He is an associate professor at Institute of Big Data, Central South University, China. His research interests include big data analytics and applications, medical artificial intelligence, and machine learning.