World Wide Web (2023) 26:3983–3999
https://doi.org/10.1007/s11280-023-01216-5

KC-GEE: knowledge-based conditioning for generative event extraction

Tongtong Wu[1,2] · Fatemeh Shiri[2] · Jingqi Kang[1] · Guilin Qi[1] · Gholamreza Haffari[2] · Yuan-Fang Li[2]

Received: 21 October 2022 / Revised: 4 September 2023 / Accepted: 1 October 2023 /
Published online: 25 October 2023

Abstract
Event extraction is an important, but challenging task. Many existing techniques decompose it into event and argument detection/classification subtasks, which are complex structured prediction problems. Generation-based extraction techniques lessen the complexity of the problem formulation and are able to leverage the reasoning capabilities of large pretrained language models. However, they still suffer from poor zero-shot generalizability and are ineffective in handling long contexts such as documents. We propose a generative event extraction model, KC-GEE, that addresses these limitations. A key contribution of KC-GEE is a novel knowledge-based conditioning technique that injects the schema of candidate event types as the prefix into each layer of an encoder-decoder language model. This enables effective zero-shot learning and improves supervised learning. Our experiments on two benchmark datasets demonstrate the strong performance of our KC-GEE model. It achieves particularly strong results in the challenging document-level extraction task and in the zero-shot learning setting, outperforming state-of-the-art models by up to 5.4 absolute F1 points.

Keywords Event extraction · Information extraction · Zero-shot learning · Document-level event extraction

1 Introduction

Event extraction [1] aims at extracting structured event records from unstructured text. For example, as shown in Figure 1, the goal of event extraction is to map the document "Two homemade pressure-cooker bombs are detonated remotely by the Tsarnaevs near the finish line of the Boston Marathon, killing three and injuring some 260 others. Seventeen people lost limbs." to four predefined event types (highlighted with celeste), such as <*event type*: Attack, *trigger word*: detonated, *role:Attacker*: Tsarnaevs, . . . , *role:ExplosiveDevice*: bombs, *role:Place*: Boston Marathon>, as well as other events that are triggered by words *killing* and *injuring*.

✉ Yuan-Fang Li
yuanfang.li@monash.edu

Extended author information available on the last page of the article

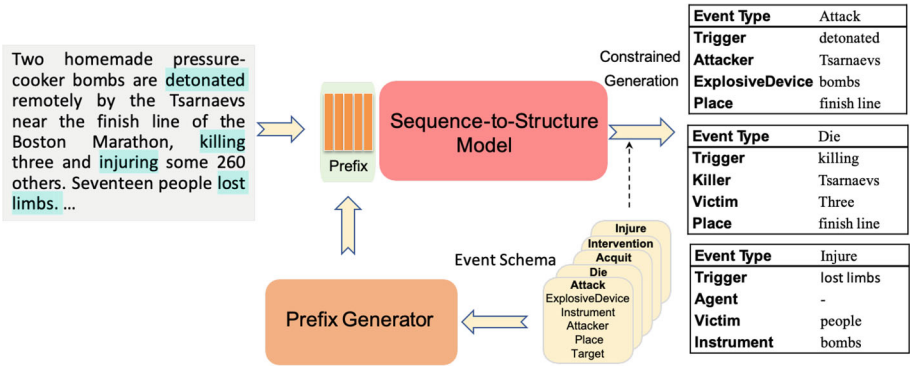

Figure 1 The event extraction task. In each event schema, we delineate the event type along with its associated roles. For instance, within the "Attack" event schema, roles such as "Attacker," "ExplosiveDevice," and "Place" are encompassed

Event extraction is challenging due to the diversity of natural language expressions and the complexity of event structures. These challenges are amplified in document-level event extraction where the text is a full document and typically contains more events. Currently, most event extraction methods employ a decomposition-based approach [2], which involves breaking down the structured prediction problem of a complex event into classifications of substructures like trigger detection, entity recognition, and argument classification. Many of these methods tackle the subproblems separately, which necessitates additional annotations for each stage [3].

Natural language generation techniques have been successfully applied to a number of NLP tasks [4–6]. These techniques have inspired the use of controlled event generation to tackle event extraction. These approaches use manually designed templates to wrap input sentences and train a model for cloze-style filling. The study by [7] proposes generating linearised event records via a pretrained encoder-decoder architecture combined with a constrained decoding mechanism that alleviates the complexity associated with template combination when extracting multiple events. The advantage of the extraction-as-generation approach is the removal of the need for fine-grained token-level annotations, which are typically used in previous event extraction approaches [8], thus enjoying greater feasibility.

Although good generalizability has been achieved for other tasks, we have observed a significant decrease in performance when it comes to generation-based event extraction over documents or unseen event types. Structured prediction tasks, such as event extraction, often rely on an external schema to format the output, whereas natural language generation tasks do not. To bridge this gap, we introduce a novel technique called *knowledge-based conditioning*. This approach involves injecting event type information as *prefixes* on different layers of the underlying pretrained language model. By incorporating this information, we aim to improve the performance of event extraction tasks. Additionally, to address the challenge of adapting to new scenarios, we consider event extraction from the perspective of zero-shot learning [9, 10]. Our model, KC-GEE, is capable of document-level event extraction and is generalizable to the zero-shot setting.

Our main contributions are as follows.

- We propose a novel knowledge-based conditioning technique that injects event type information into the model, enabling zero-shot learning capability.

- We carefully design a prefix-based injection mechanism that incorporates cross-attention to improve document-level event extraction.
- We conducted extensive experiments on two benchmark datasets, in both fully supervised and zero-shot settings. Our evaluation consistently shows strong performance across all settings. In particular, our model achieves substantial superiority in the challenging settings of document-level event extraction and zero-shot transfer, outperforming state-of-the-art models by up to 5.4 absolute F1 points.

2 Related work

Document-level event extraction Event extraction is a task that extracts structured event records from unstructured text [5]. Many approaches have been proposed for sentence-level event extraction [11, 12], ranging from hand-designed features [13] and neural-learned features [14, 15]. Yet, many real-world applications require document-level event extraction [14–18], in which the information of an event may be mentioned in multiple sentences [19, 20]. Moreover, most work adopt decomposition strategies in event extraction [2], which employ trigger detection [13], entity recognition [21, 22], and argument classification [23]. These decomposition strategies have shown high performance while introducing more detailed annotation to model training [5, 7].

Zero-shot event extraction Several previous supervised event extraction methods have relied on features derived from manual annotations, limiting their applicability to new event types without additional annotation effort [9, 24, 25]. These methods often struggle to effectively generalize to new label taxonomies and domains. In contrast, [26] proposes a zero-shot event extraction approach. They first utilize existing tools, such as SRL (Semantic Role Labeling), to identify events and subsequently map them to a predefined taxonomy of event types without the need for specific training data. Lyu et al. [27] explores the possibility of zero-shot event extraction by formulating it as a series of Textual Entailment (TE) and/or Question Answering (QA) queries. For instance, they utilize pretrained TE/QA models to establish a direct transfer of knowledge, using statements like "A city was attacked" which entails "There is an attack." In this paper, we propose a novel approach for zero-shot event extraction by jointly training a prefix generator for event schemas. Our method is designed to be parameter-efficient and lightweight, allowing for effective event extraction even in scenarios with limited or no training data.

Generative event extraction . Generative event extraction has emerged as a promising approach for automatically extracting event information from text by employing generative models. Motivated by the achievements of pretrained language models and the associated natural language generation-based approach in diverse NLP tasks [4, 28–32], some researchers have approached event extraction as controlled event generation. As shown in Figure 2, [5, 6] are end-to-end conditional generation methods with manually designed discrete prompts for each event type, requiring more human effort to find the optimal prompt. To remove the complexity of template combination in extracting multiple events, [7] proposed a method for generating the event records directly using a pretrained encoder-decoder architecture and a constrained decoding mechanism. This extraction-as-generation approach does not require fine-grained token-level annotations, which are typically needed by previous event extraction methods. Liu et al. [33] proposes a generative template-based event extraction method that uses dynamic prefixes and integrates context information with type-specific prefixes to learn

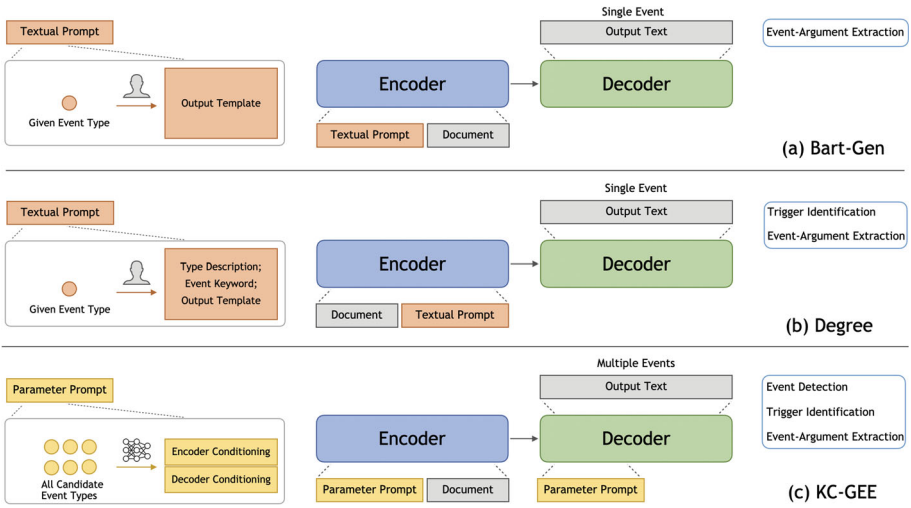

Figure 2 The comparison among KC-GEE and the other prompt-based generative methods, e.g., BART-Gen [5] and Gegree [6]

a context-specific prefix for each context. However, this method does not consider zero-shot extraction or document-level extraction, which we consider in our paper.

3 Generation-based event extraction

Problem definition We denote $\mathcal{E}$ and $\mathcal{R}$ as the set of predefined event types and role categories, respectively. An input sequence $\boldsymbol{x} := \{x_1, \ldots, x_{|\boldsymbol{x}|}\}$ comprises tokens x_i, where $|\ \boldsymbol{x}\ |$ denotes the sequence length. Given an input document, an event extraction model aims to extract one or more structured events, where each event is specified by (i) the event type $e \in \mathcal{E}$ along with the trigger word t from the document, and (ii) the roles $\mathcal{R}_e \subseteq \mathcal{R}$ along with their corresponding arguments from the document.

Event extraction as generation Given $\mathcal{E}$ and $\mathcal{R}$ in the predefined event schema, generation-based event extraction models generate a structured sequence based on an input document that is constrained by the schema [7].

The generated sequence is a linearised representation of events mentioned in the document. Specifically, given a document with token sequence $\boldsymbol{x}$ as input, a generation-based extraction model, such as KC-GEE, outputs the linearised events representations $\boldsymbol{y} = \langle y_1, y_2, \ldots, y_{|\boldsymbol{y}|}\rangle$, where each event y_i is denoted by $\langle e_i, t_i, \langle r_{i,1}, a_{i,1}\rangle, \ldots, \langle r_{i,|r|}, a_{i,|r|}\rangle\rangle$. The angled brackets $\langle\cdot\rangle$ are special tokens indicating the sequence structure. The $e \in \mathcal{E}$ and t are the event type and the trigger words (a subspan of the document $\boldsymbol{x}$); furthermore, $r_i \in \mathcal{R}$ and a_i denote roles and arguments (subspans of the document $\boldsymbol{x}$).

Architecture Our KC-GEE model adopts a Transformer-based encoder-decoder architecture for event structure generation. KC-GEE outputs the sequentialized event representation $\boldsymbol{y}$ for an input document $\boldsymbol{x}$. First, it computes the hidden representation $\mathbf{H}_{\boldsymbol{x}} = (\boldsymbol{h}_1, \boldsymbol{h}_2, \ldots, \boldsymbol{h}_{|\boldsymbol{x}|}) \in \mathbb{R}^{|\boldsymbol{x}|\times d}$ for each token in the document via a multi-layer Transformer encoder:

$$\mathbf{H}_{\boldsymbol{x}} = \text{Encoder}(\boldsymbol{x}), \tag{1}$$

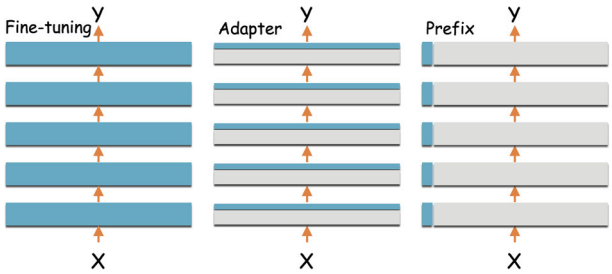

Figure 3 A high-level illustration of three candidates knowledge-based conditioning injection paradigm for encoder-decoder models: fine-tuning, adapter-tuning, and prefix-tuning. For each tuning type, the gray blocks indicate the frozen parameters in a pretrained model, and the blue blocks indicate the trainable parameters in a pretrained model

where each layer of Encoder$(\cdot)$ is a Transformer block [34] with the multi-head self-attention mechanism.

Given the encoding $\mathbf{H}_{\boldsymbol{x}}$, the decoder generates each token sequentially to produce the sequence of events. At step t, the Transformer-based decoder generates the token y_t and hidden state $\boldsymbol{h}_t$ as:

$$y_t, \boldsymbol{h}_t = \text{Decoder}(y_{t-1}; \mathbf{H}_{\boldsymbol{y}_{<t}}, \mathbf{H}_{\boldsymbol{x}}), \tag{2}$$

where each layer of Decoder$(\cdot)$ is a Transformer block, with both the self-attention to past hidden states $\mathbf{H}_{\boldsymbol{y}_{<t}} \in \mathbb{R}^{(t-1)\times d}$ during decoding and the cross-attention to the encoding $\mathbf{H}_{\boldsymbol{x}}$. The conditional probability of the output sequence $p(\boldsymbol{y} \mid \boldsymbol{x})$ is then,

$$p_\theta(\boldsymbol{y} \mid \boldsymbol{x}) = \prod_{t=1}^{|y|} p_\theta(y_t \mid \boldsymbol{y}_{<t}, \boldsymbol{x}), \tag{3}$$

where θ denotes the parameters of the Transformer-based encoder-decoder model.

4 Knowledge-based conditioning in event generation

This paper investigates the best way to leverage pretrained language models (PLMs) as the backbone encoder-decoder model for event extraction.[1] Using PLMs is now standard practice in NLP, as they lead to strong performance and generalisation.

Given a labeled training dataset $\mathcal{D}$, we investigate the best way to specialise the PLM for the event extraction task via prefix-tuning [35]. In this section, we show how to effectively *condition* the generation process on the event extraction task as well as the given document.

One may specialise the underlying PLM to the event extraction task through other methods, such as fine-tuning the PLM parameters or injecting adapters to the encoder and/or decoder of the PLM (see Figure 3). Our experiments show that prefix-tuning is more effective than those methods.

Our desiderata for prefix-conditioning of a PLM for event extraction are as follows: It should enable the model to be aware of (i) the candidate event schemas in the task, (ii) the specific input document, and (iii) flexible schema modifications that may occur after the

[1] In our experiments, we make use of T5 [4], but our methods are applicable to other large pretrained encoder-decoder models as well.

model is trained in real-world settings. In what follows, we explain how we achieve these desiderata by producing prefixes for the encoder and the decoder based on the events of the task and the input document. Please refer to Figure 4 for an overview of the framework.

4.1 Encoder conditioning

We condition the encoder on the event types of the underlying event extraction task. Given the event types $\boldsymbol{e} = \{e_1, e_2, \ldots, e_{|\boldsymbol{e}|}\} \subseteq \mathcal{E}$ for a task, we use the encoder to get the encoding representation for the event types $\mathbf{H}_{\boldsymbol{e}} \in \mathbb{R}^{|\boldsymbol{e}| \times d}$. We then combine these events representations through a function $f_{enc} : \mathbb{R}^{|\boldsymbol{k}| \times d} \mapsto \mathbb{R}^{d'}$ to create the events conditioning context, i.e.

$$\mathbf{H}_{\boldsymbol{e}} = \text{Encoder}(\boldsymbol{e}); \quad \boldsymbol{h}_{\boldsymbol{e},enc} = f_{enc}(\mathbf{H}_{\boldsymbol{e}}) \tag{4}$$

Since we assume each event type is equally probable *a priori*, we use the pooling average operator as f_{enc}. The vector $\boldsymbol{h}_{\boldsymbol{e},enc}$ is used by a prefix generation network g_{enc} to produce the prefix. As shown in Figure 4, by $\pm$ in $f_enc(.)$, we suggest that adding or removing an event type representation from knowledge-based conditioning is flexible.

4.2 Decoder conditioning

It is expected that the representation of instances could help the downstream generation in the decoder. Hence we use the representation of both the task and the input document to create a prefix for the decoder.

Specifically, let $\mathbf{H}_{\boldsymbol{x}}$ denote the representation of the tokens of the input document $\boldsymbol{x}$. We combine the document representation $\mathbf{H}_{\boldsymbol{x}}$ and the task representation $\mathbf{H}_{\boldsymbol{e}}$ through the function $f_{dec} : \mathbb{R}^{|\boldsymbol{e}| \times d} \times \mathbb{R}^{|\boldsymbol{e}| \times d} \mapsto \mathbb{R}^{d'} \times \mathbb{R}^{d'}$ as follows,

$$\boldsymbol{h}_{\boldsymbol{e},dec}, \boldsymbol{h}_{\boldsymbol{x},dec} = f_{dec}(\mathbf{H}_{\boldsymbol{e}}, \mathbf{H}_{\boldsymbol{x}}) \tag{5}$$

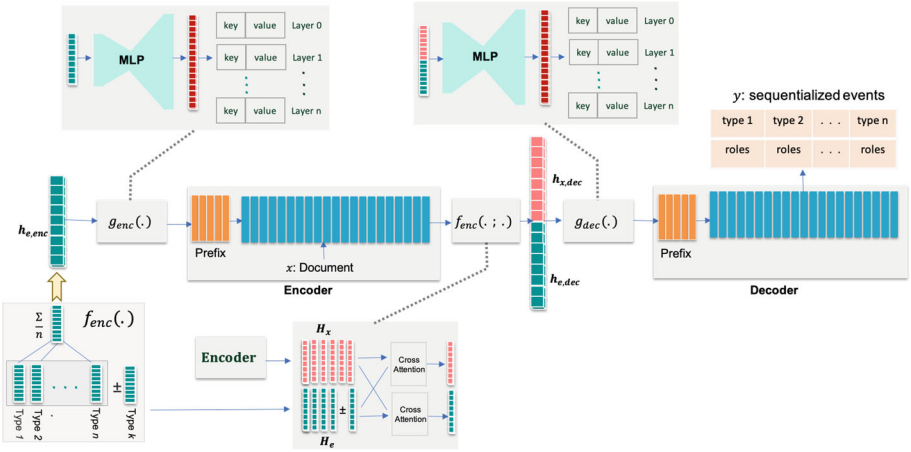

Figure 4 An illustration of our end-to-end framework KC-GEE, where the main architecture is a transformer-based encoder-decoder in the center. The lower blocks represent the conditioning construction modules for encoder and decoder, respectively. The upper blocks represent the conditioning injection modules for encoder and decoder, respectively

where f_{dec} is based on dot product-based cross-attention, and $\boldsymbol{h}_{\boldsymbol{e},dec} \in \mathbb{R}^{d'}$, $\boldsymbol{h}_{\boldsymbol{x},dec} \in \mathbb{R}^{d'}$ are the resulting fixed-dimensional vector summaries for decoder conditioning.

4.3 Prefix generation

We create the encoder prefix $\mathbf{Z}_{enc}$ and decoder prefix $\mathbf{Z}_{dec}$ as follows,

$$\begin{aligned} \mathbf{Z}_{enc} &= g_{enc}(\boldsymbol{h}_{\boldsymbol{e},enc}) \\ \mathbf{Z}_{dec} &= g_{dec}([\boldsymbol{h}_{\boldsymbol{x},dec}; \boldsymbol{h}_{\boldsymbol{x},dec}]) \end{aligned} \tag{6}$$

where g_{enc} and g_{dec} are both mapping function $g\colon \mathbb{R}^{2\times d'} \mapsto \mathbb{R}^{k\times|\mathbf{H}_i|}$, where k is the length of injected prefix and $|\ \mathbf{H}_i\ |$ is the number of parameters of the ith injected prefix maintained in the Transformer architecture. With the injection of $\mathbf{Z}_{enc}$ and $\mathbf{Z}_{dec}$, the encoder and the decoder in (1) and (2) are modified as follows:

$$\mathbf{H}_{\boldsymbol{x}} = \mathrm{E}ncoder(\boldsymbol{x}; \mathbf{Z}_{enc}) \tag{7}$$

$$y_t, \boldsymbol{h}_t = \mathrm{Decoder}(y_{t-1}; \mathbf{H}_{\boldsymbol{y}_{<t}}, \mathbf{Z}_{dec}, \mathbf{H}_{\boldsymbol{x}}), \tag{8}$$

where $\mathbf{Z}_{enc}$ and $\mathbf{Z}_{dec}$ can be thought as pseudo-prefix tokens impacting the generation process [35].

4.4 Training and inference

We train the model by minimising the negative log-likelihood loss:

$$\theta^* = \arg\min_{\theta} \sum_{(\boldsymbol{x},\boldsymbol{y})\in\mathcal{D}} -\log p_\theta(\boldsymbol{y} \mid \boldsymbol{x}, \boldsymbol{e}) \tag{9}$$

where $\mathcal{D}$ is the training set, θ^* denotes the optimal parameters, $\boldsymbol{e} = \{e_1, e_2, \ldots, e_{|\boldsymbol{e}|}\} \subseteq \mathcal{E}$ denotes the event type for a task, $\boldsymbol{x}$ is the input document and $\boldsymbol{y}$ is the predicted event structure. As we formulate the event extraction problem as a sequence generation problem, the overall likelihood $p_\theta(\boldsymbol{y} \mid \boldsymbol{x}, \boldsymbol{e})$ is formulated as follows:

$$p_\theta(\boldsymbol{y} \mid \boldsymbol{x}, \boldsymbol{e}) = \prod_{t=1}^{|\boldsymbol{y}|} p_\theta(y_t \mid \boldsymbol{y}_{<t}, \boldsymbol{x}, \boldsymbol{e}). \tag{10}$$

where $p_\theta(\boldsymbol{y} \mid \boldsymbol{x}, \boldsymbol{e})$ is defined as the cumulative product of $p_\theta(y_t \mid \boldsymbol{y}_{<t}, \boldsymbol{x}, \boldsymbol{e})$, in which y_t is the t-th token in the output sequence $\boldsymbol{y}$.

For inference, we use constrained decoding [7].

5 Experiments

We compare our KC-GEE model with several recent strong models, evaluating in both supervised learning and zero-shot learning settings, as well as for the document-level extraction task. Our aim is to demonstrate the greater generalizability and effectiveness of our model in these challenging scenarios.

5.1 Evaluation setup

5.1.1 Datasets

We carry out experiments on two event extraction datasets: the sentence-level dataset Automatic Content Extraction 2005 (ACE05-EN) [11] and the document-level dataset: WIKIEVENTS [20]. The statistics for both are provided in Table 1. Note that we use the official dataset splits of the two datasets to ensure better reproducibility. It is worth noting that WIKIEVENTS presents significant challenges due to three factors. (1) ***Context length***: each instance in ACE05-EN contains only one sentence, whereas instances in WIKIEVENTS are documents. (2) ***Event density***: almost every instance in ACE05-EN contains only one event, whereas multiple events could be present in one instance in WIKIEVENTS. (3) ***Data scarcity***: the amount of training data in ACE05-EN is more than 77 times greater than that in WIKIEVENTS.

5.1.2 Evaluation metrics

We employ the same evaluation metrics used in previous work [7, 36] for both trigger extraction (Trig-C) and arguments extraction (Arg-C). These metrics include F1, precision, and recall.

As KC-GEE is a text generation model, we consider the input sequence one by one to find the matched utterance to reconstruct the offset of predicted trigger mentions. Additionally, in the case of argument mentions, we identify the trigger offset as the nearest matched utterance to the predicted trigger mention.

5.1.3 Baselines

We evaluate KC-GEE against three groups of baselines which use different levels of annotations of decreasing granularity: *Both token-level and entity-level annotation*, *Token-level annotation*, and *Parallel text-record annotation*. Some methods utilise token annotations, in which each token in an instance is annotated with event labels and golden entity annotation to facilitate event extraction. Joint3EE [36] is a multi-task model that jointly performs entity, trigger, and argument extraction by shared Bi-GRU hidden representations. DYGIE++ [8] is a BERT-based extraction framework that models text spans and captures within-sentence and cross-sentence context. GAIL [37] is an ELMo-based model that proposes a joint entity and event extraction framework based on generative adversarial imitation learning, which is an inverse reinforcement learning method. OneIE [36] introduces a classification-based information extraction system that employs global features and beam search to extract event structures. Some other methods use token-level annotation. For instance, TANL [3] is a sequence generation-based method that tackles event extraction in a trigger-argument

Table 1 Statistics of the event extraction datasets used in the paper, including the numbers of event types, argument types, the type of instances, events per instance, and the number of instances in different splits

Dataset	Event Type	Argument Type	Train	Dev	Test	Instance	Events per Instance
ACE05-EN	33	22	17,172	923	832	one sentence	Single/Multiple
WIKIEVENTS	33	59	206	20	20	Document	Multiple

pipeline. Multi-task TANL is the extended version of TANL that transfers structure knowledge from other tasks. BERT-QA [38] and MQAEE [39] consider event extraction as a sequence of extractive question-answering problems. Similar to Text2Event [7], we use *Parallel text-record annotation*, which only requires (instance, event) pairs without expensive, fine-grained token-level or entity-level annotations. As shown in an instance of such an annotation, ⟨"Evidence at a makeshift morgue points to mass executions by the Iraqi regime.", {Type: Execute, Trigger: executions, ...}⟩, parallel text-record annotation is the least demanding and therefore more practical annotation level. We compare our method with Text2Event [7], which introduces a sequence-to-structure generation model that addresses the missing event structure issue via constrained decoding. Given the BART-Gen [5] and Degree [6] both using BART-large as the backbone model while we are using T5-base, and both of the methods require the detected event type as prior, we list their results as another group distinguished from our method and Text2Event. Furthermore, we evaluate KC-GEE against zero-shot approaches on ACE05-EN [9, 10, 27].

5.1.4 Implementation details

We develop our KC-GEE method based on the T5-base pretrained language model, and train it for 50 epochs with a learning rate of 1e-4 and batch size of 8 for the supervised setting. For the zero-shot setting, we use a learning rate of 5e-5 and batch size of 16. To optimize KC-GEE, we employ label smoothing [41] and AdamW [42]. The prefix length is set to 20 for all experiments in Section 5.2.

5.2 Main results

We compare our KC-GEE model in two evaluation settings: fully supervised and zero-shot. For each setting, we organise the model evaluation by the characteristics of the datasets including sentence-level (ACE05-EN) and document-level (WIKIEVENTS).

5.2.1 Supervised setting

In this setting, each model is trained on the full training data of the respective dataset. Table 2 presents the sentence-level event extraction results on ACE05-EN. Note that except for the last block, performance numbers of all baselines are taken directly from Text2Event [7].

From the table, it can be observed that our KC-GEE model outperforms Text2Event in terms of F1 for both argument extraction and trigger extraction.

Sentence-level performance As discussed above, among all compared models, our KC-GEE model, together with Text2Event [7], is trained on parallel text-record annotations which represent the weakest form of supervision. In contrast, the other baseline models require token-level annotations and entity annotations, which are more fine-grained and expensive to collect. It is expected that models trained on more extensive data would perform better. The last column of the table also shows that the better-performing models use larger pretrained language models (PLMs), such as BERT-large. The larger capacity of these PLMs also contributes to model's performance.

Document-level performance Table 3 shows the performance of the baseline (Text2Event), our model KC-GEE, and its different variants for document-level event extraction on the

Table 2 Experiment results for the fully supervised event extraction on ACE05-EN. PLM represents the pretrained language model used by each model. We use *text-record* annotation, which only provides (instance, event) pairs without expensive, fine-grained token-level or entity-level annotations

Models	Annotation	Arg-C			Trig-C			PLM
		F1	P	R	F1	P	R	
Joint3EE [8]	Token+Entity	52.1	52.1	52.1	69.8	68	71.8	-
DYGIE++ [40]	Token+Entity	48.8	-	-	69.7	-	-	BERT-large
GAIL[37]	Token+Entity	52.4	61.6	45.7	72.0	74.8	69.4	ELMo
OneIE [36]	Token+Entity	56.8	-	-	74.7	-	-	BERT-large
BERT-QA [38]	Token	53.3	56.8	50.2	72.4	71.1	73.7	2 x BERT-base
MQAEE [39]	Token	53.4	-	-	71.7	-	-	3 x BERT-large
TANL [3]	Token	47.6	-	-	68.4	-	-	T5-base
Multi-Task TANL [3]	Token	48.5	-	-	68.5	-	-	T5-base
BART-GEN [5]	Text-record+Type	53.7	-	-	71.1	-	-	BART-large
Degree [6]	Text-record+Type	55.8	-	-	73.3	-	-	BART-large
Text2Event [7]	Text-record	49.8	46.7	53.4	69.2	67.5	71.2	T5-base
KC-GEE$_{\text{Fine tuning+Prefix}}$	Text-record	49.0	47.3	50.7	69.3	69.1	69.5	T5-base
KC-GEE$_{\text{Full}}$	Text-record	51.5	48.1	55.6	70.1	66.7	73.9	T5-base

WIKIEVENTS dataset. Please note that BART-Gen [39] and Degree [6] rely on explicit annotations of event type and assume the event-specific templates are given for document-level argument and trigger extraction, while both Text2Event and our model implicitly perform event detection and subsequent extraction. Furthermore, the remaining models listed in Table 2 are designed for sentence-level tasks and do not support this specific task. To ensure a fair comparison with other methods, we rigorously report our results under the identical setting as indicated in Table 3.

The majority of document-level baselines focus only on event argument extraction from WIKIEVENTS dataset, which did not handle event types and triggers [20, 43, 44]. Our model supports the joint extraction of both event triggers and arguments from the WIKIEVENTS dataset.

We can observe from the table that our full model achieves the best F1 values for both argument extraction (Arg-C) and trigger detection (Trig-C) on WIKIEVENTS. It is especially

Table 3 Results for supervised learning on the document-level event extraction dataset WIKIEVENTS

Models	Arg-C			Trig-C		
	F1	P	R	F1	P	R
Text2Event [7]	17.4	32.8	11.9	29.3	45.2	20.1
KC-GEE$_{\text{Full}}$	28.5 (+11)	41.2	21.8	38.7 (+9.4)	46.2	33.2
KC-GEE$_{\text{Adapter}}$	2.2	23.5	1.1	2.1	12.0	1.2
KC-GEE$_{\text{Fine tuning+Adapter}}$	16.1	25.1	11.9	28.3	37.3	22.8
KC-GEE$_{\text{Prefix}}$	4.5	8.5	3.1	9.7	9.5	10.0
KC-GEE$_{\text{Fine tuning+Prefix}}$	23.2	32.5	18.1	38.2	45.0	33.2

noteworthy that KC-GEE achieves significant performance advantages over Text2Event of +11.1 and +9.4 absolute F1 points for Arg-C and Trig-C, respectively.

The superiority of our model can be attributed to two design features. Firstly, our cross-attention mechanism filters event type tokens and argument tokens, allowing the model to handle long context better. Secondly, our knowledge-based conditioning mechanism injects event type information into the model, enabling it to learn more effectively with less data. A detailed analysis of the contributions of each model component are presented below.

5.2.2 Zero-shot setting

We evaluate KC-GEE's ability to generalize to unseen event types in the zero-shot setting for both sentence-level (ACE05-EN) and document-level (WIKIEVENTS) event extraction tasks. Specifically, for both datasets, we randomly split the instances into two subsets *Source* and *Target*. *Source* contains the annotations of 23 event types, while *Target* only retains the annotations of 10 instances for each of the 10 unseen event types. In this experiment, we first pretrained each model on the *Source* dataset, which is evaluated on the 10 new event types in the *Target* dataset without fine-tuning.

The results for both datasets are shown in Table 4. Once again, our full model significantly outperforms baselines. On ACE05-EN, it obtains F1 gains of 27.7 and 9.2 absolute points for Arg-C and Trig-C, respectively. On WIKIEVENTS, the F1 gains over Text2Event are 4.4 and 5.4 absolute points for Arg-C and Trig-C, respectively. We attribute the strong zero-shot generalizability of our model to knowledge-based conditioning. By casting the event extraction task as a generation problem, and injecting event type names, the model gains task-specific information that is especially valuable.

Table 4 Experiment results for zero-shot learning on sentence-level (ACE05-EN) and document-level (WIKIEVENTS) datasets

ACE05-EN Models	Arg-C			Trig-C		
	F1	P	R	F1	P	R
Zero-Shot [10]	6.3	-	-	53.5	-	-
Zero-Shot [9]	15.8	-	-	49.1	-	-
Zero-Shot [27]	16.8	-	-	41.7	-	-
Text2Event [7]	21.8	40.3	14.9	31.8	62.7	21.3
KC-GEE$_{\text{Full}}$	49.5 (+27.7)	43.6	57.5	62.7 (+9.2)	59.1	66.7
KC-GEE$_{\text{Adapter}}$	34.0	34.0	34.0	59.7	60.5	59.0
KC-GEE$_{\text{Fine tuning+Adapter}}$	42.6	32.6	61.7	53.1	44.1	66.7
KC-GEE$_{\text{Prefix}}$	20.2	21.4	19.1	48.8	46.5	51.3
KC-GEE$_{\text{Fine tuning+Prefix}}$	43.2	37.5	51.1	58.1	53.2	64.1
WIKIEVENTS						
Text2Event [7]	15.8	20.1	13.0	30.7	41.9	24.2
KC-GEE$_{\text{Full}}$	20.2 (+4.4)	23.3	17.8	36.1 (+5.4)	42.1	31.6
KC-GEE$_{\text{Adapter}}$	0.9	3.1	0.5	2.7	9.4	1.6
KC-GEE$_{\text{Fine tuning+Adapter}}$	9.7	25.9	6.0	16.9	41.4	10.6
KC-GEE$_{\text{Prefix}}$	7.1	11.3	5.1	14.9	17.0	13.3
KC-GEE$_{\text{Fine tuning+Prefix}}$	13.6	24.5	9.1	24.3	42.6	16.6

5.3 Ablation study

This section analyzes the effects of prefix encoder conditioning, prefix decoder conditioning, prefix cross-attention, and constrained decoding in KC-GEE. We designed five ablated variants based on T5-base:

- w/o enc-cond indicates KC-GEE without prefix encoder conditioning.
- w/o dec-cond indicates KC-GEE without prefix decoder conditioning.
- w/o both-cond indicates KC-GEE without both prefix encoder and prefix decoder conditioning.
- w/o const-dec discards the constrained decoding during inference and generates event structures as an unconstrained generation model.
- w/o cross-att indicates KC-GEE without prefix cross-attention.

Table 5 shows the results of ACE05-EN on the test set for the supervised learning setting. We observe that:

- constrained decoding helps, but not too much;
- prefix encoder and decoder conditioning are the most effective module if we use both of them together.

Furthermore, as constraint decoding limits the argument and the trigger word generated by the model, our method does not suffer from hallucination problems.

5.4 Analysis

In this section, we conduct comprehensive studies to analyze the design of our method from different perspectives.

5.4.1 Prefix length

Longer prefixes provide more knowledge-based conditioning information to the model. Table 6 summarizes the result of the model performance with different prefix lengths on the WIKIEVENTS dataset. As shown in the table, longer prefixes improve model performance on Arg-C, while performance on Trig-C improves with increases in prefix length until 20, after which the F1 value plateaus. However, longer prefixes requires more model parameters. Therefore, we set the prefix length to 20 as a trade-off between model performance and computational efficiency.

Table 5 The ablation study in the supervised learning setting on the ACE05-EN dataset based on T5-base

Models	Arg-C			Trig-C		
	F1	P	R	F1	P	R
w/o enc-cond	46.91	44.60	49.48	68.80	65.91	71.96
w/o dec-cond	45.59	42.02	49.83	68.79	65.89	71.94
w/o both-cond	49.41	47.44	51.56	68.35	66.35	70.47
w/o const-dec	48.06	45.83	50.52	67.92	64.72	71.46
w/o cross-att	49.10	45.01	53.99	68.77	64.84	73.20
KC-GEE Full	51.5	48.1	55.6	70.1	66.7	73.9

Table 6 Zero-shot learning on WIKIEVENTS with different prefix lengths

Prefix length	Arg-C			Trig-C		
	F1	P	R	F1	P	R
5	4.44	16.67	2.56	10.61	36.84	6.19
10	13.14	45.00	7.69	13.64	47.37	7.96
20	20.2	23.3	17.8	36.1	42.1	31.6
50	21.10	25.00	18.26	29.07	45.00	21.47
100	25.56	36.51	19.66	28.89	38.81	23.01

5.4.2 Knowledge-based conditioning

A key contribution to our method is the introduction of knowledge-based conditioning information. We analyze this component from two perspectives: (1) conditioning information and (2) injection mechanism.

Conditioning information In Table 7, we analyze in detail the effect of different knowledge-based conditioning, in which we fix the prefix length at 20. As can be seen, having no knowledge-based conditioning (None) results in poor performance across the board. Injecting task-agnostic information (Pseudo token) provides noticeable gains on Trig-C. Furthermore, injecting event type information substantially improves performance on both Arg-C and Trig-C. Adding role information improves performance on Arg-C but decreases performance on Trig-C. Finally, having all three types of conditioning does not bring additional benefits.

This comparison highlights the effectiveness of knowledge-based conditioning on event type information. Additionally, incorporating role information enhances argument extraction performance, although it comes at the expense of trigger extraction.

Injection mechanism The bottom four rows in Tables 3 and 4 display variants of our KCGEE model, where knowledge-based conditioning information is injected in different ways, as depicted in Figure 3. Specifically, the "Adapter" variant injects knowledge-based conditioning information in an adapter layer over each Transformer layer while freezing the parameters of the underlying language model. The "Fine tuning+Adapter" variant employs adapter layers and updates the language model's parameters. The "Prefix" variant prepends the knowledge-based conditioning vectors $\boldsymbol{h}$ to each layer in the language model while keeping the language model's parameters frozen. Finally, the "Fine tuning+Prefix" variant additionally updates the parameters of the language model. We can make the following observations from Tables 3 and 4.

Table 7 Zero-shot learning on WIKIEVENTS with different knowledge-based conditioning

Conditioning information	Arg-C			Trig-C		
	F1	P	R	F1	P	R
None	6.3	11.9	4.3	3.9	7.5	2.7
Pseudo token	5.7	16.7	3.4	17.6	52.2	10.6
Event type	15.7	26.5	11.1	29.5	53.5	20.4
Event type + Role	20.2	23.3.7	17.8	36.1	42.1	31.6
Pseudo token + Event type + Role	15.4	21.5	12.0	16.6	22.1	13.3

As expected, updating the language model's parameters (i.e. "Fine-tuning") is much more effective than when the parameters are frozen, regardless of whether Knowledge-based conditioning information is injected as adapters or prefixes.

The "Adapter" style of injection performs especially poorly on WIKIEVENTS in both the supervised and zero-shot settings. In comparison, on WIKIEVENTS, "Prefix" inject is able to outperform Text2Event in the zero-shot setting and achieves competitive performance in the supervised setting.

6 Conclusion

In this paper, we formulate the problem of event extraction as a natural-language generation task. We propose KC-GEE, a generation-based document-level event extraction technique that leverages large pretrained language models. A key component in KC-GEE is a novel Knowledge-based conditioning technique that injects event type information into the model as prefixes to enable zero-shot learning capability. The cross-attention mechanism in the prefix generator also facilitates effective document handling. Extensive experiments on two benchmark datasets demonstrate the effectiveness of KC-GEE, which achieves state-of-the-art performance in document-level extraction in fully supervised and zero-shot settings. On the challenging WIKIEVENTS dataset, KC-GEE substantially outperforms the current best model by up to 27.7 absolute points in F1. In future work, we will investigate incorporating attention mechanisms or graph-based techniques to integrate external knowledge, like event descriptions, improving zero-shot event extraction performance.

7 Limitations

In this paper, we explore a new method for solving zero-shot and document-level event extraction through Knowledge-based conditioning. The model has the ability of zero-shot transfer learning primarily from *seen roles composition*. This means that although the model has not seen any instances of zero-shot event types during training, the schema of these event types is available in the training stage. There is still a gap in the true zero-shot generalization.

Author Contributions Tongtong Wu: Model and algorithm design, implementation, data curation, evaluation and analysis of results, writing the paper draft and final version. Fatemeh Shiri: Model and algorithm design implementation, data curation, evaluation and analysis of results, writing the paper draft and final version. Jingqi Kang: Data curation, conducting the ablation study. Guilin Qi: Results analysis, writing the final version. Gholamreza Haffari: Conceptualization, model and algorithm design, results analysis and discussions, writing the final version. Yuan-Fang Li: Conceptualisation, model and algorithm design, results analysis and discussions, writing the final version.

Funding Open Access funding enabled and organized by CAUL and its Member Institutions. This material is based on research sponsored by Air Force Research Laboratory and DARPA under agreement number FA8750-19-2-0501 and HR001122C0029. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon.

Availability of data and materials All code and data will be made publicly available on acceptance of the paper.

Declarations

Ethical approval Not applicable.

Competing interests The authors declare no competing interests.

References

1. Li, Q., Peng, H., Li, J., Hei, Y., Sun, R., Sheng, J., Guo, S., Wang, L., Yu, P.S.: A survey on deep learning event extraction: Approaches and applications. TNNLS, 1–21 (2022)
2. Xu, R., Liu, T., Li, L., Chang, B.: Document-level event extraction via heterogeneous graph-based interaction model with a tracker. In: Proceedings of ACL, pp. 3533–3546 (2021)
3. Paolini, G., Athiwaratkun, B., Krone, J., Ma, J., Achille, A., Anubhai, R., Santos, C.N.d., Xiang, B., Soatto, S.: Structured prediction as translation between augmented natural languages. In: ICLR (2021)
4. Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., Liu, P.J.: Exploring the limits of transfer learning with a unified text-to-text transformer. J Mach Learn Res **21**(140), 1–67 (2020)
5. Li, S., Ji, H., Han, J.: Document-level event argument extraction by conditional generation. In: Proceedings of NAACL, pp. 894–908 (2021)
6. Hsu, I.-H., Huang, K.-H., Boschee, E., Miller, S., Natarajan, P., Chang, K.-W., Peng, N.: Degree: A data-efficient generation-based event extraction model. In: Proceedings of NUSE@NAACL (2022)
7. Lu, Y., Lin, H., Xu, J., Han, X., Tang, J., Li, A., Sun, L., Liao, M., Chen, S.: Text2Event: Controllable sequence-to-structure generation for end-to-end event extraction. In: Proceedings of ACL, pp. 2795–2806 (2021)
8. Nguyen, T.M., Nguyen, T.H.: One for all: Neural joint modeling of entities and events. In: Proceedings of AAAI, vol. 33, pp. 6851–6858 (2019)
9. Huang, L., Ji, H., Cho, K., Dagan, I., Riedel, S., Voss, C.: Zero-shot transfer learning for event extraction. In: Proceedings of ACL, pp. 2160–2170 (2018)
10. Zhang, H., Wang, H., Roth, D.: Zero-shot Label-aware Event Trigger and Argument Classification. In: Findings of ACL, pp. 1331–1340 (2021)
11. Walker, C., Strassel, J.M.S., Maeda, K.: Ace 2005 multilingual training corpus. https://catalog.ldc.upenn.edu/LDC2006T06 (2006)
12. Shiri, F., Wu, T., Li, Y.-F., Haffari, G.: Tcg-event: Effective task conditioning for generation-based event extraction. In: Proceedings of ALTA workshop (2022)
13. Shen, S., Wu, T., Qi, G., Li, Y., Haffari, G., Bi, S.: Adaptive knowledge-enhanced bayesian meta-learning for few-shot event detection. In: Findings of ACL, pp. 2417–2429 (2021)
14. Zhang, N., Chen, X., Xie, X., Deng, S., Tan, C., Chen, M., Huang, F., Si, L., Chen, H.: Document-level relation extraction as semantic segmentation. In: Proceedings of IJCAI, pp. 3999–4006 (2021)
15. Huang, K.-H., Peng, N.: Document-level event extraction with efficient end-to-end learning of cross-event dependencies. In: Proceedings of NUSE@NAACL, pp. 36–47 (2021)
16. Yang, H., Sui, D., Chen, Y., Liu, K., Zhao, J., Wang, T.: Document-level event extraction via parallel prediction networks. In: Proceedings of ACL, pp. 6298–6308 (2021)
17. Liu, X., Luo, Z., Huang, H.: Jointly multiple events extraction via attention-based graph information aggregation. In: Proceedings of EMNLP, pp. 1247–1256 (2018)
18. Lou, D., Liao, Z., Deng, S., Zhang, N., Chen, H.: Mlbinet: A cross-sentence collective event detection network. In: Proceedings of ACL, pp. 4829–4839 (2021)
19. Ebner, S., Xia, P., Culkin, R., Rawlins, K., Durme, B.V.: Multi-sentence argument linking. In: Proceedings of ACL, pp. 8057–8077 (2020)

20. Li, S., Ji, H., Han, J.: Document-level event argument extraction by conditional generation. In: Proceedings of NAACL, pp. 894–908 (2021)
21. Lison, P., Barnes, J., Hubin, A., Touileb, S.: Named entity recognition without labelled data: A weak supervision approach. In: Proceedings of ACL, pp. 1518–1533 (2020)
22. Du, X., Rush, A.M., Cardie, C.: GRIT: generative role-filler transformers for document-level event entity extraction. In: Proceedings of EACL, pp. 634–644 (2021)
23. Zhang, Z., Kong, X., Liu, Z., Ma, X., Hovy, E.H.: A two-step approach for implicit event argument detection. In: Proceedings of ACL, pp. 7479–7485 (2020)
24. Geng, Y., Chen, J., Zhuang, X., Chen, Z., Pan, J.Z., Li, J., Yuan, Z., Chen, H.: Benchmarking knowledge-driven zero-shot learning. J. Web Semant., 100757 (2023)
25. Chen, J., Geng, Y., Chen, Z., Horrocks, I., Pan, J.Z., Chen, H.: Knowledge-aware zero-shot learning: Survey and perspective. In: Proceedings of IJCAI, pp. 4366–4373 (2021)
26. Zhang, H., Wang, H., Roth, D.: Zero-shot label-aware event trigger and argument classification. In: Findings of ACL, pp. 1331–1340 (2021)
27. Lyu, Q., Zhang, H., Sulem, E., Roth, D.: Zero-shot event extraction via transfer learning: Challenges and insights. In: Proceedings of ACL, pp. 322–332 (2021)
28. Wang, Y., Wood, I., Wan, S., Dras, M., Johnson, M.: Mention flags (MF): Constraining transformer-based text generators. In: Proceedings of ACL), pp. 103–113 (2021)
29. Wang, Y., Xu, C., Sun, Q., Hu, H., Tao, C., Geng, X., Jiang, D.: PromDA: Prompt-based data augmentation for low-resource NLU tasks. In: Proceedings of ACL), pp. 4242–4255 (2022)
30. Wang, Y., Zheng, J., Xu, C., Geng, X., Shen, T., Tao, C., Jiang, D.: KnowDA: All-in-one knowledge mixture model for data augmentation in low-resource NLP. In: ICLR (2023)
31. Ye, H., Zhang, N., Bi, Z., Deng, S., Tan, C., Chen, H., Huang, F., Chen, H.: Learning to ask for data-efficient event argument extraction (student abstract). In: Proceedings of AAAI, pp. 13099–13100 (2022)
32. Yao, Y., Mao, S., Chen, X., Zhang, N., Deng, S., Chen, H.: Schema-aware reference as prompt improves data-efficient relational triple and event extraction. CoRR (2022)
33. Liu, X., Huang, H., Shi, G., Wang, B.: Dynamic prefix-tuning for generative template-based event extraction. Proceedings of ACL (2022)
34. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L., Polosukhin, I.: Attention is all you need. In: Proceedings of NeurIPS, pp. 5998–6008 (2017)
35. Li, X.L., Liang, P.: Prefix-tuning: Optimizing continuous prompts for generation. In: Proceedings of ACL, pp. 4582–4597 (2021)
36. Lin, Y., Ji, H., Huang, F., Wu, L.: A joint neural model for information extraction with global features. In: Proceedings of ACL, pp. 7999–8009 (2020)
37. Zhang, T., Ji, H., Sil, A.: Joint entity and event extraction with generative adversarial imitation learning. Data Intell. **1**(2), 99–120 (2019)
38. Du, X., Cardie, C.: Event extraction by answering (almost) natural questions. In: Proceedings of EMNLP, pp. 671–683 (2020)
39. Li, F., Peng, W., Chen, Y., Wang, Q., Pan, L., Lyu, Y., Zhu, Y.: Event extraction as multi-turn question answering. In: Findings of EMNLP, pp. 829–838 (2020)
40. Wadden, D., Wennberg, U., Luan, Y., Hajishirzi, H.: Entity, relation, and event extraction with contextualized span representations. In: Proceedings of EMNLP, pp. 5783–5788 (2019)
41. Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., Wojna, Z.: Rethinking the inception architecture for computer vision. In: Proceedings of CVPR, pp. 2818–2826 (2016)
42. Loshchilov, I., Hutter, F.: Decoupled weight decay regularization. In: ICLR (2017)
43. Liu, J., Chen, Y., Xu, J.: Machine reading comprehension as data augmentation: A case study on implicit event argument extraction. In: Proceedings of EMNLP, pp. 2716–2725 (2021)
44. Lin, J., Chen, Q.: Poke: A prompt-based knowledge eliciting approach for event argument extraction. CoRR (2021)

Authors and Affiliations

Tongtong Wu[1,2] · Fatemeh Shiri[2] · Jingqi Kang[1] · Guilin Qi[1] · Gholamreza Haffari[2] · Yuan-Fang Li[2]

Tongtong Wu
wutong8023@seu.edu.cn; tongtong.wu@monash.edu

Fatemeh Shiri
fatemeh.shiri@monash.edu

Jingqi Kang
kjq@seu.edu.cn

Guilin Qi
gqi@seu.edu.cn

Gholamreza Haffari
gholamreza.haffari@monash.edu

1 School of Computer Science and Engineering, Southeast University, Nanjing 211189, Jiangsu, China

2 Department of DS&AI, Faculty of IT, Monash University, Melbourne 3800, VIC, Australia