TCG-Event: Effective Task Conditioning for Generation-based Event Extraction

Fatemeh Shiri, Tongtong Wu, Yuan-Fang Li, Gholamreza Haffari
Department of DS&AI, Faculty of Information Technology, Monash University, Melbourne, Australia
fatemeh.shiri, tongtong.wu, YuanFang.Li, Gholamreza.Haffari@monash.edu

Abstract

Event extraction is an important but challenging task. Many existing techniques decompose it into subtasks of event and argument detection/classification, which are themselves 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 pre-trained language models. However, the large diversity of available event types makes it hard for generative models to effectively select the correct corresponding templates to predict the structured sequence. In this paper, we propose a task-conditioned generation-based event extraction model, TCG-Event, that addresses these challenges. A key contribution of TCG-Event is a novel task conditioning technique that injects event name information as prefixes into each layer of an encoder-decoder-based language model, thus enabling effective supervised learning. Our experiments on two benchmark datasets demonstrate the strong performance of our TCG-Event model.

1 Introduction

Event extraction (Li et al., 2021a) aims at extracting structured event records from unstructured text. For example, as shown in Figure 1, event extraction aims to map the sentence "Two homemade pressure-cooker bombs are detonated remotely, killing three and injuring some 260 others" to four predefined event types, e.g., <*event type*: explosion, *trigger word*: detonated, *role:bomber*: Tsarnaevs, ..., *role:bomb*: homemade pressure-cooker bombs, *role:place*: Boston Marathon>, as well as other events that are triggered by words *killing*, *injuring* and *lost limbs*.

Event extraction is challenging because of the diversity of natural language expressions and the complexity of event structures. These challenges become even more severe in sentences in which the

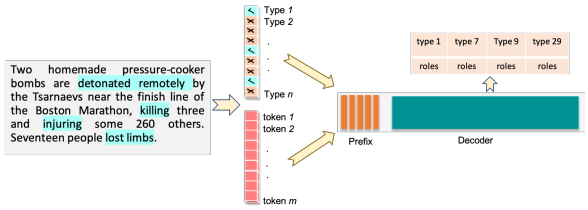

Figure 1: An illustration of the event extraction as a structured generation, with introducing the event type as the task-conditioning to prefix, the decoder can selectively generate sequentialized event representation, which is mentioned in the input text.

text generally contains more events. Currently, most event extraction methods employ a decomposition-based approach (Xu et al., 2021), i.e., decomposing the structured prediction problem of a complex event into classification over substructures, such as trigger detection, entity recognition, and argument classification. Many of these methods tackle the subproblems separately, which requires additional annotations for each stage (Paolini et al., 2021). Furthermore, designing an optimal composition architecture of different subtasks is very challenging.

Natural language generation techniques have been successfully applied to several NLP tasks (Raffel et al., 2020; Li et al., 2021b; Hsu et al., 2022). They 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 Lu et al. (2021) proposes to generate 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 approach of extraction-as-generation is the removal of the need for fine-grained token-level annotations, which are typically utilised in previous event extraction approaches (Nguyen and Nguyen,

2019), thus enjoying greater feasibility.

Structured prediction problems such as event extraction usually assume an external schema to format the output. In contrast, natural language generation problems do not make this assumption. Motivated by this distinction, we propose a novel *task conditioning* technique that injects event type information as *prefixes* on layers of the underlying pretrained language model.

Our main contributions are as follows.

- We propose a novel task conditioning technique that dynamically injects event-type information to both the encoder and decoder of a pretrained language model.
- We carefully design a prefix-based injection mechanism that incorporates cross-attention to improve event extraction.
- We conducted extensive experiments in the fully supervised setting on two benchmark datasets. Our evaluation consistently shows strong performance.

2 Related Work

Event extraction is the task of extracting structured event records from unstructured text (Li et al., 2021b; Shiri et al., 2021). Many approaches have been proposed for sentence-level event extraction (Christopher Walker and Maeda, 2006), varies from hand-designed features (Shen et al., 2021) and neural-learned features (Zhang et al., 2021; Huang and Peng, 2021). Yet many real-world applications need event extraction (Frisoni et al., 2021; He et al., 2021; Verspoor et al., 2016; Nguyen and Verspoor, 2019; Yang et al., 2021; Zhang et al., 2021; Huang and Peng, 2021), in which the information of an event may be mentioned in multi-sentences (Ebner et al., 2019; Li et al., 2021c). Moreover, most of works adopt decomposition strategies in event extraction (Xu et al., 2021), which employ trigger detection (Shen et al., 2021), entity recognition (Lison et al., 2020; Du et al., 2021), and argument classification (Zhang et al., 2020). These decomposition strategies showed high performance while introducing more detailed annotation to model training (Lu et al., 2021; Li et al., 2021b). Inspired by the success of pretrained language models and the corresponding natural language generation-based paradigm for various NLP tasks (Raffel et al., 2020; Li et al., 2021b; Hsu et al., 2022) tackle event extraction as controlled event generation. (Hsu et al., 2022) is an end-to-end conditional generation method with manually designed discrete prompts for each event type, which needs more human effort to find the optimal prompt. To remove the complexity of template combination in extracting multiple events, Lu et al. (2021) proposed to generate the event records directly with a pretrained encoder-decoder architecture and a constrained decoding mechanism. This extraction-as-generation approach does not require fine-grained token-level annotations that are typically needed by previous event extraction methods (Huang et al., 2021; Li et al., 2021b). Liu et al. (2022) proposed a generative template-based event extraction method with dynamic prefixes by integrating context information with type-specific prefixes only to the encoder to learn a context-specific prefix for each context. In contrast, we inject event-type information into both the encoder and the decoder. Particularly, we use interactions between the event type information and the context and inject it into the decoder.

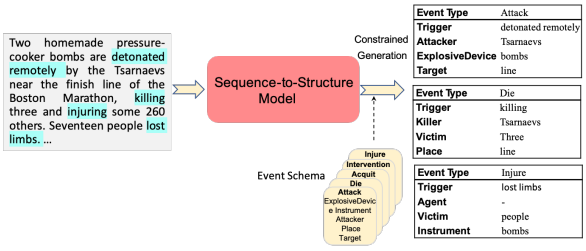

Figure 2: The event extraction task.

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 $\boldsymbol{x} := \{x_1, ..., x_{|\boldsymbol{x}|}\}$ comprises tokens x_i, where $|\boldsymbol{x}|$ denotes the sequence length. Given an input sequence, an event extraction model aims to extract one or more structured events, where each event is specified by (i) the event type $e \in \mathcal{E}$ filled with the trigger word t from the sequence, and (ii) the roles $\mathcal{R}_e \subseteq \mathcal{R}$ filled with the corresponding arguments from the sequence (see Figure 2).

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 sequence constrained by the schema (Lu et al., 2021).

The generated sequence is a linearised representation of events mentioned in the sequence. Specifically, given a text with token sequence $\boldsymbol{x}$ as

input, a generation-based extraction model such as TCG-Event outputs the linearised events representations $\boldsymbol{y} = \langle y_1, y_2, ..., y_{|\boldsymbol{y}|} \rangle$, where each event y_i is denoted by $\langle e_i, t_i, \langle r_{i,1}, a_{i,1} \rangle, ..., \langle r_{i,|r|}, a_{i,|r|} \rangle \rangle$. 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 sequence $\boldsymbol{x}$); furthermore, $r_i \in \mathcal{R}$ and a_i denote roles and arguments (subspans of the sequence $\boldsymbol{x}$).

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

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

where each layer of Encoder$(\cdot)$ is a Transformer block (Vaswani et al., 2017) 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:

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

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 $p(\boldsymbol{y}|\boldsymbol{x})$ is then,

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

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

4 Task Conditioning in Event Generation

In this paper, we investigate how to best leverage pre-trained large language models (LLMs) as the backbone encoder-decoder model for event extraction.[1] Using LLMs is nowadays part of standard practice in NLP, as they lead to strong performance.

Given a labeled training dataset $\mathcal{D}$, we investigate how to best specialise the pre-trained LLM to the event extraction task via prefix-tuning (Li and Liang, 2021). In this section, we show how to effectively *condition* the generation process on the event extraction task as well as the given sequence. One may specialise the underlying LLM to the event extraction task through other methods as well, e.g. fine-tuning of the LLM parameters or adapters injected to the encoder and/or decoder of the LLM (see Figure 3). We show in our experiments that prefix-tuning is more effective than those methods.

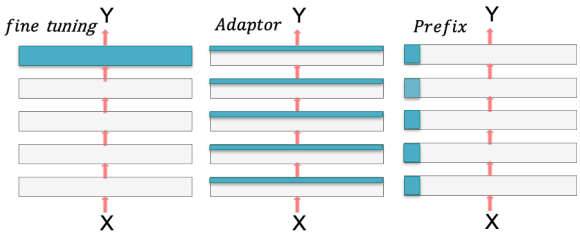

Figure 3: A high-level illustration of three candidate task-conditioning injection paradigm for encoder-decoder models: fine-tuning, adapter-tuning, and prefix-tuning. For each tuning type, each block represents a transformer block in the pretrained language model, and the blue blocks indicate the new-added parameters in the pretrained model.

Our desiderata for prefix-conditioning of a pre-trained LLM 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 sequence, and (iii) flexible schema modifications that may happen after the model is trained in the 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 sequence. See Figure 4 for an overview of the framework.

Encoder Conditioning. We condition the encoder on the event types of the underlying event extraction task. Given the event types $\boldsymbol{e} = \{e_1, e_2, ..., e_{|\boldsymbol{e}|}\} \subseteq \mathcal{E}$ 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.

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

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

[1] In our experiments, we make use of T5 (Raffel et al., 2020), but our methods are applicable to other large pre-trained encoder-decoder models as well.

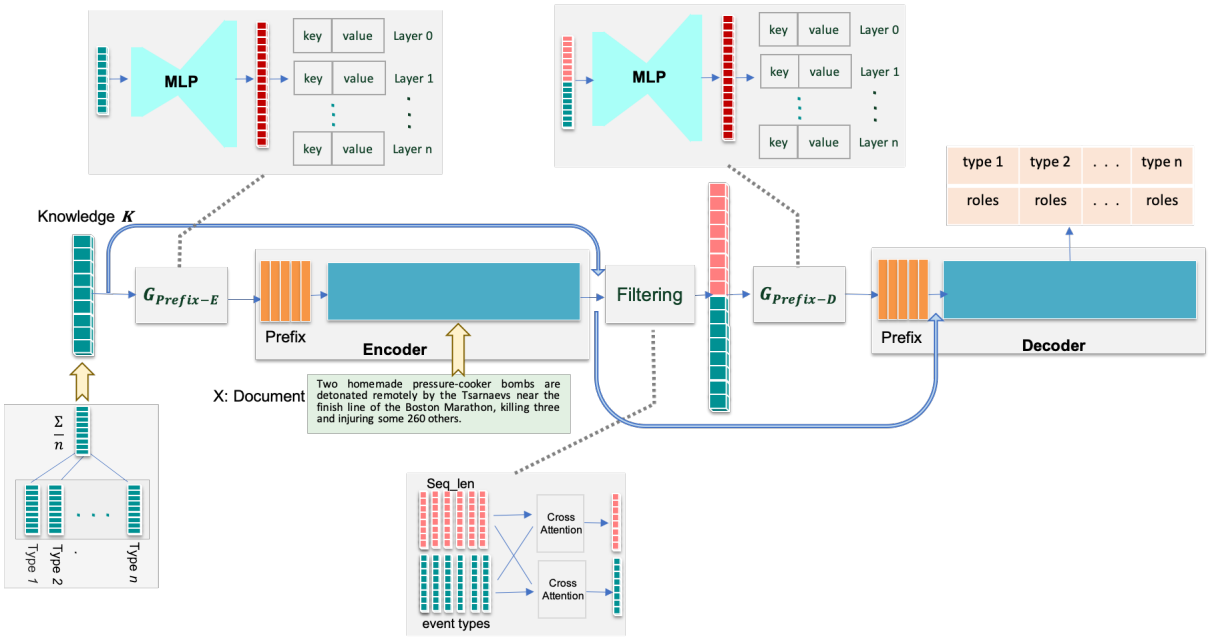

Figure 4: An illustration of our end-to-end framework TCG-Event, where the main architecture in the central is a transformer-based encoder-decoder, the lower blocks represent the task-conditioning construction modules for encoder and decoder respectively and the upper blocks represent the task-conditioning injection modules for encoder and decoder respectively.

suggest that it is flexible to add or remove an event type representation from task conditioning.

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

Specifically, let $\mathbf{H}_{\boldsymbol{x}}$ denote the representation of the tokens of the input sequence $\boldsymbol{x}$. We combine the sequence 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,

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

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.

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 $g: \mathbb{R}^{2\times d'} \mapsto \mathbb{R}^{k\times|\mathbf{H}_i|}$, 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 Equations 1 and 2 are modified as follows:

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

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

where $\mathbf{Z}_{enc}$ and $\mathbf{Z}_{dec}$ can be thought as pseudo-prefix tokens impacting the generation process (Li and Liang, 2021).

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

$$\theta^* = \underset{\theta}{\text{argmin}} \sum_{(\boldsymbol{x},\boldsymbol{y})\in\mathcal{D}} \log p_\theta(\boldsymbol{y}|\boldsymbol{x},\boldsymbol{e}) \tag{9}$$

where $\mathcal{D}$ is the training set, and

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

For inference, we use constrained decoding (Lu et al., 2021).

5 Experiments

We evaluate our TCG-Event model against a number of recent strong models. In particular, we

Dataset	Event Type	Argument Type	Train	Dev	Test	Instance	Events per Instance
ACE05-EN	33	22	17,172	923	832	1-sent.	Single/Multiple
RAMS	38	65	7,329	924	871	5-sent.	Single

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 number of instances in different splits.

evaluate it in the supervised learning setting for both sentence-level and paragraph-level extraction tasks to demonstrate the greater effectiveness of our model in these challenging scenarios.

5.1 Evaluation setup

Datasets. We carry out experiments on two event extraction datasets, including the sentence-level dataset Automatic Content Extraction 2005 (ACE05-EN) (Christopher Walker and Maeda, 2006); as well as a paragraph-level dataset: Roles Across Multiple Sentences (RAMS) (Ebner et al., 2019). Statistics of the two datasets can be found in Table 1. Note that we use the official splits of the two datasets for better reproducibility. It is worth noting that these datasets are challenging due to three factors. (1) ***Context length***: each instance in ACE05-EN contains only one sentence, while in RAMS, instances are paragraphs (five sentences). (2) ***Event density***: each instance in RAMS contains only one event, while multiple events could be present in one instance in ACE05-EN. (3) ***Data scarcity***: the amount of training data in ACE05-EN is more than two times that in RAMS.

Evaluation Metrics. We employ the same evaluation metrics used in previous work (Lu et al., 2021; Lin et al., 2020), i.e. F1, precision and recall, for both trigger extraction (Trig-C) and arguments extraction (Arg-C).

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

Baselines. We evaluate TCG-Event 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 utilize token annotations, in which each token in an instance is annotated with event labels, along with golden entity annotation to facilitate event extraction. Joint3EE (Lin et al., 2020) is a multi-task model that jointly performs entity, trigger, and argument extraction by shared Bi-GRU hidden representations. DYGIE++ (Nguyen and Nguyen, 2019) is a BERT-based extraction framework that models text spans and captures within-sentence and cross-sentence context. GAIL (Zhang et al., 2019) 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 (Lin et al., 2020) 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. TANL (Paolini et al., 2021), a sequence generation-based method, tackles event extraction in a trigger-argument pipeline. Multi-task TANL is the extended version of TANL which transfers structure knowledge from other tasks. BERT-QA (Du and Cardie, 2020) and MQAEE (Li et al., 2020) consider event extraction as a sequence of extractive question answering problems.

Similar to Text2Event (Lu et al., 2021), we use *Parallel text-record annotation*, which only requires (instance, event) pairs without expensive, fine-grained token-level or entity-level annotations. As can be seen in an instance of such an annotation, ⟨"His two brothers were executed.", {Type: Injure, Trigger: tortured, ...}⟩, parallel text-record annotation is the least demanding and thus more practical annotation level. We compare our method with Text2Event (Lu et al., 2021), which introduces a sequence-to-structure generation model that addresses the missing event structure issue via constrained decoding.

Implementation Details We build our TCG-Event method on the T5-base pretraind language model and train it for 120 epochs with a learning rate of 1e-4 and batch size of 8 for the supervised setting. We also optimized TCG-Event using label smoothing Szegedy et al. (2016) and AdamW Loshchilov and Hutter (2017). The prefix length is set to 20 for all experiments in Section 5.2.

Models	Annotation	Arg-C			Trig-C			PLM
		F1	P	R	F1	P	R	
Joint3EE (Nguyen and Nguyen, 2019)	Token+Entity	52.1	52.1	52.1	69.8	68	71.8	-
DYGIE++ (Wadden et al., 2019)	Token+Entity	48.8	-	-	69.7	-	-	BERT-large
GAIL(Zhang et al., 2019)	Token+Entity	52.4	61.6	45.7	72.0	74.8	69.4	ELMo
OneIE (Lin et al., 2020)	Token+Entity	56.8	-	-	74.7	-	-	BERT-large
BERT-QA (Du and Cardie, 2020)	Token	53.3	56.8	50.2	72.4	71.1	73.7	2 x BERT-base
MQAEE (Li et al., 2020)	Token	53.4	-	-	71.7	-	-	3 x BERT-large
TANL (Paolini et al., 2021)	Token	47.6	-	-	68.4	-	-	T5-base
Multi-Task TANL (Paolini et al., 2021)	Token	48.5	-	-	68.5	-	-	T5-base
Text2Event (Lu et al., 2021)	Text-record	49.8	46.7	53.4	69.2	67.5	71.2	T5-base
TCG-Event$_{\text{Fine tuning+Prefix}}$	Text-record	49.0	47.3	50.7	69.3	**69.1**	69.5	T5-base
TCG-Event$_{\text{Full}}$	Text-record	**51.5**	**48.1**	**55.6**	**70.1**	66.7	**73.9**	T5-base

Table 2: Experiment results for the fully supervised event extraction on ACE05-EN. PLM represents the pre-trained 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.

5.2 Main Results

We compare our TCG-Event model in the fully supervised setting. The model evaluation is organised by dataset characteristics: sentence-level (ACE05-EN) and paragraph-level (RAMS).

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 (Lu et al., 2021).

It can be observed from the table that our TCG-Event model outperforms Text2Event on F1 for both argument extraction and trigger extraction. Moreover, our model surpasses the generation-based baselines using token annotation and achieves competitive performance with SOTA.

Sentence-level performance. As discussed above, among all compared models, our TCG-Event model, together with Text2Event (Lu et al., 2021), is trained on parallel text-record annotations, 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. As expected, more extensive training data induces stronger model performance. The last column also shows that the better-performing models make use of larger pretrained language models (PLMs), such as BERT-large. The larger capacity of these PLMs also contributes to model performance.

Paragraph-level performance. Table 3 shows the performance of the baseline (Text2Event), our model TCG-Event and its different variants for paragraph-level event extraction on the RAMS dataset. The other models in Table 2 are sentence-level and do not support this task. The majority of baselines focus only on event argument extraction from RAMS dataset, which did not handle triggers (Li et al., 2021c; Liu et al., 2021; Lin et al., 2021). Our model supports the joint extraction of both event triggers and arguments from the RAMS 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 RAMS. It is especially noteworthy that TCG-Event achieves better performance advantages over Text2Event.

The superiority can be attributed to a model design. Our cross-attention mechanism filters event-type tokens and argument tokens, allowing the model to better handle long context. Detailed analysis on the contributions of each model component will be presented below.

5.3 Ablation Study

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

- "w/o$_{\text{_encoder conditioning}}$": indicates TCG-Event without prefix encoder conditioning.
- "w/o$_{\text{_decoder conditioning}}$" indicates TCG-Event without prefix decoder conditioning.
- "w/o$_{\text{_both conditioning}}$" indicates TCG-Event without both prefix encoder and prefix decoder

Models	Arg-C			Trig-C		
	F1	P	R	F1	P	R
Text2Event (Lu et al., 2021)	29.81	28.98	30.69	67.13	67.09	67.16
TCG-Event$_{\text{Full}}$	**30.88**	**31.07**	30.70	**68.42**	**68.31**	**68.54**
TCG-Event$_{\text{Adapter}}$	23.11	21.34	25.21	62.28	62.10	62.46
TCG-Event$_{\text{Fine tuning+Adapter}}$	30.00	29.95	30.05	67.62	67.27	67.97
TCG-Event$_{\text{Prefix}}$	9.60	18.18	6.53	24.51	20.73	29.97
TCG-Event$_{\text{Fine tuning+Prefix}}$	30.53	30.19	**30.89**	65.87	65.17	66.59

Table 3: Results for supervised learning on the paragraph-level event extraction dataset RAMS.

Models	Arg-C			Trig-C		
	F1	P	R	F1	P	R
w/o$_{\text{_encoder conditioning}}$	46.91	44.60	49.48	68.80	65.91	71.96
w/o$_{\text{_decoder conditioning}}$	45.59	42.02	49.83	68.79	65.89	71.94
w/o$_{\text{_both conditioning}}$	49.41	47.44	51.56	68.35	66.35	70.47
w/o$_{\text{_constraint decoding}}$	48.06	45.83	50.52	67.92	64.72	71.46
w/o$_{\text{_cross attention}}$	49.10	45.01	53.99	68.77	64.84	73.20
TCG-Event-full	**51.5**	**48.1**	**55.6**	**70.1**	**66.7**	**73.9**

Table 4: The ablation study in the supervised learning setting on the ACE05-EN dataset based on T5-base.

Prefix length	Arg-C			Trig-C		
	F1	P	R	F1	P	R
5	45.67	41.79	50.35	68.74	66.21	71.46
10	46.58	42.96	50.87	69.50	66.37	72.95
20	51.51	48.08	55.55	70.12	66.71	73.89
50	51.50	48.00	55.56	70.19	66.94	73.77
100	51.80	48.31	55.83	68.64	66.2	72.95

Table 5: Results for supervised learning on ACE05-EN with different prefix lengths.

conditioning.

- "w/o$_{\text{_constraint decoding}}$" discards the constrained decoding during inference and generates event structures as an unconstrained generation model.
- "w/o$_{\text{_cross attention}}$" indicates TCG-Event without prefix cross-attention.

Table 4 shows the results of the test set of ACE05-EN 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 id we use both of them together.

Furthermore, as constraint decoding limits the argument and trigger words 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 prefix length perspectives.

Longer prefixes provide more task-conditioning information to the model. Table 5 summarizes the result of model performance of different prefix lengths on the ACE05-EN dataset. As can be seen, longer prefixes improve model performance on Arg-C, while performance on Trig-C improves with increases in prefix length until 20, after which F1 value plateaus. As longer prefixes demand more model parameters, we set the prefix length to 20 as a trade-off between model performance and computational efficiency.

6 Conclusion

In this paper, we formulate the problem of event extraction as a natural-language generation task. We propose TCG-Event, a generation-based event extraction technique that leverages large pre-trained language models. A key component in TCG-Event is a novel task conditioning technique that injects event-type information into the model as prefixes. The cross-attention mechanism in the prefix generator also facilitates effective long-text handling. Extensive experiments on two benchmark datasets demonstrate the effectiveness of TCG-Event, which achieves state-of-the-art performance in event extraction. On the challenging RAMS dataset, TCG-Event outperforms the current best model. For future work, we plan to further investigate new mechanisms of injecting task-specific information.

References

Julie Medero Christopher Walker, Stephanie Strassel and Kazuaki Maeda. 2006. Ace 2005 multilingual training corpus. https://catalog.ldc.upenn.edu/LDC2006T06.

Xinya Du and Claire Cardie. 2020. Event extraction by

answering (almost) natural questions. *arXiv preprint arXiv:2004.13625*.

Xinya Du, Alexander M. Rush, and Claire Cardie. 2021. GRIT: generative role-filler transformers for document-level event entity extraction. In *Proceedings of EACL*, pages 634–644.

Seth Ebner, Patrick Xia, Ryan Culkin, Kyle Rawlins, and Benjamin Van Durme. 2019. Multi-sentence argument linking. *arXiv preprint arXiv:1911.03766*.

Giacomo Frisoni, Gianluca Moro, and Antonella Carbonaro. 2021. A survey on event extraction for natural language understanding: Riding the biomedical literature wave. *IEEE Access*, 9:160721–160757.

Jiayuan He, Dat Quoc Nguyen, Saber A Akhondi, Christian Druckenbrodt, Camilo Thorne, Ralph Hoessel, Zubair Afzal, Zenan Zhai, Biaoyan Fang, Hiyori Yoshikawa, et al. 2021. Chemu 2020: Natural language processing methods are effective for information extraction from chemical patents. *Frontiers in Research Metrics and Analytics*, 6:654438.

I-Hung Hsu, Kuan-Hao Huang, Elizabeth Boschee, Scott Miller, Premkumar Natarajan, Kai-Wei Chang, and Nanyun Peng. 2022. Degree: A data-efficient generation-based event extraction model.

Kung-Hsiang Huang and Nanyun Peng. 2021. Document-level event extraction with efficient end-to-end learning of cross-event dependencies. In *Proceedings of NUSE-NAACL*, pages 36–47.

Kung-Hsiang Huang, Sam Tang, and Nanyun Peng. 2021. Document-level entity-based extraction as template generation. *CoRR*, abs/2109.04901.

Fayuan Li, Weihua Peng, Yuguang Chen, Quan Wang, Lu Pan, Yajuan Lyu, and Yong Zhu. 2020. Event extraction as multi-turn question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings*, pages 829–838.

Qian Li, Hao Peng, Jianxin Li, Yiming Hei, Rui Sun, Jiawei Sheng, Shu Guo, Lihong Wang, and Philip S. Yu. 2021a. Deep learning schema-based event extraction: Literature review and current trends. *CoRR*, abs/2107.02126.

Sha Li, Heng Ji, and Jiawei Han. 2021b. Document-level event argument extraction by conditional generation. In *Proceedings of NAACL*, pages 894–908.

Sha Li, Heng Ji, and Jiawei Han. 2021c. Document-level event argument extraction by conditional generation. *arXiv preprint arXiv:2104.05919*.

Xiang Lisa Li and Percy Liang. 2021. Prefix-tuning: Optimizing continuous prompts for generation. *CoRR*, abs/2101.00190.

Jiaju Lin, Jin Jian, and Qin Chen. 2021. Eliciting knowledge from language models for event extraction. *arXiv preprint arXiv:2109.05190*.

Ying Lin, Heng Ji, Fei Huang, and Lingfei Wu. 2020. A joint neural model for information extraction with global features. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7999–8009.

Pierre Lison, Jeremy Barnes, Aliaksandr Hubin, and Samia Touileb. 2020. Named entity recognition without labelled data: A weak supervision approach. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1518–1533, Online. Association for Computational Linguistics.

Jian Liu, Yufeng Chen, and Jinan Xu. 2021. Machine reading comprehension as data augmentation: A case study on implicit event argument extraction. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 2716–2725.

Xiao Liu, Heyan Huang, Ge Shi, and Bo Wang. 2022. Dynamic prefix-tuning for generative template-based event extraction. *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics*.

Ilya Loshchilov and Frank Hutter. 2017. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*.

Yaojie Lu, Hongyu Lin, Jin Xu, Xianpei Han, Jialong Tang, Annan Li, Le Sun, Meng Liao, and Shaoyi Chen. 2021. Text2Event: Controllable sequence-to-structure generation for end-to-end event extraction. In *Proceedings of ACL*, pages 2795–2806.

Dat Quoc Nguyen and Karin Verspoor. 2019. End-to-end neural relation extraction using deep biaffine attention. In *European conference on information retrieval*, pages 729–738. Springer.

Trung Minh Nguyen and Thien Huu Nguyen. 2019. One for all: Neural joint modeling of entities and events. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 6851–6858.

Giovanni Paolini, Ben Athiwaratkun, Jason Krone, Jie Ma, Alessandro Achille, Rishita Anubhai, Cicero Nogueira dos Santos, Bing Xiang, and Stefano Soatto. 2021. Structured prediction as translation between augmented natural languages. *arXiv preprint arXiv:2101.05779*.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67.

Shirong Shen, Tongtong Wu, Guilin Qi, Yuan-Fang Li, Gholamreza Haffari, and Sheng Bi. 2021. Adaptive knowledge-enhanced bayesian meta-learning for few-shot event detection. In *ACL Findings*, pages 2417–2429.

Fatemeh Shiri, Teresa Wang, Shirui Pan, Xiaojun Chang, Yuan-Fang Li, Reza Haffari, Van Nguyen, and Shuang Yu. 2021. Toward the automated construction of probabilistic knowledge graphs for the maritime domain. In *2021 IEEE 24th International Conference on Information Fusion (FUSION)*, pages 1–8. IEEE.

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. 2016. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 5998–6008.

Karin M Verspoor, Go Eun Heo, Keun Young Kang, and Min Song. 2016. Establishing a baseline for literature mining human genetic variants and their relationships to disease cohorts. *BMC medical informatics and decision making*, 16(1):37–47.

David Wadden, Ulme Wennberg, Yi Luan, and Hannaneh Hajishirzi. 2019. Entity, relation, and event extraction with contextualized span representations. *arXiv preprint arXiv:1909.03546*.

Runxin Xu, Tianyu Liu, Lei Li, and Baobao Chang. 2021. Document-level event extraction via heterogeneous graph-based interaction model with a tracker. In *Proceedings of ACL*, pages 3533–3546.

Hang Yang, Dianbo Sui, Yubo Chen, Kang Liu, Jun Zhao, and Taifeng Wang. 2021. Document-level event extraction via parallel prediction networks. In *Proceedings of ACL*, pages 6298–6308, Online. Association for Computational Linguistics.

Ningyu Zhang, Xiang Chen, Xin Xie, Shumin Deng, Chuanqi Tan, Mosha Chen, Fei Huang, Luo Si, and Huajun Chen. 2021. Document-level relation extraction as semantic segmentation. In *Proceedings of IJCAI*, pages 3999–4006.

Tongtao Zhang, Heng Ji, and Avirup Sil. 2019. Joint entity and event extraction with generative adversarial imitation learning. *Data Intelligence*, 1(2):99–120.

Zhisong Zhang, Xiang Kong, Zhengzhong Liu, Xuezhe Ma, and Eduard H. Hovy. 2020. A two-step approach for implicit event argument detection. In *Proceedings of ACL*, pages 7479–7485.