TEXT2EVENT: Controllable Sequence-to-Structure Generation for End-to-end Event Extraction

Yaojie Lu[1,3], **Hongyu Lin**[1], **Jin Xu**[4,*], **Xianpei Han**[1,2,*], **Jialong Tang**[1,3], **Annan Li**[1,3], **Le Sun**[1,2], **Meng Liao**[4], **Shaoyi Chen**[4]

[1]Chinese Information Processing Laboratory [2]State Key Laboratory of Computer Science
Institute of Software, Chinese Academy of Sciences, Beijing, China
[3]University of Chinese Academy of Sciences, Beijing, China
[4]Data Quality Team, WeChat, Tencent Inc., China
{yaojie2017,jialong2019,liannan2019}@iscas.ac.cn
{hongyu,xianpei,sunle}@iscas.ac.cn
{jinxxu,maricoliao,shaoyichen}@tencent.com

Abstract

Event extraction is challenging due to the complex structure of event records and the semantic gap between text and event. Traditional methods usually extract event records by decomposing the complex structure prediction task into multiple subtasks. In this paper, we propose TEXT2EVENT, a sequence-to-structure generation paradigm that can directly extract events from the text in an end-to-end manner. Specifically, we design a sequence-to-structure network for unified event extraction, a constrained decoding algorithm for event knowledge injection during inference, and a curriculum learning algorithm for efficient model learning. Experimental results show that, by uniformly modeling all tasks in a single model and universally predicting different labels, our method can achieve competitive performance using only record-level annotations in both supervised learning and transfer learning settings.

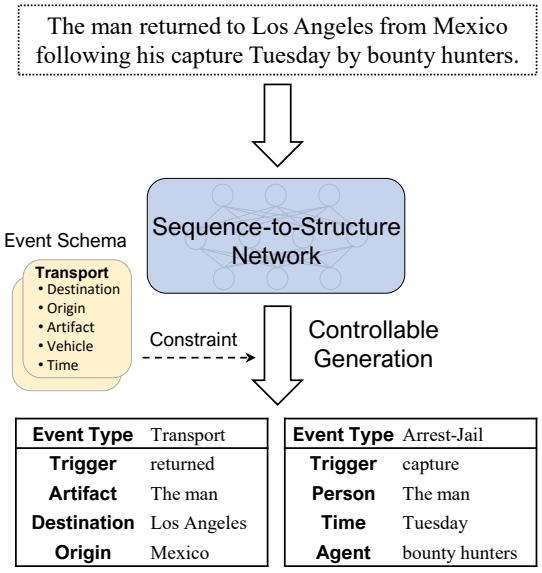


Figure 1: The framework of TEXT2EVENT. Here, TEXT2EVENT takes raw text as input and generates a *Transport* event and an *Arrest-Jail* event.

1 Introduction

Event extraction is an essential task for natural language understanding, aiming to transform the text into event records (Doddington et al., 2004; Ahn, 2006). For example, in Figure 1, mapping "The man returned to Los Angeles from Mexico following his capture Tuesday by bounty hunters." into two event records {Type: *Transport*, Trigger: returned, Arg1 Role: *Artifact*, Arg1: The man, Arg2 Role: *Destination*, Arg2: Los Angeles, ... } and {Type: *Arrest-Jail*, Trigger: capture, Arg1 Role: *Person*, Arg1: The man, Arg2 Role: *Agent*, Arg2: bounty hunters, ... }.

Event extraction is challenging due to the complex structure of event records and the semantic gap between text and event. First, an event record contains event type, trigger, and arguments, which form a table-like structure. And different event types have different structures. For example, in Figure 1, *Transport* and *Arrest-Jail* have entirely different structures. Second, an event can be expressed using very different utterances, such as diversified trigger words and heterogeneous syntactic structures. For example, both "the dismission of the man" and "the man departed his job" express the same event record {Type: *End-Position*, Arg1 Role: *PERSON*, Arg1: the man}.

Currently, most event extraction methods employ the decomposition strategy (Chen et al., 2015; Nguyen and Nguyen, 2019; Wadden et al., 2019; Zhang et al., 2019b; Du and Cardie, 2020; Li et al., 2020; Paolini et al., 2021), i.e., decomposing the prediction of complex event structures into multiple separated subtasks (mostly including entity recognition, trigger detection, argument classifica-

*Corresponding authors.

Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, pages 2795–2806
August 1–6, 2021. ©2021 Association for Computational Linguistics

tion), and then compose the components of different subtasks for predicting the whole event structure (e.g., pipeline modeling, joint modeling or joint inference). The main drawbacks of these decomposition-based methods are: (1) They need massive and fine-grained annotations for different subtasks, often resulting in the data inefficiency problem. For example, they need different fine-grained annotations for *Transport* trigger detection, for *Person* entity recognition, for *Transport.Artifact* argument classification, etc. (2) It is very challenging to design the optimal composition architecture of different subtasks manually. For instance, the pipeline models often lead to error propagation. And the joint models need to heuristically predefine the information sharing and decision dependence between trigger detection, argument classification, and entity recognition, often resulting in suboptimal and inflexible architectures.

In this paper, we propose a sequence-to-structure generation paradigm for event extraction – TEXT2EVENT, which can directly extract events from the text in an end-to-end manner. Specifically, instead of decomposing event structure prediction into different subtasks and predicting labels, we uniformly model the whole event extraction process in a neural network-based sequence-to-structure architecture, and all triggers, arguments, and their labels are universally generated as natural language words. For example, we generate a subsequence "Attack fire" for trigger extraction, where both "Attack" and "fire" are treated as natural language words. Compared with previous methods, our method is more data-efficient: it can be learned using only coarse parallel text-record annotations, i.e., pairs of ⟨sentence, event records⟩, rather than fine-grained token-level annotations. Besides, the uniform architecture makes it easy to model, learn and exploit the interactions between different underlying predictions, and the knowledge can be seamlessly shared and transferred between different components.

Furthermore, we design two algorithms for effective sequence-to-structure event extraction. First, we propose a constrained decoding algorithm, which can guide the generation process using event schemas. In this way, the event knowledge can be injected and exploited during inference on-the-fly. Second, we design a curriculum learning algorithm, which starts with current pre-trained language models (PLMs), then trains them on simple event substructure generation tasks such as trigger generation and independent argument generation, finally trains the model on the full event structure generation task.

We conducted experiments[1] on ACE and ERE datasets, and the results verified the effectiveness of TEXT2EVENT in both supervised learning and transfer learning settings. In summary, the contributions are as follows:

1. We propose a new paradigm for event extraction -– sequence-to-structure generation, which can directly extract events from the text in an end-to-end manner. By uniformly modeling all tasks in a single model and universally predicting different labels, our method is effective, data-efficient, and easy to implement.

2. We design an effective sequence-to-structure architecture, which is enhanced with a constrained decoding algorithm for event knowledge injection during inference and a curriculum learning algorithm for efficient model learning.

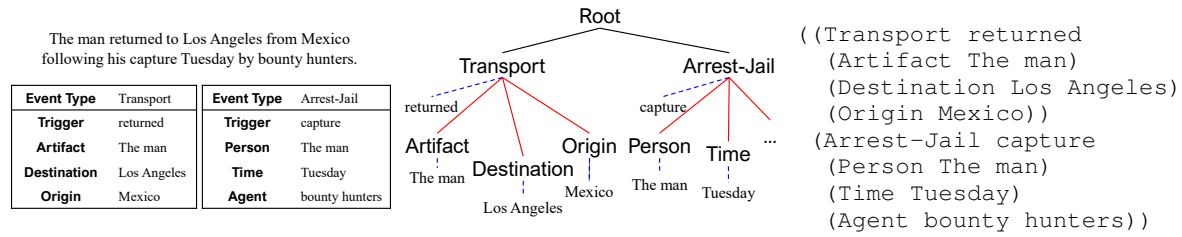
3. Many information extraction tasks can be formulated as structure prediction tasks. Our sequence-to-structure method can motivate the learning of other information extraction models.

2 TEXT2EVENT: End-to-end Event Extraction as Controllable Generation

Given the token sequence $x = x_1, ..., x_{|x|}$ of the input text, TEXT2EVENT directly generate the event structures $E = e_1, ..., e_{|E|}$ via an encoder-decoder architecture. For example, in Figure 1, TEXT2EVENT take the raw text as input and output two event records including {Type: *Transport*, Trigger: returned, Arg1 Role: *Artifact*, Arg1: The man, ...} and {Type: *Arrest-Jail*, Trigger: capture, ..., Arg2 Role: *Agent*, Arg2: bounty hunters, ...}.

For end-to-end event extraction, TEXT2EVENT first encodes input text, then generates the linearized structure using the constrained decoding algorithm. In the following, we first introduce how to reformulate event extraction as structure generation via structure linearization, then describe the sequence-to-structure model and the constrained decoding algorithm.

[1]Our source codes are openly available at https://github.com/luyaojie/text2event



(a) Record format. (b) Tree format. (c) Linearized format.

Figure 2: Examples of three event representations. The red solid line indicates the event-role relation; the blue dotted line indicates the label-span relation where the head is a label and the tail is a text span. For example, "*Transport*-returned" is a label-span relation edge, which head is "*Transport*" and tail is "returned".

2.1 Event Extraction as Structure Generation

This section describes how to linearize event structure so that events can be generated in an end-to-end manner. Specifically, the linearized event representations should: (1) be able to express multiple event records in a text as one expression; (2) be easy to reversibly converted to event records in a deterministic way; (3) be similar to the token sequence of general text generation tasks so that text generation models can be leveraged and transferred easily.

Concretely, the process of converting from record format to linearized format is shown in Figure 2. We first convert event records (Figure 2a) into a labeled tree (Figure 2b) by: 1) first labeling the root of the tree with the type of event (Root - Transport, Root - Arrest-Jail), 2) then connecting multiple event argument role types with event types (Transport - Artifact, Transport - Origin, etc.), and 3) finally linking the text spans from the raw text to the corresponding nodes as leaves (Transport - returned, Transport - Origin - Mexico, Transport - Artifact - The man, etc.). Given the converted event tree, we linearize it into a token sequence (Figure 2c) via depth-first traversal (Vinyals et al., 2015), where "(" and ")" are structure indicators used to represent the semantic structure of linear expressions. The traversal order of the same depth is the order in which the text spans appear in the text, e.g., first "return" then "capture" in Figure 2b. Noted that each linearized form has a virtual root – *Root*. For a sentence that contains multiple event records, each event links to *Root* directly. For a sentence that doesn't express any event, its tree format will be linearized as "()".

2.2 Sequence-to-Structure Network

Based on the above linearization strategy, TEXT2EVENT generates the event structure via a transformer-based encoder-decoder architecture (Vaswani et al., 2017). Given the token sequence $x = x_1, ..., x_{|x|}$ as input, TEXT2EVENT outputs the linearized event representation $y = y_1, ..., y_{|y|}$. To this end, TEXT2EVENT first computes the hidden vector representation $\mathbf{H} = \mathbf{h}_1, ..., \mathbf{h}_{|x|}$ of the input via a multi-layer transformer encoder:

$$\mathbf{H} = \text{Encoder}(x_1, ..., x_{|x|}) \quad (1)$$

where each layer of Encoder$(\cdot)$ is a transformer block with the multi-head attention mechanism.

After the input token sequence is encoded, the decoder predicts the output structure token-by-token with the sequential input tokens' hidden vectors. At the step i of generation, the self-attention decoder predicts the i-th token y_i in the linearized form and decoder state $\mathbf{h}_i^d$ as:

$$y_i, \mathbf{h}_i^d = \text{Decoder}([\mathbf{H}; \mathbf{h}_1^d, ..., \mathbf{h}_{i-1}^d], y_{i-1}) \quad (2)$$

where each layer of Decoder$(\cdot)$ is a transformer block that contains self-attention with decoder state $\mathbf{h}_i^d$ and cross-attention with encoder state $\mathbf{H}$.

The generated output structured sequence starts from the start token "⟨bos⟩" and ends with the end token "⟨eos⟩". The conditional probability of the whole output sequence $p(y|x)$ is progressively combined by the probability of each step $p(y_i|y_{<i}, x)$:

$$p(y|x) = \prod_{i}^{|y|} p(y_i|y_{<i}, x) \quad (3)$$

where $y_{<i} = y_1...y_{i-1}$, and $p(y_i|y_{<i}, x)$ is the probability over the target vocabulary $\mathcal{V}$ normalized by softmax$(\cdot)$.

Because all tokens in linearized event representations are also natural language words, we adopt the pre-trained language model T5 (Raffel et al., 2020) as our transformer-based encoder-decoder architecture. In this way, the general text generation knowledge can be directly reused.

2.3 Constrained Decoding

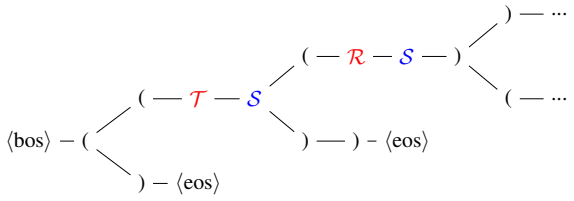
Given the hidden sequence $\mathbf{H}$, the sequence-to-structure network needs to generate the linearized event representations token-by-token. One straightforward solution is to use a greedy decoding algorithm, which selects the token with the highest predicted probability $p(y_i|y_{<i}, x)$ at each decoding step i. Unfortunately, this greedy decoding algorithm cannot guarantee the generation of valid event structures. In other words, it could end up with invalid event types, mismatch of argument-type, and incomplete structure. Furthermore, the greedy decoding algorithm ignores the useful event schema knowledge, which can be used to guide the decoding effectively. For example, we can constrain the model to only generate event type tokens in the type position.

To exploit the event schema knowledge, we propose to employ a trie-based constrained decoding algorithm (Chen et al., 2020a; Cao et al., 2021) for event generation. During constrained decoding, the event schema knowledge is injected as the prompt of the decoder and ensures the generation of valid event structures.

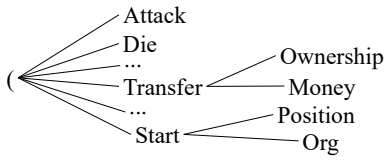
Concretely, unlike the greedy decoding algorithm that selects the token from the whole target vocabulary $\mathcal{V}$ at each step, our trie-based constrained decoding method dynamically chooses and prunes a candidate vocabulary $\mathcal{V}'$ based on the current generated state. A complete linearized form decoding process can be represented by executing a trie tree search, as shown in Figure 3a. Specifically, each generation step of TEXT2EVENT has three kinds of candidate vocabulary $\mathcal{V}'$:

- Event schema: label names of event types $\mathcal{T}$ and argument roles $\mathcal{R}$;
- Mention strings: event trigger word and argument mention $\mathcal{S}$, which is the text span in the raw input;
- Structure indicator: "(" and ")" which are used to combine event schemas and mention strings.

The decoding starts from the root "⟨bos⟩" and ends at the terminator "⟨eos⟩". At the generation step i, the candidate vocabulary $\mathcal{V}'$ is the children nodes of the last generated node. For instance, at the generation step with the generated string "⟨bos⟩ (", the candidate vocabulary $\mathcal{V}'$ is {"(", ")"} in Figure 3a. When generating the event type name



(a) The trie of event structure.



(b) The trie of event type $\mathcal{T}$.

Figure 3: The prefix tree (trie) of the constrained decoding algorithm for controllable structure generation. $\mathcal{T}$ and $\mathcal{R}$ indicate the label name of event type and argument role. $\mathcal{S}$ indicates the text span in the raw text, which is the event trigger or argument mention of the extracted event.

$\mathcal{T}$, argument role name $\mathcal{R}$ and text span $\mathcal{S}$, the decoding process can be considered as executing search on a subtree of the trie tree. For example, in Figure 3b, the candidate vocabulary $\mathcal{V}'$ for "(Transfer" is {"Ownership", "Money"}.

Finally, the decoder's output will be transformed to event records and used as final extraction results.

3 Learning

This section describes how to learn the TEXT2EVENT neural network in an end-to-end manner. Our method can be learned using only the coarse parallel text-record annotations, i.e., pairs of ⟨sentence, event records⟩, with no need for fine-grained token-level annotation used in traditional methods. Given a training dataset $\mathcal{D} = \{(x_1, y_1), ...(x_{|\mathcal{D}|}, y_{|\mathcal{D}|})\}$ where each instance is a ⟨sentence, event records⟩ pair, the learning objective is the negative log-likelihood function as:

$$\mathcal{L} = -\sum_{(x,y)\in\mathcal{D}} \log p(y|x, \theta) \quad (4)$$

where θ is model parameters.

Unfortunately, unlike general text-to-text generation models, the learning of sequence-to-structure generation models is more challenging: 1) There is an output gap between the event generation model and the text-to-text generation model. Compared with natural word sequences, the linearized event structure contains many non-semantic indicators

such as "(" and ")", and they don't follow the syntax constraints of natural language sentences. 2) The non-semantic indicators "(" and ")" appear very frequently but contain little semantic information, which will mislead the learning process.

To address the above challenges, we employ a curriculum learning (Bengio et al., 2009; Xu et al., 2020) strategy. Specifically, we first train PLMs using simple event substructure generation tasks so that they would not overfit in non-semantic indicators; then we train the model on the full event structure generation task.

Substructure Learning. Because event representations often have complex structures and their token sequences are different from natural language word sequences, it is challenging to train them with the full sequence generation task directly. Therefore, we first train TEXT2EVENT on simple event substructures.

Specifically, we learn our model by starting from generating only "(label, span)" substructures, including "(type, trigger words)" and "(role, argument words)" substructures. For example, we will extract substructure tasks in Figure 2c in this stage as: `(Transport returned) (Artifact The man) (Arrest-Jail capture)`, etc. We construct a ⟨sentence, substructures⟩ pair for each extracted substructures, then train our model using the loss in equation 4.

Full Structure Learning. After the substructure learning stage, we further train our model for the full structure generation task using the loss in equation 4. We found the curriculum learning strategy uses data annotation more efficiently and makes the learning process more smooth.

4 Experiments

This section evaluates the proposed TEXT2EVENT model by conducting experiments in both supervised learning and transfer learning settings.

4.1 Experimental Settings

Datasets. We conducted experiments on the event extraction benchmark – ACE2005 (Walker et al., 2006), which has 599 English annotated documents and 33 event types. We used the same split and preprocessing step as the previous work (Zhang et al., 2019b; Wadden et al., 2019; Du and Cardie, 2020), and we denote it as ACE05-EN.

Dataset	Split	#Sents	#Events	#Roles
ACE05-EN	Train	17,172	4,202	4,859
	Dev	923	450	605
	Test	832	403	576
ACE05-EN$^+$	Train	19,216	4,419	6,607
	Dev	901	468	759
	Test	676	424	689
ERE-EN	Train	14,736	6,208	8,924
	Dev	1,209	525	730
	Test	1,163	551	822

Table 1: Dataset statistics.

In addition to ACE05-EN, we also conducted experiments on two other benchmarks: ACE05-EN$^+$ and ERE-EN, using the same split and preprocessing step in the previous work (Lin et al., 2020). Compared to ACE05-EN, ACE05-EN$^+$ and ERE-EN further consider pronoun roles and multi-token event triggers. ERE-EN contains 38 event categories and 458 documents.

Statistics of all datasets are shown in Table 1.

For evaluation, we used the same criteria in previous work (Zhang et al., 2019b; Wadden et al., 2019; Lin et al., 2020). Since TEXT2EVENT is a text generation model, we reconstructed the offset of predicted trigger mentions by finding the matched utterance in the input sequence one by one. For argument mentions, we found the nearest matched utterance to the predicted trigger mention as the predicted offset.

Baselines. Currently, event extraction supervision can be conducted at two different levels: 1) *Token-level annotation*, which labels each token in a sentence with event labels, e.g., "The/O dismission/B-End-Position of/O ..."; 2) *Parallel text-record annotation*, which only gives ⟨sentence, event⟩ pairs but without expensive token-level annotations, e.g., ⟨The dismission of ..., {Type: End-Position, Trigger: dismission, ...}⟩. Furthermore, some previous works also leverage golden entity annotation for model training, which marks all entity mentions with their golden types, to facilitate event extraction. Introducing more supervision knowledge will benefit the event extraction but is more label-intensive. The proposed Text2Event only uses parallel text-record annotation, which makes it more practical in a real-world application.

To verify TEXT2EVENT, we compare our method with the following groups of baselines:

1. Baselines using token annotation: *TANL* is the

Models	Trig-C			Arg-C			PLM
	P	R	F1	P	R	F1	
Models using Token Annotation + Entity Annotation							
Joint3EE (Nguyen and Nguyen, 2019)	68.0	71.8	69.8	52.1	52.1	52.1	-
DYGIE++ (Wadden et al., 2019)	-	-	69.7	-	-	48.8	BERT-large
GAIL (Zhang et al., 2019b)	74.8	69.4	72.0	61.6	45.7	52.4	ELMo
$\text{OneIE}_{\text{w/o Global}}$ (Lin et al., 2020)	-	-	73.5	-	-	53.9	BERT-large
OneIE (Lin et al., 2020)	-	-	74.7	-	-	56.8	BERT-large
Models using Token Annotation							
EEQA (Du and Cardie, 2020)	71.1	73.7	72.4	56.8	50.2	53.3	2×BERT-base
MQAEE (Li et al., 2020)	-	-	71.7	-	-	53.4	3×BERT-large
Generation-based Baselines using Token Annotation							
TANL (Paolini et al., 2021)	-	-	68.4	-	-	47.6	T5-base
Multi-Task TANL (Paolini et al., 2021)	-	-	68.5	-	-	48.5	T5-base
Our Model using Parallel Text-Record Annotation							
TEXT2EVENT	67.5	71.2	69.2	46.7	53.4	49.8	T5-base
TEXT2EVENT	69.6	74.4	71.9	52.5	55.2	53.8	T5-large

Table 2: Experiment results on ACE05-EN. Trig-C indicates trigger identification and classification. Arg-C indicates argument identification and classification. PLM represents the pre-trained language models used by each model.

SOTA sequence generation-based method that models event extraction as a trigger-argument pipeline manner (Paolini et al., 2021); *Multi-task TANL* extends *TANL* by transferring structure knowledge from other tasks; *EEQA* (Du and Cardie, 2020) and *MQAEE* (Li et al., 2020) are QA-based models which use machine reading comprehension model for trigger detection and argument extraction.

2. Baselines using both token annotation and entity annotation: *Joint3EE* is a joint entity, trigger, argument extraction model based on the shared hidden representations (Nguyen and Nguyen, 2019); *DYGIE++* is a BERT-based model which captures both within-sentence and cross-sentence context (Wadden et al., 2019); *GAIL* is an inverse reinforcement learning-based joint entity and event extraction model (Zhang et al., 2019b); *OneIE* is an end-to-end IE system which employs global feature and beam search to extract globally optimal event structures (Lin et al., 2020).

Implementations. We optimized our model using label smoothing (Szegedy et al., 2016; Müller et al., 2019) and AdamW (Loshchilov and Hutter, 2019) with learning rate=5e-5 for T5-large, 1e-4 for T5-base. For curriculum learning, the epoch of substructure learning is 5, and full structure learning is 30. We conducted each experiment on a single NVIDIA GeForce RTX 3090 24GB. Due to GPU memory limitation, we used different batch sizes for different models: 8 for T5-large and 16 for T5-base; and truncated the max length of raw text to 256 and linearized form to 128 during training. We added the task name as the prefix for the T5 default setup.

4.2 Results in Supervised Learning Setting

Table 2 presents the performance of all baselines and TEXT2EVENT on ACE05-EN. And Table 3 shows the performance of SOTA and TEXT2EVENT on ACE05-EN$^+$ and ERE-EN. We can see that:

1) *By uniformly modeling all tasks in a single model and predicting labels universally,* TEXT2EVENT *can achieve competitive performance with weaker supervision and simpler architecture.* Our method, only using the weak parallel text-record annotations, surpasses most of the baselines using token and entity annotations and achieves competitive performance with SOTA. Furthermore, using the simple encoder-decoder architecture, TEXT2EVENT outperforms most of the counterparts with complicated architectures.

Datasets	Trig-C			Arg-C		
	P	R	F1	P	R	F1
SOTA (Token + Entity Annotation)						
ACE05-EN$^+$	-	-	72.8	-	-	54.8
ERE-EN*	56.9	58.7	57.8	51.9	47.8	49.8
TEXT2EVENT (Parallel Text-Record Annotation)						
ACE05-EN$^+$	71.2	72.5	71.8	54.0	54.8	54.4
ERE-EN	59.2	59.6	59.4	49.4	47.2	48.3

Table 3: Experiment results on ACE05-EN$^+$ and ERE-EN. SOTA indicates the state-of-the-art system – OneIE. * The result of SOTA for ERE-EN is reproduced by the official release code because of the slightly different dataset statistic result on ERE-EN.

2) *By directly generating event structure from the text,* TEXT2EVENT *can significantly outperform sequence generation-based methods.* Our method improves Arg-C F1 by 4.6% and 2.7% over the SOTA generation baseline and its extended multi-task TANL. Compared with sequence generation, structure generation can be effectively guided using event schema knowledge during inference, and there is no need to generate irrelevant information.

3) *By uniformly modeling and sharing information between different tasks and labels, the sequence-to-structure framework can achieve robust performance.* From Table 2 and Table 3, we can see that the performance of OneIE decreases on the harder dataset ACE05-EN$^+$, which has more pronoun roles and multi-token triggers. By contrast, the performance of TEXT2EVENT remains nearly the same on ACE05-EN. We believe this may be because the proposed sequence-to-structure model is a universal model that doesn't specialize in labels and can better share information between different labels.

4.3 Results in Transfer Learning Setting

TEXT2EVENT is a universal model, therefore can facilitate the knowledge transfer between different labels. To verify the transfer ability of TEXT2EVENT, we conducted experiments in the transfer learning setting, and the results are shown in Table 4. Specifically, we first randomly split the sentences which length larger than 8 in ACE05-EN$^+$ into two equal-sized subsets *src* and *tgt*: *src* only retains the annotations of the top 10 frequent event types, and *tgt* only retains the annotations of the remaining 23 event types. For both *src* and *tgt*, we use 80% of the dataset for model training and 20% for evaluation. For transfer learning, We first pre-trained an event extraction model on the *src* dataset, then fine-tuned the pre-trained model for extracting the new event types in *tgt*. From Table 4, we can see that:

Settings	Trig-C			Arg-C		
	P	R	F1	P	R	F1
OneIE (Token + Entity Annotation)						
Non-transfer	78.1	62.3	69.3	50.9	37.9	43.5
Transfer	78.9	61.7	69.2	57.1	40.0	47.0
Gain			-0.1			+3.5
EEQA (Token Annotation)						
Non-transfer	69.9	67.3	68.6	36.5	37.4	36.9
Transfer	79.5	61.7	69.5	33.9	41.2	37.2
Gain			+0.9			+0.3
TEXT2EVENT (Parallel Text-Record Annotation)						
Non-transfer	79.4	61.1	69.0	58.4	40.9	48.0
Transfer	82.1	65.3	72.7	58.8	45.4	51.2
Gain			+3.7			+3.2

Table 4: Experiment results on the *tgt* subset of ACE05-EN$^+$ in the transfer learning setting.

1) *Data-efficient* TEXT2EVENT *can make better use of supervision signals.* Even training on *tgt* from scratch, the proposed method also outperforms strong baselines. We believe that this may because baselines using token and entity annotation require massive fine-grained data for model learning. Different from baselines, TEXT2EVENT uniformly models all subtasks, thus the knowledge can be seamlessly transferred, which is more data-efficient.

2) TEXT2EVENT *can effectively transfer knowledge between different labels.* Compared with the non-transfer setting, which is directly trained on *tgt* training set, the transfer setting of TEXT2EVENT can achieve significant F1 improvements of 3.7 and 3.2 on Trig-C and Arg-C, respectively. By contrast, the other two baselines cannot obtain significant F1 improvements of both Trig-C and Arg-C via transfer learning. Note that the information of entity annotation is shared across *src* and *tgt*. As a result, OneIE can leverage such information to better argument prediction even with worse trigger prediction. However, even without using entity annotation, the proposed method can still achieve a similar improvement in the transfer learning setting. This is because the labels are provided universally in TEXT2EVENT, so the parameters are not label-specific.

Trig-C F1	1%	5%	25%	100%
TEXT2EVENT + CL	24.6	52.8	65.5	71.4
TEXT2EVENT	17.9	52.1	65.0	69.6
w/o CD	13.2	46.8	64.3	68.6
w/o ES	0.0	24.3	31.6	55.5
Arg-C F1	1%	5%	25%	100%
TEXT2EVENT + CL	8.6	33.6	44.0	53.3
TEXT2EVENT	3.7	30.9	44.7	52.6
w/o CD	2.3	27.3	44.4	52.3
w/o ES	0.0	7.0	8.2	28.9

Table 5: Experiment results of variants trained with different-sized training set on the development set of ACE05-EN.

4.4 Detailed Analysis

This section analyzes the effects of event schema knowledge, constrained decoding, and curriculum learning algorithm in TEXT2EVENT. We designed four ablated variants based on T5-base:

- "TEXT2EVENT" is the base model that is directly trained with the full structure learning.
- "+ CL" indicates training TEXT2EVENT with the proposed curriculum learning algorithm.
- "w/o CD" discards the constrained decoding during inference and generates event structures as an unconstrained generation model.
- "w/o ES" replaces the names of event types and roles with meaningless symbols, which is used to verify the effect of event schema knowledge.

Table 5 shows the results on the development set of ACE05-EN using different training data sizes. We can see that: 1) *Constrained decoding can effectively guide the generation with event schemas, especially in low-resource settings.* Comparing to "w/o CD", constrained decoding improves the performance of TEXT2EVENT, especially in low-resource scenarios, e.g., using 1%, 5% training set. 2) *Curriculum learning is useful for model learning.* Substructure learning improves 4.7% Trig-C F1 and 5.8% Arg-C F1 on average. 3) *It is crucial to encode and generate event labels as words, rather than meaningless symbols.* Because by encoding labels as natural language words, our method can effectively transfer knowledge from pre-trained language models.

5 Related Work

Our work is a synthesis of two research directions: event extraction and structure prediction via neural generation model.

Event extraction has received widespread attention in recent years, and mainstream methods usually use different strategies to obtain a complete event structure. These methods can be divided into: 1) pipeline classification (Ahn, 2006; Ji and Grishman, 2008; Liao and Grishman, 2010; Hong et al., 2011, 2018; Huang and Riloff, 2012; Chen et al., 2015; Sha et al., 2016; Lin et al., 2018; Yang et al., 2019; Wang et al., 2019; Ma et al., 2020; Zhang et al., 2020c), 2) multi-task joint models (McClosky et al., 2011; Li et al., 2013, 2014; Yang and Mitchell, 2016; Nguyen et al., 2016; Liu et al., 2018; Zhang et al., 2019a; Zheng et al., 2019), 3) semantic structure grounding (Huang et al., 2016, 2018; Zhang et al., 2020a), and 4) question-answering (Chen et al., 2020b; Du and Cardie, 2020; Li et al., 2020; Liu et al., 2020).

Compared with previous methods, we model all subtasks of event extraction in a uniform sequence-to-structure framework, which leads to better decision interactions and information sharing. The neural encoder-decoder generation architecture (Sutskever et al., 2014; Bahdanau et al., 2015) has shown its strong structure prediction ability and has been widely used in many NLP tasks, such as machine translation (Kalchbrenner and Blunsom, 2013), semantic parsing (Dong and Lapata, 2016; Song et al., 2020), entity extraction (Straková et al., 2019), relation extraction (Zeng et al., 2018; Zhang et al., 2020b), and aspect term extraction (Ma et al., 2019). Like TEXT2EVENT in this paper, TANL (Paolini et al., 2021) and GRIT (Du et al., 2021) also employ neural generation models for event extraction, but they focus on sequence generation, rather than structure generation. Different from previous works that extract text span via labeling (Straková et al., 2019) or copy/pointer mechanism (Zeng et al., 2018; Du et al., 2021), TEXT2EVENT directly generate event schemas and text spans to form event records via constrained decoding (Cao et al., 2021; Chen et al., 2020a), which allows TEXT2EVENT to handle various event types and transfer to new types easily.

6 Conclusions

In this paper, we propose TEXT2EVENT, a sequence-to-structure generation paradigm for

event extraction. TEXT2EVENT directly learns from parallel text-record annotation and uniformly models all subtasks of event extraction in a sequence-to-structure framework. Concretely, we propose an effective sequence-to-structure network for event extraction, which is further enhanced by a constrained decoding algorithm for event knowledge injection during inference and a curriculum learning algorithm for efficient model learning. Experimental results in supervised learning and transfer learning settings show that TEXT2EVENT can achieve competitive performance with the previous SOTA using only coarse text-record annotation.

For future work, we plan to adapt our method to other information extraction tasks, such as N-ary relation extraction.

Acknowledgments

We sincerely thank the reviewers for their insightful comments and valuable suggestions. This work is supported by the National Natural Science Foundation of China under Grants no. U1936207 and 61772505, Beijing Academy of Artificial Intelligence (BAAI2019QN0502), and in part by the Youth Innovation Promotion Association CAS(2018141).

References

David Ahn. 2006. The stages of event extraction. In *Proceedings of the Workshop on Annotating and Reasoning about Time and Events*, pages 1–8, Sydney, Australia. Association for Computational Linguistics.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *The Third International Conference on Learning Representations*.

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. 2009. Curriculum learning. In *Proceedings of the 26th International Conference on Machine Learning*, pages 41–48, Montreal. Omnipress.

Nicola De Cao, Gautier Izacard, Sebastian Riedel, and Fabio Petroni. 2021. Autoregressive entity retrieval. In *International Conference on Learning Representations*.

Pinzhen Chen, Nikolay Bogoychev, Kenneth Heafield, and Faheem Kirefu. 2020a. Parallel sentence mining by constrained decoding. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1672–1678, Online. Association for Computational Linguistics.

Yubo Chen, Liheng Xu, Kang Liu, Daojian Zeng, and Jun Zhao. 2015. Event extraction via dynamic multi-pooling convolutional neural networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 167–176, Beijing, China. Association for Computational Linguistics.

Yunmo Chen, Tongfei Chen, Seth Ebner, Aaron Steven White, and Benjamin Van Durme. 2020b. Reading the manual: Event extraction as definition comprehension. In *Proceedings of the Fourth Workshop on Structured Prediction for NLP*, pages 74–83, Online. Association for Computational Linguistics.

George Doddington, Alexis Mitchell, Mark Przybocki, Lance Ramshaw, Stephanie Strassel, and Ralph Weischedel. 2004. The automatic content extraction (ACE) program – tasks, data, and evaluation. In *Proceedings of the Fourth International Conference on Language Resources and Evaluation (LREC'04)*, Lisbon, Portugal. European Language Resources Association (ELRA).

Li Dong and Mirella Lapata. 2016. Language to logical form with neural attention. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 33–43, Berlin, Germany. Association for Computational Linguistics.

Xinya Du and Claire Cardie. 2020. Event extraction by answering (almost) natural questions. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 671–683, Online. Association for Computational Linguistics.

Xinya Du, Alexander Rush, and Claire Cardie. 2021. GRIT: Generative role-filler transformers for document-level event entity extraction. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 634–644, Online. Association for Computational Linguistics.

Yu Hong, Jianfeng Zhang, Bin Ma, Jianmin Yao, Guodong Zhou, and Qiaoming Zhu. 2011. Using cross-entity inference to improve event extraction. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 1127–1136, Portland, Oregon, USA. Association for Computational Linguistics.

Yu Hong, Wenxuan Zhou, Jingli Zhang, Guodong Zhou, and Qiaoming Zhu. 2018. Self-regulation: Employing a generative adversarial network to improve event detection. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 515–526, Melbourne, Australia. Association for Computational Linguistics.

Lifu Huang, Taylor Cassidy, Xiaocheng Feng, Heng Ji, Clare R. Voss, Jiawei Han, and Avirup Sil. 2016.

Liberal event extraction and event schema induction. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 258–268, Berlin, Germany. Association for Computational Linguistics.

Lifu Huang, Heng Ji, Kyunghyun Cho, Ido Dagan, Sebastian Riedel, and Clare Voss. 2018. Zero-shot transfer learning for event extraction. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2160–2170, Melbourne, Australia. Association for Computational Linguistics.

Ruihong Huang and Ellen Riloff. 2012. Modeling textual cohesion for event extraction. In *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence*, AAAI'12, page 1664–1670. AAAI Press.

Heng Ji and Ralph Grishman. 2008. Refining event extraction through cross-document inference. In *Proceedings of ACL-08: HLT*, pages 254–262. Association for Computational Linguistics.

Nal Kalchbrenner and Phil Blunsom. 2013. Recurrent continuous translation models. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1700–1709, Seattle, Washington, USA. Association for Computational Linguistics.

Fayuan Li, Weihua Peng, Yuguang Chen, Quan Wang, Lu Pan, Yajuan Lyu, and Yong Zhu. 2020. Event extraction as multi-turn question answering. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 829–838, Online. Association for Computational Linguistics.

Qi Li, Heng Ji, Yu Hong, and Sujian Li. 2014. Constructing information networks using one single model. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1846–1851, Doha, Qatar. Association for Computational Linguistics.

Qi Li, Heng Ji, and Liang Huang. 2013. Joint event extraction via structured prediction with global features. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 73–82, Sofia, Bulgaria. Association for Computational Linguistics.

Shasha Liao and Ralph Grishman. 2010. Using document level cross-event inference to improve event extraction. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 789–797, Uppsala, Sweden. Association for Computational Linguistics.

Hongyu Lin, Yaojie Lu, Xianpei Han, and Le Sun. 2018. Nugget proposal networks for Chinese event detection. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1565–1574, Melbourne, Australia. Association for Computational Linguistics.

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, Online. Association for Computational Linguistics.

Jian Liu, Yubo Chen, Kang Liu, Wei Bi, and Xiaojiang Liu. 2020. Event extraction as machine reading comprehension. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1641–1651, Online. Association for Computational Linguistics.

Xiao Liu, Zhunchen Luo, and Heyan Huang. 2018. Jointly multiple events extraction via attention-based graph information aggregation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1247–1256, Brussels, Belgium. Association for Computational Linguistics.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. In *Seventh International Conference on Learning Representations*.

Dehong Ma, Sujian Li, Fangzhao Wu, Xing Xie, and Houfeng Wang. 2019. Exploring sequence-to-sequence learning in aspect term extraction. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3538–3547, Florence, Italy. Association for Computational Linguistics.

Jie Ma, Shuai Wang, Rishita Anubhai, Miguel Ballesteros, and Yaser Al-Onaizan. 2020. Resource-enhanced neural model for event argument extraction. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 3554–3559, Online. Association for Computational Linguistics.

David McClosky, Mihai Surdeanu, and Christopher Manning. 2011. Event extraction as dependency parsing. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 1626–1635, Portland, Oregon, USA. Association for Computational Linguistics.

Rafael Müller, Simon Kornblith, and Geoffrey E Hinton. 2019. When does label smoothing help? In *Advances in Neural Information Processing Systems*, volume 32, pages 4694–4703. Curran Associates, Inc.

Thien Huu Nguyen, Kyunghyun Cho, and Ralph Grishman. 2016. Joint event extraction via recurrent neural networks. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 300–309, San Diego, California. Association for Computational Linguistics.

Trung Minh Nguyen and Thien Huu Nguyen. 2019. One for all: Neural joint modeling of entities and events. In *The Thirty-Third AAAI Conference on*

Artificial Intelligence, AAAI '2019. Association for the Advancement of Artificial Intelligence.

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. In *The Ninth International Conference on Learning Representations*.

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.

Lei Sha, Jing Liu, Chin-Yew Lin, Sujian Li, Baobao Chang, and Zhifang Sui. 2016. RBPB: Regularization-based pattern balancing method for event extraction. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1224–1234, Berlin, Germany. Association for Computational Linguistics.

Linfeng Song, Ante Wang, Jinsong Su, Yue Zhang, Kun Xu, Yubin Ge, and Dong Yu. 2020. Structural information preserving for graph-to-text generation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7987–7998, Online. Association for Computational Linguistics.

Jana Straková, Milan Straka, and Jan Hajic. 2019. Neural architectures for nested NER through linearization. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5326–5331, Florence, Italy. Association for Computational Linguistics.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems*, volume 27, pages 3104–3112. Curran Associates, Inc.

C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. 2016. Rethinking the inception architecture for computer vision. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2818–2826.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30, pages 5998–6008. Curran Associates, Inc.

Oriol Vinyals, Ł ukasz Kaiser, Terry Koo, Slav Petrov, Ilya Sutskever, and Geoffrey Hinton. 2015. Grammar as a foreign language. In *Advances in Neural Information Processing Systems*, volume 28, pages 2773–2781. Curran Associates, Inc.

David Wadden, Ulme Wennberg, Yi Luan, and Hannaneh Hajishirzi. 2019. Entity, relation, and event extraction with contextualized span representations. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5784–5789, Hong Kong, China. Association for Computational Linguistics.

Christopher Walker, Stephanie Strassel, Julie Medero, and Kazuaki Maeda. 2006. Ace 2005 multilingual training corpus.

Xiaozhi Wang, Ziqi Wang, Xu Han, Zhiyuan Liu, Juanzi Li, Peng Li, Maosong Sun, Jie Zhou, and Xiang Ren. 2019. HMEAE: Hierarchical modular event argument extraction. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5777–5783, Hong Kong, China. Association for Computational Linguistics.

Benfeng Xu, Licheng Zhang, Zhendong Mao, Quan Wang, Hongtao Xie, and Yongdong Zhang. 2020. Curriculum learning for natural language understanding. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6095–6104, Online. Association for Computational Linguistics.

Bishan Yang and Tom M. Mitchell. 2016. Joint extraction of events and entities within a document context. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 289–299, San Diego, California. Association for Computational Linguistics.

Sen Yang, Dawei Feng, Linbo Qiao, Zhigang Kan, and Dongsheng Li. 2019. Exploring pre-trained language models for event extraction and generation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5284–5294, Florence, Italy. Association for Computational Linguistics.

Xiangrong Zeng, Daojian Zeng, Shizhu He, Kang Liu, and Jun Zhao. 2018. Extracting relational facts by an end-to-end neural model with copy mechanism. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 506–514, Melbourne, Australia. Association for Computational Linguistics.

Hongming Zhang, Haoyu Wang, and Dan Roth. 2020a. Unsupervised label-aware event trigger and argument classification.

Junchi Zhang, Yanxia Qin, Yue Zhang, Mengchi Liu, and Donghong Ji. 2019a. Extracting entities and events as a single task using a transition-based neural model. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence,*

IJCAI-19, pages 5422–5428. International Joint Conferences on Artificial Intelligence Organization.

Ranran Haoran Zhang, Qianying Liu, Aysa Xuemo Fan, Heng Ji, Daojian Zeng, Fei Cheng, Daisuke Kawahara, and Sadao Kurohashi. 2020b. Minimize exposure bias of Seq2Seq models in joint entity and relation extraction. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 236–246, Online. Association for Computational Linguistics.

Tongtao Zhang, Heng Ji, and Avirup Sil. 2019b. 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 Hovy. 2020c. A two-step approach for implicit event argument detection. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7479–7485, Online. Association for Computational Linguistics.

Shun Zheng, Wei Cao, Wei Xu, and Jiang Bian. 2019. Doc2EDAG: An end-to-end document-level framework for Chinese financial event extraction. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 337–346, Hong Kong, China. Association for Computational Linguistics.