Seq2EG: A Novel and Effective Event Graph Parsing Approach for Event Extraction

Junsheng Zhou (zhoujs@njnu.edu.cn)
Nanjing Normal University
Haotong Sun
Nanjing Normal University
Li Kong
Nanjing Normal University
Yanhui Gu
Nanjing Normal University
Weiguang Qu
Nanjing Normal University

Research Article

Keywords: Event extraction , Event detection , Argument extraction , Graph parsing · Seq2seq

Posted Date: April 7th, 2022

DOI: https://doi.org/10.21203/rs.3.rs-1523905/v1

Noname manuscript No.
(will be inserted by the editor)

Seq2EG: A Novel and Effective Event Graph Parsing Approach for Event Extraction

Haotong Sun · Junsheng Zhou · Li Kong · Yanhui Gu · Weiguang Qu

Received: date / Accepted: date

Abstract Event extraction is a fundamental task in information extraction. Most previous approaches typically transform event extraction into two subtasks: trigger classification and argument classification, and solve them via classification-based methods, which suffer from some inherent drawbacks. To overcome these issues, in this paper we propose a novel event extraction model Seq2EG by first formulating event extraction as an event graph parsing problem, and then exploiting a pre-trained sequence-to-sequence (seq2seq) model to transduce an input sentence into an accurate event graph without the need for trigger words. Based on the generative event graph parsing formulation, our model Seq2EG can explicitly model the multiple event correlations and argument sharing, and can naturally incorporate some graph-structured features and the rich semantic information conveyed by the labels of event types and argument roles. Extensive experimental results on the public ACE2005 dataset show that, our approach outperforms all previous state-of-the-art models for event extraction by a large margin, respectively obtaining an improvement of 3.4% F1 score for event detection and an improvement of 4.7% F1 score for argument classification over the best baselines.

Keywords Event extraction · Event detection · Argument extraction · Graph parsing · Seq2seq

1 Introduction

Event Extraction (EE) is an essential and challenging Information Extraction (IE) task for natural language understanding. The event extraction task has been shown

Haotong Sun, E-mail: sunht@nnu.edu.cn
Junsheng Zhou (corresponding author), E-mail: zhoujs@njnu.edu.cn
Li Kong, E-mail: kl_nju@126.com
Yanhui Gu, E-mail: gu@njnu.edu.cn
Weiguang Qu, E-mail: wgqu@njnu.edu.cn
School of Computer and Electronic Information, Nanjing Normal University, China

beneficial to a wide range of downstream tasks, such as document summarization, question answering and so on [1; 2]. Technically speaking, as defined by the ACE 2005 dataset, a benchmark for event extraction [3], the event extraction task can be divided into two subtasks, i.e., event detection (identifying instances of specified types of events) and argument extraction (identifying arguments of each event type and labeling their roles). For example, an event extraction instance is shown in **Fig. 1**.

EE is an actively studied task in IE where deep learning models have been the dominant approach to deliver the state-of-the-art performance. Nevertheless, most previous work typically treat EE as a classification problem. Specifically, most existing approaches generally transform the event extraction task into two subtasks: trigger classification and argument classification, and then perform the two subtasks in a joint fashion or a pipelined fashion [4; 5; 6; 7; 8]. Some recent works focus on use syntactic dependency structure or external knowledge to boost the classification performance [9; 10; 11; 12]. More recently, Li *et al*. [13] proposes to first perform the trigger classification and then to reformulate argument extraction as a Machine Reading Comprehension (MRC) task to utilize sophisticated MRC methods and large annotated external MRC data. Methodologically speaking, the existing event extrac-

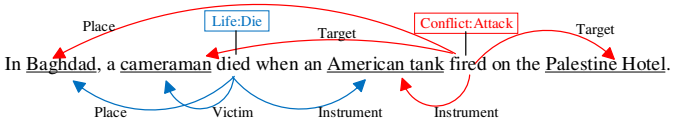

Fig. 1: An example of event extraction. In this sentence, two different events are expressed, which are denoted in red and in blue respectively.

tion approaches suffer from the following inherent drawbacks:

Firstly, most previous approaches depend heavily on the trigger word. On the one hand, triggers are nonessential to event detection and event extraction; on the other hand, the identification and classification of trigger words may, to some extent, impede the accurate recognition of the events, due to the fact that some events may be expressed by multiple discontinuous words or phrases in one sentence (See more illustrations in **Section 3.2**). Particularly, the trigger-based models are prone to suffer from the long tail issue [14]. Literatures available show that, Liu *et al*. [15] is the only work for event detection without using trigger words, by simply casting event detection as a multi-label classification problem for input sentences, which cannot address the inherent issues with the trigger-based approaches, as illustrated below.

Secondly, current EE models do not exhibit good solutions to explicitly modeling the correlations between multiple events in one sentence and multiple arguments of different roles, and the event argument sharing issue. Though some existing works have investigated the multiple events phenomenon [10; 7]. These approaches explore to aggregate more contextual information from surrounding trigger candidates to generate a powerful representing vector for current candidate trigger by employing a self-attention mechanism or a hierarchical tagging scheme, then respectively predict

the trigger label [10; 7]. However, note that, modeling the associations between triggers is not equivalent to modeling the correlations between events. That is to say, the existing models cannot explicitly model the correlations between multiple events and multiple arguments.

Lastly, the existing approaches cannot leverage the semantic information of the labels of event type and argument role. As a matter of fact, both of them are informative and conducive to event extraction. However, such rich semantic information is neglected by the existing approaches.

To address these issues stated above simultaneously, we take a fresh look at event extraction and formulate it as a graph parsing problem. By regarding the multiple events expressed by one sentence as a whole, we argue that the goal of the EE task is to output an event graph, as shown in **Fig. 2**. On the one hand, the event graph is constructed to model the potential interactions between the multiple events; on the other hand, this graph parsing formulation can flexibly integrate some graph-structured features. Furthermore, we employ a pre-trained sequence-to-sequence model to generate the event graph, without the need for the identification of the trigger words. The experimental results demonstrate that our method substantially outperforms all previous state-of-the-art models on the public dataset ACE2005.

To sum up, this paper makes the following contributions:

- In this paper, we innovatively formulate the event extraction task as graph parsing, which delivers some typical benefits compared to the existing EE models. First, this graph parsing formulation can naturally model the correlations between multiple events in one sentence and the argument sharing; second, the event graph can be flexibly constructed to utilize more useful information, such as the semantic representations of the event type labels and argument role labels.
- We propose a transformer-based encoder-decoder model to derive the events from the global contextual information in the input sentences without relying on the trigger words. Furthermore, we propose some skillful strategies for the event graph linearization and an effective decoding algorithm to boost the generation performance.
- The extensive experiments over the public dataset ACE2005 demonstrate that the proposed simple approach outperforms the previous state-of-the-art models for event extraction by a large margin[1]. Particularly, our model does not use any syntactic dependency information and external knowledge.

This paper is a significant extension of our conference paper [16], which presents the first work to formulate event detection as a graph parsing task, and to introduce a novel generation-based method to predict event graph containing only event type nodes. In this paper, we further demonstrate the universality of the graph parsing framework by extending it to more complicated event extraction task, and propose some skillful strategies for the complete event graph linearization and an effective decoding algorithm to boost the generation performance.

The rest of this article is organized as follows. **Section 2** discusses the related work. **Section 3** describes the novel view of event extraction. **Section 4** presents the

[1] The source code will be publicly released upon acceptance.

detailed event parsing method via a seq2graph transducer. **Section 5** and **Section 6** describe the experiment settings and report the experimental results and model analysis. In **section 7** we summarize the proposed approach and describe future work.

2 Related Work

2.1 Event Extraction

In this paper, we focus on the event extraction task that includes two basic subtasks: event detection and argument extraction. Most recent works have focused on using neural networks in this task and have achieved significant progress. We roughly divide the recent approaches into three categories as following:

- **Sequence-based models:** This line of research operates on the word sequences using the deep neural networks. Chen *et al*. [4] devises a dynamic multi-pooling convolutional neural network to capture more information. Nguyen *et al*. [5] presents a joint model based on bidirectional RNN for event extraction. Sha *et al*. [6] adds dependency arcs with weight to BiLSTM to make use of tree structure and sequence structure simultaneously.
- **GCN-based models:** This line of research adopts the Graph Convolutional Network (GCN) over the dependency tree of a sentence to boost the performance. Nguyen *et al*. [9] is the first attempt to use GCN in ED. Liu *et al*. [10] employs a syntactic GCN and a self-attention mechanism to model multiple events extraction. Yan *et al*. [11] improves GCN by combining multiorder word representation from different GCN layers.
- **Machine Reading Comprehension (MRC)-based models:** Span-based MRC tasks involve extracting a span from a paragraph [17] or multiple paragraphs [18]. Du *et al*. [19] introduces a new paradigm for event extraction by formulating it as a question answering (QA) task. Liu *et al*. [20] and Li *et al*. [13] propose to first perform the trigger classification and then to reformulate argument extraction as a Machine Reading Comprehension (MRC) task to utilize sophisticated MRC methods and large annotated external MRC data.

Unlike the existing EE models based on trigger classification and argument classification, we formulate EE as a novel graph parsing problem, therefore it can explicitly model the multiple event correlations and incorporate some graph-structured features and the rich information regarding the event types and arguments.

Recently, Lu *et al*. [21] proposes a sequence-to-structure model Text2Event for EE, which can directly extract events from the text in an end-to-end manner. This work has the similar spirit with our work, and it is roughly orthogonal to our work in terms of time. However, different from the tree structure framework proposed in Lu *et al*. [21], we use a powerful event graph structure to model the correlations between events, which can provide a natural formulation to express event argument sharing relations between different event types in a sentence.

Additionally, slightly different from the aim of this paper, another recent line of research explores the joint entity recognition and event extraction [22; 19; 23].

2.2 Pre-trained Seq2seq Models

Pre-training a universal model and then fine-tuning the model on a downstream task have recently become a popular strategy in the field of natural language processing [24]. Recent studies also propose approaches to pre-training seq2seq models, such as MASS [25], PoDA [26], PEGASUS [27], BART [28], and T5 [29].

In this paper, our experiments only examine BART. We leave explorations of these models for future work.

3 A Novel View of Event Extraction

3.1 Task Description

Given a text document, an event extraction system should predict specified types of events mentioned in the input text and their arguments from each sentence. The most common used benchmark dataset in previous work is ACE 2005 corpus. The task defines 8 event types such as Life, Business and so on, and 33 subtypes such as Attack, End-Position, etc. **Table 1** summarizes relevant terminologies.

Table 1: The terminologies of event extraction task.

Entity mention	a reference to an entity, usually a noun phrase (NP)
Event trigger	main word which most clearly expresses an event occurrence
Event arguments	the entity mentions that are involved in an event
Argument roles	the relation of arguments to the event where they participate (35 total possible roles defined by ACE)
Event mention	a phrase or sentence within which an event is described including trigger and arguments

Following some previous work [8; 10; 20], we also assume that the golden-standard entity mentions are provided as the argument candidates to the event extraction systems.

3.2 Formulating EE as Event Graph Parsing

Traditionally, an event extraction system first recognizes a single word or phrase as the trigger in order to predict the event types of interest, and then identifies the event arguments for each derived event type [10; 7; 15]. However, as pointed out by Liu *et al*. [15], triggers are nonessential to event detection and event extraction. To a certain extent, the dependence on trigger may impede the accurate recognition of the events in the sentence.

In particular, some events may be triggered by multiple discontinuous words or phrases in one sentence, not by a single word or phrase. Take a concrete example in

ACE 2005 dataset to illustrate: *She lost her seat in the 1997 election*. In this sentence, an event type (Personnel:Elect) is mentioned, and its gold trigger was labelled as the word *lost*. In effect, to correctly recognize the event type (Personnel:Elect) from this sentence, we should comprehensively consider both the phrase *lost her seat* and the word *election* in the sentence (See more cases in **Section 6.5**). Therefore, it does not seem plausible that the problem of predicting an event from a whole sentence is reduced to the representation learning of the single trigger word for trigger classification or sequence labelling. Additionally, the trigger-based models are prone to suffer from the long tail issue, which makes supervised methods prone to overfitting and perform poorly on unseen/sparsely labeled triggers [14].

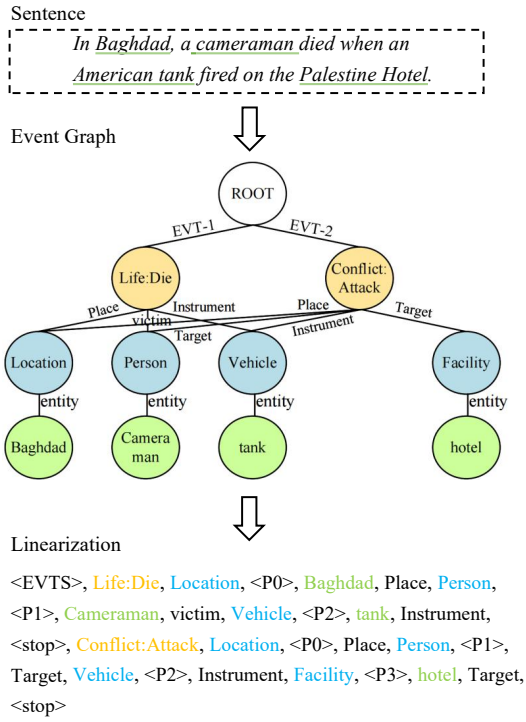

Fig. 2: An illustrative diagram of Seq2EG for the instance in Fig. 1.

In this paper, we look at the EE task from a new perspective. Given an input text, EE aims to recognize and predict the mentioned event types and their corresponding arguments. Intuitively, the multiple events derived from the same sentence should have a certain degree of correlations between them. Therefore, to model the correlations, we can view the multiple events expressed by the same sentence as a whole, by linking them together as a single graph, as shown in **Fig. 2**.

Specifically, we first introduce a special node as the root, and then attach each event type node as a child of the root; further, the multiple arguments of each specific

event type are linked as its children, with the edges being labeled as the argument roles. It is worth noting that the root of this event graph is not a virtual node. The root can take two possible values: *EVTS* and *NA*. While the input sentence does not contain any event, the root is assigned the value *NA*; otherwise it is assigned the value *EVTS*. Therefore, the prediction of root value is to judge whether the input sentence expresses some events or not. In addition to facilitating modeling multiple event correlations, our graph parsing formulation for EE also allows for the straightforward inclusion of other types of graph-structured features:

- First of all, our event graph can be flexibly constructed to exploit more useful information. For example, we know that each argument candidate is an entity mention which has a specific entity type such as person, location, vehicle and etc. This argument type is a critical feature in predicting the role of an argument candidate. It is common practice to employ an auxiliary feature embedding to encode the argument type for each argument candidate [10; 7]. In our event graph parsing formulation, in order to make full use of the argument type feature, we skillfully introduce a kind of argument type nodes in the event graph to represent the entity types of the argument nodes to be generated next, as shown in **Fig. 2**.
- Another important benefit of our event extraction paradigm is that it can provide a natural formulation to express event argument sharing relations between different event types in an event graph, which is the exact reason why the event graph constructed is a graph instead of a tree. For instance, the entity mention *Baghdad* is an argument of the event type *Life:Die*, and it is also an argument of another event type *Conflict:Attack*, as shown in **Fig. 2**.
- Additionally, our approach can effectively utilize the semantic representation of event type label and argument role label. Most previous classification-based approaches to EE generally view each event type or argument role as a specific class, omitting the semantic information conveyed by these type labels. In fact, the type label itself, such as *Divorce*, *Injure*, etc, is informative to the learning of EE models. In our graph parsing formulation, it is straightforward to incorporate the semantic representation of type label into the model. Specifically, during decoding, we can encode every previously generated node or edge with the corresponding type label embedding to assist the prediction of later nodes.

4 Event Graph Parsing via a Seq2Graph Transducer

Under our graph parsing formulation, the EE task is to transduce an input sentence into an event graph, as illustrated in Section 3. To achieve this, we choose to predict nodes and edges sequentially rather than simultaneously, because (1) we believe the previous node generation is informative to the current node generation; (2) variants of efficient sequence-to-sequence (seq2seq) models can be employed to model this process [30; 31; 28]. Theoretically, the advantages of applying a seq2seq model to event graph parsing are two-fold. First, there is no need to use trigger words for event detection. Second, when predicting next node during decoding, the global contextual information in the input sentence can be taken into consideration by the cross-attention mechanism between the decoder and encoder.

In this section, we first introduce our strategies for event graph linearization; next, the neural network model adopted for the seq2graph transduction and the decoding algorithm are illustrated respectively; lastly, a simple postprocessing procedure is illustrated.

4.1 The Linearization Strategies of Event Graph

While applying a seq2seq paradigm to event graph parsing, we first need to convert the event graph into a sequence of tokens by using linearization techniques. We do not particularly consider the order of events in an event graph, even for the special cases where there are two event nodes with the same type. Specifically, we employ a depth-first traversal (DFS) as it is quite closely related to the way natural language syntactic trees are linearized. While building the event graph for each sentence during the training phase, we simply append the event nodes to graph by the order of appearance in the ACE annotation. Additionally, when applying DFS to linearizing an event graph, we also simply traverse the graph in a natural order from left to right. However, different from the conventional graph traversal procedures, we particularly propose some effective strategies for the event graph linearization to boost the generation performance.

Firstly, to tackle the argument sharing problem in the event graph, we innovatively propose the use of special pointer symbols $<P_0>$, $<P_1>$, . . . , $<P_i>$ to represent argument nodes in the linearized sequence and to handle sharing arguments. Whenever such special symbols occur more than once it indicates that a specific argument node serves as multiple roles for multiple different events in an event graph. Our special symbols approach is used in combination with the graph traversal techniques, i.e. DFS.

Secondly, we introduce a flexible linearization ordering strategy while traversing the event graph. Generally speaking, the linearized sequence of a graph consists of the values of nodes and the labels of edges by the visiting order in the traversal procedure; that is to say, for a given event type node, the edge label (argument role) always comes in front of its child (argument type node). However, it is intuitively plausible that the argument role should be predicted after both the event type node and its child (argument type node) are generated. Therefore, we specially adjust the linearization order by postponing the output of argument role to the back of the argument type node and argument node in the linearized sequence of an event graph. For instance, for the example event graph shown in **Fig. 2**, the linearized representation generated by using standard DFS procedure is "*<EVTS>, Life:Die, Place, Location, $<P_0>$, Baghdad, victim, Person, $<P_1>$, Cameraman, Instrument, Vehicle, $<P_2>$, tank, <stop>, Conflict:Attack, Place, Location, $<P_0>$, Target, Person, $<P_1>$, Instrument, Vehicle, $<P_2>$, Target, Facility, $<P_3>$, hotel, <stop>*". As a contrast, the sequential representation generated by applying our linearization ordering strategy is also illustrated in **Fig. 2**. The importance of this linearization ordering strategy is also verified by the results of ablation experiments (see **Section 6.3**).

Lastly, we found that the labels of part of edges (e.g. EVT-1, entity, etc.) are not informative to event extraction in the preliminary experiments; we therefore omit these labels while linearizing the event graph.

4.2 The Transformer-based Generation Network

Let $x =< x_1, ..., x_n >$ be an input sentence and each x_i is a token in the sentence. Also, let $E =< e_1, ..., e_k >$ be the entity mentions in this sentence (k is the number of the entity mentions and can be zero). Each entity mention comes with the head and the entity type. Our approach sequentially decodes a list of tokens $y =< y_1, ..., y_m >$ where each y_i may be an event type, an event argument (i.e. an entity mention e_j), an argument role, an entity type, or a special symbol. When generating the argument nodes for a specific event type node, our model predicts the head of each argument as the argument output. Let Y be the output space. The transduction problem is to seek the most-likely sequence of nodes given x:

$$\begin{aligned} \hat{y} &= \arg\max_{y \in Y} p(y|x) \\ &= \arg\max_{y \in Y} \prod_{j}^{m} p(y_j|y_{<j}, x) \end{aligned} \tag{1}$$

To tackle the transduction problem, we adopt the transformer-based encoder-decoder architecture to generate the event graph [31]:

$$\{h_i\}_{i=1}^{n} = Encoder(\{x_i\}_{i=1}^{n}) \tag{2}$$

$$\{s_j\}_{j=1}^{m} = Decoder(\{y_{<j}\}_{j=1}^{m}, \{h_i\}_{i=1}^{n}) \tag{3}$$

$$P(y_j|x, y_{<j}) = softmax(g(s_j)) \tag{4}$$

At the encoding stage, we convert the input text into the hidden vector representation by employing a multi-layer transformer encoder with the multi-head attention mechanism. It is worth noting that our encoder just encodes the tokens in the input sentence without using any additional information, including the POS tags and the syntactic dependency structures. The decoder predicts the output sequence by following a similar scheme as the encoder, but including an encoder-decoder attention sublayer in between to deal with input-output alignment. The generated sequence starts from the special token "*BOS*" and ends with the special token "*EOS*".

In order to alleviate the data sparsity, we adopt the pre-trained language model BART as our transformer-based encoder-decoder architecture [28], so that we can exploit the model's latent knowledge (e.g., of semantics, linguistic relations, etc.) that has been captured through pre-training. The BART architecture can be viewed as a natural progression of "vanilla transformers" by Vaswani *et al*. [31], but with pre-training inspired by BERT's masked language model objective.

4.3 A Nested Constrained Beam Search Decoding Algorithm

For decoding in testing phase, it is a natural choice for our Seq2EG model to design a decoding algorithm based on beam search that generates the token sequence of an output event graph incrementally. However, while designing the beam search algorithm, we face two practical problems: 1) how to guarantee the generation of a valid event graph; 2) how to achieve fair and reasonable comparison when picking top-k best partial graphs among all the candidate items on each step of beam search.

For the first problem, it is relatively easy to tackle by incorporating the event schema knowledge into the search process to construct a constrained beam search algorithm. To be specific, at the different generation step during the search process, we can limit the candidate vocabulary for the choice of current item by referring to the knowledge of event schema. For example, if the current item to be predicted should be an event type name, we simply set the candidate vocabulary as the set of type names defined by the event schema. Relatively speaking, the second problem is more challenging. Unlike the target sentence generation task in traditional seq2seq models for machine translation where all elements in the target sequences are words, the elements in a linearized event graph sequence include many distinct types, such as the event type, argument type, argument role, entity mention, and some special symbols. Thus, at each timestep during beam search, the candidates in the beam may be the partial linearized event graph sequences ending with different types of elements, which may not be compared directly with each other. Therefore, the standard beam search algorithm cannot work well in this scenario. To address this issue, we propose an effective *nested beam search strategy* for the decoding. On the whole, the linearized sequence of an event graph consists of multiple events from a coarse-grained view; and at the fine-grained level, each event may contain a different number of arguments with different roles. Thus, to obtain fair comparison we introduce two types of beam-search in a single decoding process: *inter-event beam-search* and *intra-event beam-search*. In a nutshell, the inter-event beam-search is used to compete over complete event candidates; while extending an event to identify its type and its arguments, we switch on an *intra-event beam-search* to find top-k event structures. To facilitate the nested beam search process, we particularly use a special symbol <stop> to indicate the end of each event. Additionally, in the inner beam-search, an event with more arguments will result in lower score, we therefore normalize it by the number of arguments.

Based on the two considerations mentioned above, we design *a nested constrained decoding algorithm* to generate a valid and accurate event graph for the given input text. **Algorithm 1** shows the pseudocode for the complete procedure of the decoder. For purpose of brevity, we introduce some functional symbols in **Algorithm 1**. The function Normalize(y, $score$) is used to normalize the $score$ by the number of arguments in the event structure y. The function CalConstraitedSet(*last_token*) returns a set of valid candidate tokens for the prediction of next token based on the preceding token represented by the parameter *last_token*. For example, if the parameter *last_token* represents an argument type, this function returns the set of all argument role names of the event type currently being predicted by referring to the knowledge of event schema.

Algorithm 1: A nested constrained beam search decoding algorithm.

Input: The input text $x =< x_1, ..., x_n >$;
Output: A linearization sequence of an event graph $y =< y_1, ..., y_m >$;
// Initialization.
$score$ = 0; y, $finished = \{\}, \{\}$; $interBeam$ = $\{y, score\}$;
// Encoding.
Encode(x);
// Decoding.
if $P(EVTS) < P(NA)$ **then**
    return y;
for *i = 1 to max_event_num* **do**
    $new\_beam = \{\}$;
    $\{y, score\}$ = $interBeam$.pop();
    for v_i *in* $event\_type\_set \cup EOS$ **do**
        if v_i == EOS **then**
            $finished$.push($\{y, score\}$);
        else
            y_temp = $y \cup v_i$, $score\_temp$ += $P(v_i)$;
            $intraBeam$ = $\{y\_temp, score\_temp\}$;
            for *j = 1 to max_event_len* **do**
                $temp\_beam = \{\}$;
                $\{y\_t1, score\_t1\}$ = $intraBeam$.pop();
                $last\_token$ = GetLastToken(y_t1);
                if $last\_token == < stop >$ **then**
                    Normalize(y_t1, $score\_t1$);
                    new_beam.push($\{y\_t1, score\_t1\}$);
                else
                    $token\_set$ = CalConstraintedSet($last\_token$);
                    for u_i *in* $token\_set$ **do**
                        y_t2 = $y\_t1 \cup u_i$;
                        $score\_t2$ = $score\_t1$ + $P(u_i)$;
                        $temp\_beam$.push($\{y\_t2, score\_t2\}$);
                $intraBeam$ = $temp\_beam$.topK();
            new_beam += $intraBeam$.topK();
    $interBeam$ = new_beam.topK();
// Finishing.
while $interBeam$.*not_empty()* **do**
    $\{y, score\}$ = $interBeam$.pop();
    $finished$.push($\{y, score\}$);
$\{y, score\} \leftarrow finished$.topK($k$ = 1);
return y;

4.4 Postprocessing

In the preliminary experiments, we found that our event parsing model has a bias toward identifying the entity itself as an argument of the predicted event type. For example, in the sentence *Powell, the most moderate member of the Bush cabinet, said he fully agreed with the president's policy on Iraq and had no plans to leave*, for the golden event type **Personnel:End-Position** with the trigger word *leave*, the

pronoun *he*, which is adjacent to the trigger, is annotated as an answer argument with the role *Person*. However, our trigger-free generative model may tend to predict the entity name *Powell* as the argument of this event type. Conceptually speaking, the two entity mentions in this example are co-referenced and semantically equivalent. One possible reason is that while our model extracts the arguments for a specific event type, it recognizes the argument relations mainly by inspecting the contextual information surrounding the candidate arguments without depending on the triggers, and maybe the entity name itself contains richer contextual information than its mentions. In the dataset ACE2005, however, some entity mentions closer to the trigger of event type are usually annotated as the gold-standard arguments. Therefore, before the experimental evaluation we perform a light postprocessing to recover co-referring nodes in the event graph predicted by our model. Concretely, we perform a reference resolution operation by simply using the *coreferee* package that comes with python 3.8, and the ablation test of the use of coref system is shown in **Table 2** in **Section 5**.

5 Experiments

5.1 Dataset and Evaluation Metrics

We utilized the ACE 2005 corpus as our dataset. For comparison, as the same as previous work [32; 33; 8], we used the same test set with 40 newswire articles and the same development set with 30 other documents randomly selected from different genres and the rest 529 documents are used for training. Also, following previous work [33; 8; 4; 34], we use the following criteria to evaluate the results:

- An event type is correct if the predicted event type and subtype match those of a reference event.
- An argument is correctly identified if its event subtype and offsets match those of any of the reference argument mentions.
- An argument is correctly identified and classified if its event subtype, offsets and argument role match those of any of the reference argument mentions.

5.2 Implementation Details

We adopt BART-Large, which has 12 encoder and decoder layers, 1024 hidden units, and 16 attention heads, as our encoder-decoder model. Other hyper-parameters are tuned on the validation set. Specifically, the models are trained using cross-entropy with RAdam as optimizer and a learning rate of $5 * 10^{-5}$. Gradient is accumulated for 10 batches. Dropout is set to 0.25. Our models are trained for 50 epochs, the batch size in our training experiments is set to 400. For decoding, we set beam size to 3.

5.3 Overall Performance

In this section, we comprehensively compare our performance with the following state-of-the-art related methods that focus on the two event extraction subtasks: event detection and argument extraction:

- **JointBeam** [8] proposes a structure-based system by manually designed global features which explicitly capture the dependencies of multiple triggers and arguments.
- **DMCNN** [4] uses dynamic multi-pooling to extract the best features from the different parts of a sentence according to the position of trigger and argument candidate.
- **JRNN** [5] proposes a joint framework with bidirectional recurrent neural networks and manually designed features to jointly extract event triggers and arguments.
- **dbRNN** [6] is an LSTM-based framework that leverages the dependency graph information to extract event triggers and argument roles.
- **JMEE** [10] models dependency relations between words by Graph Convolutional Networks (GCNs) to exploit syntactic information.
- **RCEE** [20] proposes a new learning paradigm of EE, by explicitly casting it as a machine reading comprehension problem (MRC) based on BERT-Large model.
- **EKD** [14] leverages the wealth of the open-domain trigger knowledge to improve the event detection subtask.
- **Text2Event** [21] proposes a sequence-to-structure model Text2Event for EE, which can directly extract events from the text in an end-to-end manner.

Table 2 shows the overall performance comparison between our best system and the above state-of-the-art models. From **Table 2**, we can see that our approach achieves the best Precision, Recall and F1 score in event detection, argument identification and classification among all the compared methods. It is worth noting that our model simultaneously significantly improves both Precision and Recall without using any additional information including the POS tags, the syntactic dependency and external knowledge, which shows the superiority of the proposed graph parsing formulation for EE.

In **Table 2**, for our approach we also conduct ablation study on beam search to investigate contributions from the model architecture itself and the nested constrained beam search algorithm. Our model Seq2EG without beam search is already better than the previous best models. Further, the proposed decoding algorithm results in a significant improvement of 2.7% F1 score for final argument classification subtask. In addition, we conduct the ablation test of the use of coref system mentioned in the postprocessing section. If our model does not use the coref system, the F1 value for argument classification is 66.5%, leading to a performance drop of 1.8%.

Particularly, among all baselines, the Text2Event model is similar in spirit to our approach though the two methods have different experimental settings. For fair comparison, we modified their public code[2] to include the golden entity mention information as input by specifying the set of candidate arguments in the decoding algo-

[2] `https://github.com/luyaojie/Text2Event`

Table 2: Overall Performance compaison to the state-of-the-art methods with golden-standard entities on ACE2005 dataset. The results of baselines are adapted from their original papers. † indicates that the method uses dependency structures, ∧ indicates that the method uses external knowledge and resources, - indicates that the corresponding score is not available.

Method	Event Detection(%)			Argument Classification(%)		
	P	R	F1	P	R	F1
JointBeam(Li et al., 2013)†	73.7	62.3	67.5	64.7	44.4	52.7
DMCNN(Chen et al., 2015)†	75.6	63.6	69.1	62.2	46.9	53.5
JRNN(Nguyen et al., 2016)†	66.0	73.0	69.3	54.2	56.7	55.4
dbRNN(Sha et al., 2018)†	74.1	69.8	71.9	66.2	52.8	58.7
JMEE(Liu et al., 2018b)†	76.3	71.3	73.7	66.8	54.9	60.3
RCEE(Liu et al., 2020)∧	75.6	74.2	74.9	63.0	64.2	63.6
EKD(Tong et al., 2020)†∧	79.1	78.0	78.6	-	-	-
Text2Event(Lu et al., 2021)	69.6	74.4	71.9	52.5	55.2	53.8
- with golden-standard entities	77.9	70.0	73.8	60.0	66.3	62.9
Seq2EG (ours)	**83.8**	80.2	**82.0**	**68.3**	**68.3**	**68.3**
- w/o nested constrained beam	81.2	**80.9**	81.0	66.0	65.2	65.6
- w/o coref system	83.8	80.2	82.0	65.9	67.1	66.5

rithm, and presented the corresponding results in **Table 2**. Thus, in the same setting, our model significantly outperform the Text2Event model by 5.4% F1 value for argument classification. The possible reasons are two-fold: 1) our trigger-free fashion leads to more accurate event detection performance; 2) more importantly, our graph parsing framework can naturally model shared argument elements compared to the tree-based model (see more experimental analyses in **Section 6.2**).

6 Model Analysis

6.1 Effect of Multiple Event Extraction

Compared to the existing work, our EE approach provides a more natural formulation to model the multiple event correlations. To evaluate the effect of our approach to the multiple event recognition, we divide the test data into two parts (**1/1** and **1/N**) following previous work and perform evaluations separately [4; 5]. **1/1** means that one sentence only has one trigger or one argument plays a role in one sentence; otherwise, **1/N** is used.

Table 3 illustrates the performance (F1 scores) of **DMCNN** [4], **JRNN** [5], **JMEE** [10] and **HBTNGMA** [7], the four baseline models and our model for EE task. As shown in **Table 3**, our model significantly outperforms all the other methods. In the **1/N** data split, our method is **7.9%** better than the best baseline in the event detection phase, and **9.9%** better than the best baseline in the argument classification phase. The experimental results demonstrate that our method works well on the task of multiple event extraction.

Table 3: Performance comparison on single event sentences (1/1) and multiple event sentences (1/N).

Method	Event Detection			Argument Classification		
	1/1	1/N	All	1/1	1/N	All
DMCNN	66.7	45.9	54.4	74.4	70.7	72.5
JRNN	75.6	64.8	69.3	50.0	55.2	55.4
HBTNGMA	78.4	59.5	73.3	-	-	-
JMEE	75.2	72.7	73.7	59.3	57.6	58.5
Seq2EG	**83.7**	**80.6**	**82.0**	**69.6**	**67.5**	**68.3**

6.2 Effect of Exploiting of the Graph-structured Features

As illustrated in **Section 3.2**, our graph parsing formulation allows for incorporating some graph-structured features for EE, including exploiting the label semantics of event type and argument role, and introducing the argument type node to extend the event graph. In this section, we check the effects of these graph-structured features by the ablation study.

Concretely, the effect of the semantic representations of the event type and argument role labels is verified by treating them as a special symbol, without using their word embedding learned in the pre-trained language model. More specifically, we utilize both the event type label and the subtype label by averaging their word embeddings to make full use of the semantic representations with different granularities. Additionally, we evaluate the effect of the argument type nodes by removing this type of nodes from the extended event graph.

Table 4 shows F1 scores of the full Seq2EG model and with different components turned off one at a time. We can observe that, ignoring the semantic representation of event type and argument role labels leads to the decrease of F1 score of argument classification by 3.2% and 3.8%, respectively. Additionally, removing this type of nodes from the extended event graph results in a 4.2% drop in terms of F1 score of argument classification. We verified that all these components contribute to the main model, as the performance deteriorates with any of the components missing.

Table 4: Ablation studies on the the graph-structured features used in our model.

Method	Event Detection			Argument Classification		
	P	R	F1	P	R	F1
Full model	**83.8**	80.2	**82.0**	**68.3**	**68.3**	**68.3**
- w/o event type label	80.5	80.3	80.4	64.9	65.2	65.1
- w/o argument role label	80.1	**81.3**	80.7	61.6	67.6	64.5
- w/o argument type node	80.9	75.8	78.3	64.7	63.5	64.1

In order to further verify the effect of our graph parsing formulation on solving the argument sharing problem, we first construct a test subset by selecting the sentences with argument sharing phenomenon in the test data, and then run our model Seq2EG and the tree-structure based model Text2Event on this subset respectively. As shown in **Table 5**, our approach can substantially improve the argument classification performance by 8.9% in terms of F1 score compared to the baseline Text2Event, which demonstrates the great superiority of our graph parsing formulation in dealing with argument sharing phenomenon.

Table 5: Comparison of the argument classification performance between our model and the baseline Text2Event on the argument sharing test subset.

Method	**P**	**R**	**F1**
Text2Event	66.7	50.0	57.1
Seq2EG	**75.8**	**58.5**	**66.0**

6.3 Do Different Linearization Strategies Matter?

In this section, we inspect the effects of linearization strategies we proposed for the linearizing the event graph in **Section 4.1**. Firstly, we evaluate the linearization strategy for handling argument sharing by removing the use of special pointer symbols $<P_0>$, $<P_1>$, . . . , $<P_i>$ that represent the argument nodes in the linearized sequence of an event graph; next, we investigate the performance of our linearization ordering strategy by adopting the conventional graph traversal order, i.e., not postponing the output of argument role in the linearized sequence. Finally, we also try another traversing method breadth-first search (BFS) for comparison.

Table 6: Ablation studies on the linearization strategies of our model.

Method	Event Detection			Argument Classification		
	P	R	F1	P	R	F1
Full model	**83.8**	80.2	**82.0**	**68.3**	**68.3**	**68.3**
- w/o pointer symbols	80.1	78.0	79.1	64.9	57.1	60.7
- w/o ordering strategy	80.5	79.8	80.1	63.4	64.6	64.0
- w BFS linearization	80.9	**82.0**	81.5	67.3	60.3	63.6

From the **Table 6,** we can observe that both the two different linearization strategies are greatly beneficial to the performance boosting of our model. The linearization ordering strategy can improve the argument classification performance by **4.3%** in terms of F1 score. Particularly, a significant performance difference is visible in the

argument sharing strategy. Removal of argument sharing part leads to a **7.6%** drop in terms of F1 score of argument classification. This result indicates that the argument sharing plays a key role in the overall performance. Besides, it is easy to understand that the performance drops in event detection are relatively small, compared to the argument classification subtask. The results in the last row in **Table 6** demonstrate that DFS is a better traversing method for the event graph linearization compared to BFS.

6.4 Can Our Approach Alleviate the Long Tail Issue?

The trigger-based event extraction models generally suffer from the long tail issue [35; 14]. Taking the benchmark ACE2005 as an example, trigger words with frequency less than 5 account for 78.2% of the total. The long tail issue makes the trigger-based models perform poorly on unseen/sparsely labeled trigger words. In this section, we evaluate whether our approach could cope with the long tail issue.

Following previous work [14], we divide the event instances in the test set into three categories: Unseen, Sparsely-Labeled and Densely-Labeled, according to their trigger frequency in the training set. Specifically, the frequency of Sparsely Labeled is less than 5 and the frequency of Densely Labeled is more than 30. Also, following the work [14], we choose the following baselines for comparison: (1) **DMBERT** [4], (2) **DGBERT** [36], (3) **BOOTSTRAP** [37], and (4) the method **EKD** [14]. Note that the encoders in the first three baselines are replaced with more powerful BERT to make the baseline stronger.

As shown in **Table 7**, our approach substantially outperforms all baselines in two settings, especially on unseen setting (**+14.7%**). Why can our approach effectively mitigate the long tail issue? Besides the better generalization endowed by our seq2seq event graph parsing formulation, an important possible reason is that since our approach adopts a trigger-free way to detect the events, the event types corresponding to the unseen or sparsely-labeled triggers can also be expressed with other different triggers and thus appear many times in the training set, thereby alleviating the long tail problem. The experimental results clearly indicate that, the trigger-free event extraction approach may be a better alternative to the traditional trigger-based models.

6.5 Analysis of Cross-Attention Mechanism

In the absence of trigger words, can our Transformer-based seq2seq event extraction framework capture the key clues in the source sentence that express the target event type? In this section, we answer this question by the case study.

Fig. 3 presents several examples of the attention distributions learned by our model. In the first case, the target event type is *Life:Die* and the gold trigger is the word *killed*. We can see that when predicting this event type, our attention not only successfully attends the trigger word *killed*, but also attends another strongly indicative phrase *two people* with higher score. In the second case, the target event

Table 7: Performance comparison on the unseen, sparsely-labeled and densely-labeled settings.

Method	Unseen			Sparsely Labeled			Densely Labeled		
	P	R	F1	P	R	F1	P	R	F1
DMBERT	66.7	45.9	54.4	74.4	70.7	72.5	84.8	83.5	84.1
DGBERT	76.5	42.6	54.7	75.7	70.1	72.8	85.9	**83.8**	84.3
BOOTSTRAP	73.7	45.9	56.6	76.0	71.3	73.6	90.6	83.5	86.9
EKD	79.0	52.0	62.7	80.8	**72.4**	76.4	92.5	82.2	**87.1**
Seq2EG	**85.2**	**71.0**	**77.4**	**91.5**	71.4	**80.2**	**92.8**	77.6	84.5

Fig. 3: Visualization of cross-attention scores of sample instances learned by our model.

type is *Conflict:Attack*, and the gold trigger is the word *strike*. It can be observed that, the three words: *destroyed*, *houses* and *killed* are assigned with higher attention scores than the trigger *strike*, which seems plausible for this target type prediction. In the third case, the target event type is *Personnel:Elect*, and the gold trigger is the word*lost*. For this target type, there are relatively strong connections with the phrase *lost her seat* and another indicative word *election*.

These cases demonstrate that, though the triggers are not used in our model, the cross-attention mechanism between the decoder and encoder can learn to automatically capture the correlation between the target event type and multiple indicative words or phrases in the source sentence. However, on the other hand, we also found that our model may derive some redundant predictions due to the flexibility of the cross-attention mechanism. For instance, for the sentence *the demonstration came as Iraq's top US overseer Paul Bremer began his second week on the job amid continuing lawlessness in the country*, the annotated target event type is *Conflict: Demonstrate*. Given this input sentence, our model predicts an additional event type *Personnel: Start-Position* besides the target event type. Through analysis we consider that the event *Personnel: Start-Position* is wrongly predicted presumably because both the word *began* and *job* in the source sentence are strongly attended by the attention mechanism. Therefore, we will explore to employ the multiple attention mechanisms under the encoder-decoder architecture to further enhance the prediction accuracy in future work.

7 Conclusion

This paper presents the first work to formulate event extraction as a graph parsing task, and introduces a novel generation-based method to predict event graph by using a pre-trained seq2seq model. Our approach is conceptually simple and does not use syntactic dependency information and any other extra knowledge; however, it significantly outperforms the traditional classification-based encoder-only approaches, advancing the state of the art in event extraction.

In future work, we will integrate the syntactic dependency structure and external knowledge into our model to enhance the event extraction performance; additionally, we will further extend our model to perform the joint entity recognition and event extraction.

References

1. Rohini K Srihari and Wei Li. A question answering system supported by information extraction. In *Sixth Applied Natural Language Processing Conference*, pages 166–172, 2000.
2. Jonathan Berant, Vivek Srikumar, Pei-Chun Chen, Abby Vander Linden, Brittany Harding, Brad Huang, Peter Clark, and Christopher D Manning. Modeling biological processes for reading comprehension. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1499–1510, 2014.
3. Ralph Grishman, David Westbrook, and Adam Meyers. Nyu's english ace 2005 system description. *ACE*, 5, 2005.
4. Yubo Chen, Liheng Xu, Kang Liu, Daojian Zeng, and Jun Zhao. 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 (ACL-IJCNLP)*, pages 167–176, 2015.
5. Thien Huu Nguyen, Kyunghyun Cho, and Ralph Grishman. 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 (NAACL-HLT)*, 2016.
6. Lei Sha, Feng Qian, Baobao Chang, and Zhifang Sui. Jointly extracting event triggers and arguments by dependency-bridge rnn and tensor-based argument interaction. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.
7. Yubo Chen, Hang Yang, Kang Liu, Jun Zhao, and Yantao Jia. Collective event detection via a hierarchical and bias tagging networks with gated multi-level attention mechanisms. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1267–1276, 2018.
8. Qi Li, Heng Ji, and Liang Huang. Joint event extraction via structured prediction with global features. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 73–82, 2013.

9. Thien Nguyen and Ralph Grishman. Graph convolutional networks with argument-aware pooling for event detection. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.
10. Xiao Liu, Zhunchen Luo, and Heyan Huang. Jointly multiple events extraction via attention-based graph information aggregation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1247–1256, 2018.
11. Haoran Yan, Xiaolong Jin, Xiangbin Meng, Jiafeng Guo, and Xueqi Cheng. Event detection with multi-order graph convolution and aggregated attention. 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 5766–5770, 2019.
12. Viet Dac Lai, Tuan Ngo Nguyen, and Thien Huu Nguyen. Event detection: Gate diversity and syntactic importance scores for graph convolution neural networks. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 5405–5411, 2020.
13. Fayuan Li, Weihua Peng, Yuguang Chen, Quan Wang, Lu Pan, Yajuan Lyu, and Yong Zhu. Event extraction as multi-turn question answering. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 829–838, Online, November 2020. Association for Computational Linguistics.
14. Meihan Tong, Bin Xu, Shuai Wang, Yixin Cao, Lei Hou, Juanzi Li, and Jun Xie. Improving event detection via open-domain trigger knowledge. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 5887–5897, 2020.
15. Shulin Liu, Yang Li, Feng Zhang, Tao Yang, and Xinpeng Zhou. Event detection without triggers. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pages 735–744, 2019.
16. Jianye Xie, Haotong Sun, Junsheng Zhou, Weiguang Qu, and Xinyu Dai. Event detection as graph parsing. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP*, pages 1630–1640, 2021.
17. Pranav Rajpurkar, Robin Jia, and Percy Liang. Know what you don't know: Unanswerable questions for SQuAD. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 784–789, 2018.
18. Mandar Joshi, Eunsol Choi, Daniel Weld, and Luke Zettlemoyer. TriviaQA: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 1601–1611, 2017.
19. Xinya Du and Claire Cardie. Event extraction by answering (almost) natural questions. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 671–683, 2020.
20. Jian Liu, Yubo Chen, Kang Liu, Wei Bi, and Xiaojiang Liu. Event extraction as machine reading comprehension. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1641–1651, 2020.

21. Yaojie Lu, Hongyu Lin, Jin Xu, Xianpei Han, Jialong Tang, Annan Li, Le Sun, Meng Liao, and Shaoyi Chen. Text2Event: Controllable sequence-to-structure generation for end-to-end event extraction. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (ACL-IJCNLP)*, 2021.
22. David Wadden, Ulme Wennberg, Yi Luan, and Hannaneh Hajishirzi. 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, 2019.
23. Xinya Du, Alexander Rush, and Claire Cardie. 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 (EACL)*, pages 634–644, 2021.
24. Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pages 4171–4186, 2019.
25. Kaitao Song, Xu Tan, Tao Qin, Jianfeng Lu, and Tie-Yan Liu. MASS: masked sequence to sequence pre-training for language generation. *CoRR*, abs/1905.02450, 2019.
26. Xiaozhi Wang, Xu Han, Zhiyuan Liu, Maosong Sun, and Peng Li. Adversarial training for weakly supervised event detection. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pages 998–1008, 2019.
27. Jingqing Zhang, Yao Zhao, Mohammad Saleh, and Peter J. Liu. PEGASUS: pre-training with extracted gap-sentences for abstractive summarization. *CoRR*, abs/1912.08777, 2019.
28. Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 7871–7880, 2020.
29. Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *CoRR*, abs/1910.10683, 2019.
30. Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.
31. Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems (NIPS)*, pages 5998–6008, 2017.

32. Heng Ji and Ralph Grishman. Refining event extraction through cross-document inference. In *Proceedings of ACL-08: HLT*, pages 254–262, 2008.
33. Shasha Liao and Ralph Grishman. Using document level cross-event inference to improve event extraction. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 789–797, 2010.
34. Jian Liu, Yubo Chen, and Kang Liu. Exploiting the ground-truth: An adversarial imitation based knowledge distillation approach for event detection. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 6754–6761, 2019.
35. Yaojie Lu, Hongyu Lin, Xianpei Han, and Le Sun. Distilling discrimination and generalization knowledge for event detection via delta-representation learning. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 4366–4376, 2019.
36. Yubo Chen, Shulin Liu, Xiang Zhang, Kang Liu, and Jun Zhao. Automatically labeled data generation for large scale event extraction. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 409–419, 2017.
37. Hangfeng He and Xu Sun. A unified model for cross-domain and semi-supervised named entity recognition in chinese social media. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 31, 2017.