Event Extraction by Answering (Almost) Natural Questions

Xinya Du and Claire Cardie

Jayaram Gokulan – jg929 - Critique #6

As a state-of-the-art, pre-trained language model, BERT continues to gather applause for its ability to extract features with relatively less effort. Its choice as the successor to many of the current non-performant NLP machinery is hence irrefutable. The article resonates with this theme, as we see an event extraction model driven by BERT's question and answering scheme. The merits outweigh most demerits, as evidenced by the improved F1, R, and other metrics employed by the study. Yet this doesn't imply the errors generated during the process are completely random events. The error analysis described in the latter half of this study deliberates on the inefficacies of the Q&A model deployed; entities weren't consistently recognized throughout the experiment. Granted that such complications within sentences are sporadic, this needn't gather much attention other than that occasional cases could cumulatively drive down the evaluation metrics. Nevertheless, such errors provide deeper insights into probable training or data deficiencies within the proposed model. While model breakdowns at complex sentences (failure to capture longer sentences as entities) reveal possibilities that either the final hidden layer in BERT may have masked language modelling training superseding the fine-tuning training component (for event entity recognition) or that the training data supplied to the BERT model lacks representative data (about a population). On similar grounds, failures to capture exact semantic role due to insufficient context also signals improper encoding information being captured in the final layers of BERT. Though seemingly separate issues, both these encounters unanimously suggest that the underlying model performance is capped at the cost of suitable training measures. Indeed, if the attention mechanism is learned on the wrong word dependencies and relationships, the resulting informational encoding cannot be satisfactorily relied on for good predictions. The final contention raised in this study also has a similar background – the contextual word embeddings learned through the model's training may have more profound implications. The assertions presented in the study are valid in that exact matches of rare words are admittedly difficult to find within training examples. However, going by the structural definition of contextual word embeddings, one can then reason how the contextual embedding representation should align with a related word from the training data and hence necessarily find the trigger word in the test set. This predicament may then indicate poor representations being developed upstream of the BERT model, even after sufficient training, signaling how the samples used could be the culprit once more. On a lighter note, we also observe the co-occurrence of peak performances with encoding information from annotation guidelines. Considering how the extrapolation of the same event extraction technique to other possible event scenarios and datasets may demand new annotation guidelines, the readers are inevitably hinted at the presence of a specific 'data dependency'. Suitable pretraining/training may resolve all of this but still warrant a better explanation surrounding such phenomena.

Event Extraction by Answering (Almost) Natural Questions

Xinya Du and **Claire Cardie**
Department of Computer Science
Cornell University
Ithaca, NY, USA
{xdu, cardie}@cs.cornell.edu

Abstract

The problem of event extraction requires detecting the event trigger and extracting its corresponding arguments. Existing work in event argument extraction typically relies heavily on entity recognition as a preprocessing/concurrent step, causing the well-known problem of error propagation. To avoid this issue, we introduce a new paradigm for event extraction by formulating it as a question answering (QA) task that extracts the event arguments in an end-to-end manner. Empirical results demonstrate that our framework outperforms prior methods substantially; in addition, it is capable of extracting event arguments for roles not seen at training time (i.e., in a zero-shot learning setting).[1]

1 Introduction

Event extraction is a long-studied and challenging task in Information Extraction (IE) (Sundheim, 1992; Grishman and Sundheim, 1996; Riloff, 1996). Its goal is to extract structured information — "what is happening" and the persons/objects that are involved — from unstructured text. The task is illustrated via an example in Figure 1, which depicts an ownership transfer event (the *event type*), triggered by the word "sale" (the event *trigger*) and accompanied by its extracted *arguments* — text spans denoting entities that fill a set of (semantic) *roles* associated with the event type (e.g., BUYER, SELLER and ARTIFACT for ownership transfer events).

Recent successful approaches to event extraction have benefited from dense features extracted by neural models (Chen et al., 2015; Nguyen et al., 2016; Liu et al., 2018) as well as contextualized lexical representations from pretrained language models (Zhang et al., 2019b; Wadden et al., 2019).

Input:
As part of the 11-billion-dollar **sale** of USA Interactive's film and television operations to the French company and its parent company in December 2001, USA Interactive received 2.5 billion dollars in preferred shares in Vivendi Universal Entertainment.

Extracted Event:

Event type		Transaction-Transfer-Ownership
Trigger		"sale"
Args.	Buyer	"French company", "parent company"
	Seller	"USA Interactive"
	Artifact	"operations"
	Place	-

Figure 1: Event extraction example from the ACE 2005 corpus (Doddington et al., 2004).

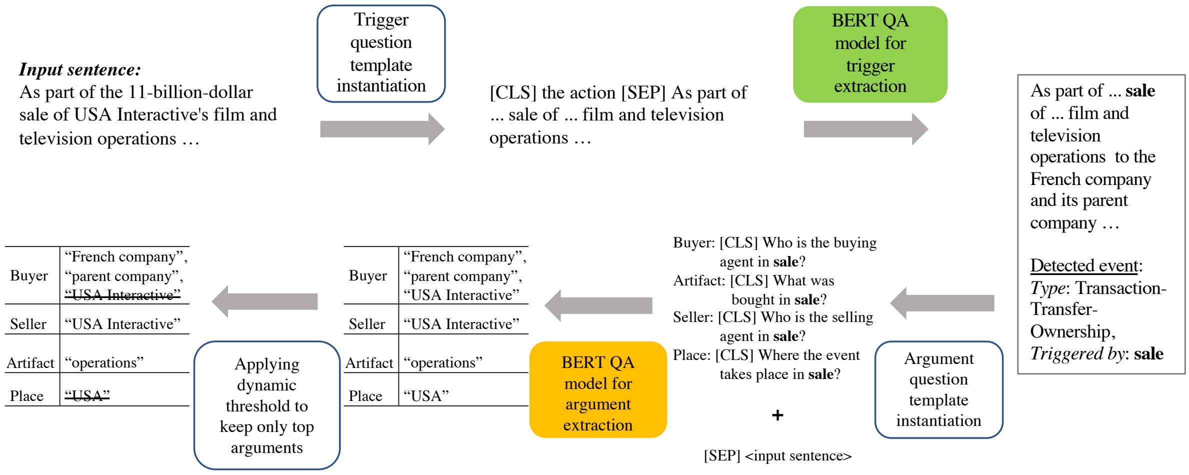
The approaches, however, exhibit two key weaknesses. First, they rely heavily on entity information for argument extraction. In particular, event argument extraction generally consists of two steps – first identifying entities and their general semantic class with trained models (Wadden et al., 2019) or a parser (Sha et al., 2018), then assigning argument roles (or no role) to each entity. Although joint models (Yang and Mitchell, 2016; Nguyen and Nguyen, 2019; Zhang et al., 2019a; Lin et al., 2020) have been proposed to mitigate this issue, error propagation (Li et al., 2013) still occurs during event argument extraction.

A second weakness of neural approaches to event extraction is their inability to exploit the similarities of related argument roles across event types. For example, the ACE 2005 (Doddington et al., 2004) CONFLICT.ATTACK events and JUSTICE.EXECUTE events have TARGET and PERSON argument roles, respectively. Both roles, however, refer to a *human being (who) is affected* by an action. Ignoring the similarity can hurt performance, especially for argument roles with few/no examples at training time (e.g., similar to the zero-shot setting in Levy et al. (2017)).

In this paper, we propose a new paradigm for the event extraction task – formulating it as a question answering (QA)/machine reading comprehension (MRC) task (**Contribution 1**). The general framework is illustrated in Figure 2. Using BERT (Devlin

[1] Our code and question templates for the work are open sourced at https://github.com/xinyadu/eeqa for reproduction purpose.

Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, pages 671–683,
November 16–20, 2020. ©2020 Association for Computational Linguistics



Input sentence:
As part of the 11-billion-dollar sale of USA Interactive's film and television operations …
Trigger question template instantiation
[CLS] the action [SEP] As part of ... sale of ... film and television operations …
BERT QA model for trigger extraction
As part of ... **sale** of ... film and television operations to the French company and its parent company …
Detected event:
Type: Transaction-Transfer-Ownership,
Triggered by: **sale**
Argument question template instantiation
Buyer: [CLS] Who is the buying agent in **sale**?
Artifact: [CLS] What was bought in **sale**?
Seller: [CLS] Who is the selling agent in **sale**?
Place: [CLS] Where the event takes place in **sale**?
+
[SEP] <input sentence>
BERT QA model for argument extraction
Buyer “French company”, “parent company”, “USA Interactive”
Seller “USA Interactive”
Artifact “operations”
Place “USA”
Applying dynamic threshold to keep only top arguments
Buyer “French company”, “parent company”, ~~“USA Interactive”~~
Seller “USA Interactive”
Artifact “operations”
Place ~~“USA”~~

Figure 2: Our framework for event extraction.

et al., 2019) as the base model for obtaining contextualized representations from the input sequences, we develop two BERT-based QA models – one for event trigger detection and the other for argument extraction. For each, we design one or more Question Templates that map the input sentence into the standard BERT input format. Thus, trigger detection becomes a request to identify “the action” or the “verb” in the input sentence and determine its event type; and argument extraction becomes a sequence of requests to identify the event's arguments, each of which is a text span in the input sentence. Details will be explained in Section 2. To the best of our knowledge, this is the first attempt to cast event extraction as a QA task.

Treating event extraction as QA overcomes the weaknesses in existing methods identified above (**Contribution 2**): (1) Our approach requires *no entity annotation* (gold or predicted entity information) and no entity recognition pre-step; event argument extraction is performed as an end-to-end task; (2) The question answering paradigm naturally permits the transfer of argument extraction knowledge across semantically related argument roles. We propose rule-based question generation strategies (including incorporating descriptions in annotation guidelines) for templates creation, and conduct extensive experiments to evaluate our framework on the Automatic Content Extraction (ACE) event extraction task and show empirically that the performance on both trigger and argument extraction outperform prior methods (Section 3.2). Finally, we show that our framework extends to the zero-shot setting – it is able to extract event arguments for unseen roles (**Contribution 3**).

2 Methodology

In this section, we first provide an overview for the framework (Figure 2), then go deeper into details of its components: question generation strategies for template creation, as well as training and inference of QA models.

2.1 Framework Overview

Our QA framework for event extraction relies on two sets of Question Templates that map an input sentence to a suitable input *sequence* for two instances of a standard pre-trained bidirectional transformer (BERT (Devlin et al., 2019)). The first of these, BERT_QA_Trigger (green box in Figure 2), extracts from the input sentence the event trigger which is a single token, and its type (one of a fixed set of pre-defined event types). The second QA model, BERT_QA_Arg (orange box in Figure 2), is applied to the input sequence, the extracted event trigger and its event type to iteratively identify candidate event arguments (spans of text) in the input sentence. Finally, a dynamic threshold is applied to the extracted candidate arguments, and only the arguments with probability above the threshold are retained.

The input sequences for the two QA models share a standard BERT-style format:

[CLS] <question> [SEP] <sentence> [SEP]

where [CLS] is BERT's special classification token, [SEP] is the special token to denote separation, and <sentence> is the tokenized input sentence. We provide details on how to obtain the **<question>** in Section 2.2. Details on the QA models and the inference process will be explained in Section 2.3.

Argument	Template 1 (Role name)	Template 2 (Type + Role question)	Template 3 (Annotation guideline question)
Artifact	artifact	What is the artifact?	What is being transported?
Agent	agent	Who is the agent?	Who is responsible for the transport event?
Vehicle	vehicle	What is the vehicle?	What is the vehicle used?
Origin	origin	What is the origination?	Where the transporting originated?
Destination	destination	What is the destination?	Where the transporting is directed?

Table 1: Arguments (of event type MOVEMENT.TRANSPORT) and corresponding questions from three templates. "in <trigger>" is not added to the questions in this example.

2.2 Question Generation Strategies

For our QA-based framework for event extraction to be easily moved from one domain to the other, we concentrated on developing question generation strategies that not only worked well for the task, but can be quickly and easily implemented. For event trigger detection, we experiment with a set of four fixed templates – "what is the trigger", "trigger", "action", "verb". Basically, we use the fixed literal phrase as the question. For example, if we choose the "action" template, the input sequence for the example sentence in Figures 1 and 2 is instantiated as:

> [CLS] action [SEP] As part of the 11-billion-dollar sale ... [SEP]

As for event argument extraction, we design three templates with argument role name, basic argument based question and annotation guideline based question, respectively:

- **Template 1 (Role Name)** For this template, <question> is simply instantiated with the argument role name (e.g., artifact, agent, place).

- **Template 2 (Type + Role)** Instead of directly using the argument role name (<role name>) as the question, we first determine the argument role's general semantic type — one of person, place, other; and construct the associated "WH" word question – *who* for person, *where* for place and *what* for all other cases, of the following form:

  <WH_word> is the <role name> ?

  Examples are shown in Table 1 for the arguments of event type MOVEMENT.TRANSPORT. By adding the WH word, more semantic information is included as compared to Template 1.

- **Template 3 (Incorporating Annotation Guidelines)** To incorporate even more semantic information and make the question more natural sounding, we utilize the descriptions of each argument role provided in the ACE annotation guidelines for events (Linguistic Data Consortium, 2005) for generating the questions.

- **+ "in <trigger>"** Finally, for each template type, it is possible to encode the trigger information by adding "in <trigger>" at the end of the question (where <trigger> is instantiated with the real trigger token obtained from the trigger detection phase). For example, the Template 2 question incorporating trigger information would be:

  <WH_word> is the <argument> in <trigger>?

To help better understand all the strategies above, Table 1 presents an example for argument roles of event type MOVEMENT.TRANSPORT. We see that the annotation guideline based questions are more natural and encode more semantics about a given argument role, than the simple Type + Role question "what is the artifact?".

2.3 Question Answering Models

We use BERT (Devlin et al., 2019) as the base model for getting contextualized representations for the input sequences for both BERT_QA_Trigger and BERT_QA_Arg. After the instantiation with question templates the sequences are of format [CLS] <question> [SEP] <sentence> [SEP].

Then we get the contextualized representations of each token for trigger detection and argument extraction with $\texttt{BERT}_{Tr}$ and $\texttt{BERT}_{Arg}$, respectively. For the input sequence $(e_1, e_2, ..., e_N)$ prepared for trigger detection, we have:

$$\mathbf{E} = [\mathbf{e}_1, \mathbf{e}_2, ..., \mathbf{e}_N]$$
$$\mathbf{e}_1, \mathbf{e}_2, ..., \mathbf{e}_N = \texttt{BERT}_{Tr}(e_1, e_2, ..., e_N)$$

For the input sequence $(a_1, a_2, ..., a_M)$ prepared for argument span extraction, we have:

$$\mathbf{A} = [\mathbf{a}_1, \mathbf{a}_2, ..., \mathbf{a}_M]$$
$$\mathbf{a}_1, \mathbf{a}_2, ..., \mathbf{a}_M = \texttt{BERT}_{Arg}(a_1, a_2, ..., a_M)$$

The output layer of each QA model, however, differs: BERT_QA_Trigger predicts the event type for each token in sentence (or None if it is not an event trigger), while BERT_QA_Arg predicts the start and end offsets for the argument span with a different decoding strategy.

More specifically, for **trigger prediction**, we introduce a new parameter matrix $\mathbf{W}_{tr} \in \mathbb{R}^{H \times T}$, where H is the hidden size of the transformer and T is the number of event types plus one (for non-trigger tokens). softmax normalization is applied across the T types to produce P_{tr}, the probability distribution across the event types:

$$P_{tr} = \texttt{softmax}(\mathbf{E}\mathbf{W}_{tr}) \in \mathbb{R}^T \times N$$

At test time, for trigger detection, to obtain the type for each token $e_1, e_2, ..., e_N$, we simply apply argmax to P_{tr}.

For **argument span prediction**, we introduce two new parameter matrices $\mathbf{W}_s \in \mathbb{R}^{H \times 1}$ and $\mathbf{W}_e \in \mathbb{R}^{H \times 1}$; softmax normalization is then applied across the input tokens $a_1, a_2, ..., a_M$ to produce the probability of each token being selected as the start/end of the argument span:

$$P_s(i) = \texttt{softmax}(\mathbf{a}_i\mathbf{W}_s)$$
$$P_e(i) = \texttt{softmax}(\mathbf{a}_i\mathbf{W}_e)$$

To train the models (BERT_QA_Trigger and BERT_QA_Arg), we minimize the negative log-likelihood loss for both models, parameters are updated during the training process. In particular, the loss for the argument extraction model is the sum of two parts: the start token loss and end end token loss. For the training examples with no argument span (no answer case), we minimize the start and end probability of the first token of the sequence ([CLS]).

$$\mathcal{L}_{arg} = \mathcal{L}_{arg\_start} + \mathcal{L}_{arg\_end}$$

Inference with Dynamic Threshold for Argument Spans At test time, predicting the argument spans is more complex – for each argument role, there can be *several* or *no* spans to be extracted. After the output layer, we have the probability of each token $a_i \in (a_1, a_2, ..., a_M)$ being the start ($P_s(i)$) and end ($P_e(i)$) of the argument span.

Algorithm 1: Harvesting Argument Spans Candidates

```
Input  : $P_s(i)$, where $i \in \{1, ..., M\}$,
         $P_e(i)$, where $i \in \{1, ..., M\}$
Output : valid candidate spans for the argument role
for $start \leftarrow 1$ to $M$ do
    for $end \leftarrow 1$ to $M$ do
        if $start$ or $end$ not in the input sentence
          then continue;
        if $end - start + 1 > MaxSpanLength$ then
          continue;
        if $P_s(start) < P_s([CLS])$ or
          $P_e(end) < P_e([CLS])$ then continue;
        // add the valid candidate
           span to the set
        $score \leftarrow P_s(start) + P_e(end)$;
        $no\_ans\_score \leftarrow P_s(1) + P_e(1) - score$;
        $candidates$.add([$start, end, no\_ans\_score$])
    end
end
```

Algorithm 2: Automatic Filtering on Argument Candidates

```
Input  : $dev\_candidates(i), i \in \{1, ..., dev\_n\}$,
         $test\_candidates(i), i \in \{1, ..., test\_n\}$.
Output : A set of top arguments from test_candidates
// get the best dynamic threshold
sort($dev\_candidates, key = no\_ans\_score$);
$best\_thresh \longleftarrow 0$;
$best\_res \longleftarrow 0$;
for $i \leftarrow 1$ to $dev\_n$ do
    $thresh \leftarrow dev\_candidates(i).no\_ans\_score$;
    $result \leftarrow$ eval($dev\_candidates$ with
      $no\_ans\_score <= thresh$);
    if $result > best\_res$ then
      $best\_thresh \leftarrow thresh$;
      $best\_res \leftarrow result$;
end
// apply the best threshold
$final\_arguments \longleftarrow \{\}$;
for $i \leftarrow 1$ to $test\_n$ do
    if $test\_candidates(i).no\_ans\_score <=$
      $best\_thresh$ then
      $final\_arguments$.add($test\_candidates(i)$);
end
```

Firstly, we run an algorithm to harvest all valid argument spans candidates for each argument role (Algorithm 1). Basically, we:

1. Enumerate all the possible combinations of

start offset ($start$) and end offset (end) of the argument spans (line 1–2);

2. Eliminate the spans not satisfying the constraints: start and end token must be within the sentence; the length of the span should be shorter than a maximum length constraint; Argument spans should have larger probability than the probability of “no argument” (which is stored at the [CLS] token) (line 3–5);

3. Calculate the relative no answer score (no_ans_score) for the candidate span and add the candidate to list (line 6–8).

Then we run another algorithm to filter out candidate arguments that should not be included (Algorithm 2). More specifically, we obtain a probability threshold ($best\_thresh$) that helps achieve best evaluation results on the dev set (line 1–9) and keep only those arguments with no_ans_score smaller than the threshold (line 10–13). With the dynamic threshold for determining the number of arguments to be extracted for each role, we avoid adding a (hard) hyperparameter for this purpose.

Another easier way to get final argument predictions is to directly include all the candidates with $no\_ans\_score < 0$, which does not require tuning the dynamic threshold $best\_thresh$.

3 Experiments

3.1 Dataset and Evaluation Metric

We conduct experiments on the ACE 2005 corpus (Doddington et al., 2004), it contains documents crawled between year 2003 and 2005 from a variety of areas such as newswire (nw), weblogs (wl), broadcast conversations (bc) and broadcast news (bn). The part that we use for evaluation is fully annotated with 5,272 event triggers and 9,612 arguments. We use the same data split and preprocessing step as in the prior works (Zhang et al., 2019b; Wadden et al., 2019).

As for evaluation, we adopt the same criteria defined in Li et al. (2013): An event trigger is correctly identified (ID) if its offsets match those of a gold-standard trigger; and it is correctly classified if its event type (33 in total) also match the type of the gold-standard trigger. An event argument is correctly identified (ID) if its offsets and event type match those of any of the reference argument mentions in the document; and it is correctly classified if its semantic role (22 in total) is also correct. Though our framework does not involve the trigger/argument identification step and tackles the identification + classification in an end-to-end way. We still report the trigger/argument identification's results to compare to prior work. It could be seen as a more lenient eval metric, as compared to the final trigger detection and argument extraction metric (ID + Classification), which requires both the offsets and the type to be correct. All the aforementioned elements are evaluated using precision (denoted as P), recall (denoted as R) and F1 scores (denoted as F1).

3.2 Results

Evaluation on ACE Event Extraction We compare our framework's performance to a number of prior competitive models: **dbRNN** (Sha et al., 2018) is an LSTM-based framework that leverages the dependency graph information to extract event triggers and argument roles. **Joint3EE** (Nguyen and Nguyen, 2019) is a multi-task model that performs entity recognition, trigger detection and argument role assignment by shared BiGRU hidden representations. **GAIL** (Zhang et al., 2019b) is an ELMo-based model that utilizes generative adversarial network to help the model focus on harder-to-detect events. **DYGIE++** (Wadden et al., 2019) is a BERT-based framework that models text spans and captures within-sentence and cross-sentence context. **OneIE** (Lin et al., 2020) is a joint neural model for extraction with global features.[2]

In Table 2, we present the comparison of models' performance on trigger detection. We also implement a BERT fine-tuning baseline and it reaches nearly same performance as its counterpart in the DYGIE++. We observe that our BERT_QA_Trigger model with best trigger questioning strategy reaches comparable (better) performance with the baseline models.[3]

Table 3 shows the comparison between our model and baseline systems on argument extraction. Notice that the performance of argument extraction is directly affected by trigger detection. Because argument extraction correctness requires the trigger to which the argument refers to be correctly identified and classified. We observe, (1)

[2] Slightly different from our and Wadden et al. (2019)'s data pre-processing, OneIE skips lines before the <text> tag (e.g., headline, datetime).

[3] Note that OneIE is concurrent to our work and reports better performance. On trigger detection, it reaches 74.7 F1 as compare to our 72.39. On argument extraction (affected by trigger detection), it reaches 56.8 as compared to our 53.31.

	Trigger Identification			Trigger ID + Classification		
	P	R	F1	P	R	F1
dbRNN (Sha et al., 2018)	-	-	-	74.10	69.80	71.90
Joint3EE (Nguyen and Nguyen, 2019)	70.50	74.50	72.50	68.00	71.80	69.80
GAIL-ELMo (Zhang et al., 2019b)	76.80	71.20	73.90	74.80	69.40	72.00
DYGIE++, BERT + LSTM (Wadden et al., 2019)	-	-	-	-	-	68.90
DYGIE++, BERT FineTune (Wadden et al., 2019)	-	-	-	-	-	69.70
Our BERT FineTune	69.77	76.18	72.84	67.15	73.20	70.04
BERT_QA_Trigger (best trigger question strategy)	74.29	77.42	75.82	71.12	73.70	**72.39**

Table 2: Trigger detection results.

	Argument Identification			Argument ID + Classification		
	P	R	F1	P	R	F1
dbRNN (Sha et al., 2018)	-	-	57.20	-	-	50.10
Joint3EE (Nguyen and Nguyen, 2019)	-	-	-	52.10	52.10	52.10
GAIL-ELMo (Zhang et al., 2019b)	63.30	48.70	55.10	61.60	45.70	52.40
DYGIE++, BERT + LSTM (Wadden et al., 2019)	-	-	54.10	-	-	51.40
DYGIE++, BERT + LSTM ensemble (Wadden et al., 2019)	-	-	55.40	-	-	52.50
BERT_QA_Arg (annot. guideline question template)	58.02	50.69	54.11	56.87	49.83	53.12*
w/o dynamic threshold	53.39	54.69	54.03	50.81	52.78	51.77
BERT_QA_Arg (ensemble argument question template 2&3)	58.90	52.08	55.29	56.77	50.24	**53.31**

Table 3: Argument extraction results. * indicates statistical significance ($p < 0.05$).

Our BERT_QA_Arg model with best argument question generation strategy (annotation guideline based questions) outperforms prior work significantly, although it uses no entity recognition resources; (2) Drop of F1 performance from argument identification (correct offset) to argument ID + classification (both correct offset and argument role) is only around 1%, while the gap is around 3% for prior models which rely on entity recognition and a multi-step process for argument extraction. This once again demonstrates the benefit of our new formulation for the task as question answering.

To better understand how the dynamic threshold is affecting our framework's performance. We conduct an ablation study on this (Table 3) and find that the threshold increases the precision and the general F1 substantially. The last row in the table shows the test time ensemble performance of the predictions from BERT_QA_Arg trained with template 2 question, and another BERT_QA_Arg trained with template 3 question. The ensemble system outperforms the non-ensemble system in both precision and recall, demonstrating the benefit from both templates.

Evaluation on Unseen Argument Roles To verify how our formulation provides advantages for extracting arguments with unseen argument roles (similar to the zero-shot relation extraction setting in Levy et al. (2017)), we conduct another experiment, where we keep 80% of the argument roles (16 roles) seen at training time, and 20% (6 roles) only seen at test time. Specifically, the unseen roles are "Vehicle, Artifact, Target, Victim, Recipient, Buyer".

	Argument ID + Classification		
	P	R	F1
Random NE	26.61	24.77	25.66
GAIL (Zhang et al., 2019b)	-	-	-
Our model			
w/ Role name	73.83	53.21	61.85
w/ Type + Role Q	77.18	55.05	64.26
w/ Annot. Guideline Q	78.52	59.63	**67.79**

Table 4: Evaluation on unseen argument roles.

Table 4 presents the results. **Random NE** is our random baseline that selects a named entity in the sentence, it comes with a reasonable performance of near 25%. Prior model such as **GAIL** is not capable of handling the unseen roles. **ZSTE** (Huang et al., 2018) is a framework for zero-shot transfer learning of event extraction with AMR. It maps each parsed candidate span to a specific type in a

	Predicted Triggers						Gold Triggers					
	Argument Identification			Argument ID + C			Argument Identification			Argument ID + C		
Question	P	R	F1	P	R	F1	P	R	F1	P	R	F1
Role name	47.50	51.22	49.29	44.85	48.78	46.74	56.12	67.01	61.09	51.95	63.19	57.02
+ in <trigger>	53.86	51.91	52.87	51.63	50.17	50.89	69.00	64.76	66.81	64.70	61.28	62.94
Type + Role question	51.02	47.74	49.33	48.64	45.83	47.19	60.31	62.15	61.22	57.17	59.20	58.17
+ in <trigger>	54.61	50.69	52.58	52.98	48.96	50.89	70.38	62.85	66.40	67.55	60.59	63.88
Annot. guideline question	51.17	51.22	51.19	48.99	49.83	49.40	60.03	68.40	63.94	57.08	65.97	61.21
+ in <trigger>	58.02	50.69	54.11	56.87	49.83	**53.12**	71.17	65.45	68.19	67.88	63.02	**65.36**

Table 5: Influence of question generation strategies on argument extraction.

target event ontology. This framework's argument extraction's results are affected by the AMR results and their reported F1 is around 20-30% in their evaluation setting.

Using our QA-based framework, as we leverage more semantic information and naturalness into the question (from question template 1 to 2, to 3), both the precision and recall increase substantially.

	Trigger ID + Classification		
	P	R	F1
leaving empty	67.15	73.20	70.04
"what is the trigger"	70.15	69.98	70.06
"What happened"	70.53	69.48	70.00
"trigger"	69.73	71.46	70.59
"action"	72.25	71.71	71.98
"verb"	71.12	73.70	**72.39**

Table 6: Effect of different questions on trigger detection.

4 Further Analysis

4.1 Influence of Question Templates

To investigate how the question generation strategies affect the performance of event extraction, we perform experiments on trigger and argument extractions with different strategies, respectively.

In Table 6, we try different fixed questions for trigger detection. By "leaving empty", we mean instantiating the **question** with empty string.[4] There's no substantial gap between different alternatives. By using "verb" as the question, our BERT_QA_Trigger model achieves best performance (measured by F1 score). The QA model also encodes the semantic *interactions* between the fixed question ("verb") and the sentence, this explains why BERT_QA_Trigger is better than BERT FineTune in trigger detection.

The comparison between different question generation strategies for argument extraction is even more interesting. In Table 5, we present the results in two settings: event argument extraction with predicted triggers (the same setting as in Table 3), and with gold triggers. In summary, we finds that:

- *Adding "in <trigger>" afterwards the question consistently improve the performance.* It serves as an indicator for what/where the trigger is in the input sentence. Without adding the "in <trigger>", for each template (1, 2 & 3), the F1 of models' predictions drop around 3 percent when given predicted triggers, and more when given gold triggers.

- *Our template 3 questioning strategy which is most natural achieves the best performance.* As we mentioned earlier, template 3 questions are based on descriptions for argument roles in the annotation guideline, thus encoding more semantic information about the role name. And this corresponds to the accuracy of models' predictions – template 3 outperforms template 1&2 in both with "in <trigger>" and without "in <trigger>" setting. What's more, we observe that template 2 (adding a WH_word to form the questions) achieves better performance than the template 1 (directly using argument role name).

4.2 Error Analysis

We further conduct error analysis and provide a number of representative examples. Table 7 summarizes error statistics for trigger detection and argument extraction.

For event triggers, the majority of the errors relates to missing or spurious predictions, and only 8.29% involves misclassified event types (e.g., a ELECT event is mistaken for a START-POSITION

[4] In this case, the model degrades to a token classification model. It matches our BERT FineTune baseline's performance.

Missing	Spurious	Wrong Type
46.08%	45.62%	8.29%

same number		more	less
exact match	not exact match		
14.48%	17.21%	13.93%	54.37%

Table 7: Trigger errors (upper table) and argument errors (lower table).

event). For event arguments, on the sentences that comes with at least one event in gold data, our framework extracts more argument spans only around 14% of the cases. Most of the time (54.37%), our framework extracts less argument spans, this corresponds to the results in Table 3, where the precision of our models are higher. In around 30% of the cases, our framework extracts same number of argument spans as in the gold data, half of them match exactly the gold arguments.

After examining the example predictions, we find that reasons for errors can be mainly divided into the following categories:

- More complex sentence structures. In the following example, where the input sentence has multiple clauses, each with trigger and arguments (such as when triggers are partial or elided). Our model is capable of also extracting "Tom" as another ENTITY of the CONTACT.MEET event.

  > [She]$_{\text{ENTITY}}$ **visited** the store and [Tom]$_{\text{ENTITY}}$ did too.

  But in the second example, when there is a higher-order event expressed spanning events in nested clauses:

  > Canadian authorities arrested two Vancouver-area men on Friday and charged them in the **deaths** of [329 passengers and crew members of an Air-India Boeing 747 that blew up over the Irish Sea in 1985, en route from Canada to London]$_{\text{VICTIM}}$.

  Our model did not extract the entire VICTIM correctly, which proves the difficulty of handling complex clauses structures.
- Lack of reasoning with document-level context. In sentence "MCI must now seize additional assets owned by Ebbers, to secure the **loan**." There is a TRANSFER-MONEY event triggered by loan, with MCI being the GIVER and Ebbers being the RECIPIENT. In the previous paragraph, it's mentioned that "Ebbers failed to make repayment of certain amount of money on the loan from MCI." Without this context, it is hard to determine that Ebbers should be the recipient of the loan.
- Lack of knowledge for obtaining exact boundary for the argument span. For example, in "Negotiations between Washington and Pyongyang on their nuclear dispute have been set for April 23 in Beijing ...", for the ENTITY role, two argument spans should be extracted ("Washington" and "Pyongyang"). While our framework predicts the entire "Washington and Pyongyang" as the argument span. Although there's an overlap between the prediction and gold-data, the model gets no credit for it.
- Data and lexical sparsity. In the following two examples, our model fails to detect the triggers of type END-POSITION. "Minister Tony Blair said **ousting** Saddam Hussein now was key to solving similar crises." "There's no indication if Erdogan would **purge** officials who opposed letting in the troops." It's partially due to they were not seen during training as trigger words. "ousting" a rare word and is not in the tokenizers' vocabulary. Purely inferring from the sentence context is hard for the purpose.

5 Related Work

Event Extraction Most event extraction research has focused on the 2005 Automatic Content Extraction (ACE) sentence-level event task (Walker et al., 2006). In recent years, continuous representations from convolutional neural network (Nguyen and Grishman, 2015; Chen et al., 2015) and recurrent neural network (Nguyen et al., 2016) have been proved to help substantially for the pipeline classifiers. To mitigate the effect of error propagation, joint models have been proposed for event extraction, Yang and Mitchell (2016) consider structural dependencies between events and entities. It requires heavy feature engineering to capture discriminative information. Nguyen and Nguyen (2019) propose a multitask model that performs entity recognition, trigger detection and argument role prediction by sharing BiGRU hidden representations. Zhang et al. (2019a) utilizes a neural transition-based extraction framework (Zhang and Clark, 2011), which requires specially designed transition actions, which still requires recognizing entities during decoding, though entity recognition and argument role prediction are done jointly.

These methods generally perform **trigger detection** → **entity recognition** → **argument role assignment** during decoding. Different from the works above, our framework completely bypasses the entity recognition stage (thus no annotation resources for NER needed), and directly tackles event argument extraction. Also related to our work includes Wadden et al. (2019), they model the entity/argument spans (with start and end offset) instead of labeling with BIO scheme. Different from our work, their learned span representations are later used to predict the entity/argument type. While our QA model directly extract the spans for certain argument role type. Contextualized representations produced by pre-trained language models (Peters et al., 2018; Devlin et al., 2019) have been proved to be helpful for event extraction (Zhang et al., 2019b; Wadden et al., 2019) and question answering (Rajpurkar et al., 2016). The attention mechanism helps capture relationships between tokens in question and input sequence. We use BERT in our framework for capturing semantic relationship between question and input sentence.

Machine Reading Comprehension (MRC) Span-based MRC tasks involve extracting a span from a paragraph (Rajpurkar et al., 2016) or multiple paragraphs (Joshi et al., 2017; Kwiatkowski et al., 2019). Recently, there have been explorations on formulating NLP tasks as a question answering problem. McCann et al. (2018) propose natural language decathlon challenge (decaNLP), which consists of ten tasks (e.g., machine translation, summarization, question answering, etc.) They cast all tasks as question answering over a context and propose a general model for this. In the information extraction literature, Levy et al. (2017) propose the zero-shot relation extraction task and reduce the task to answering crowd-sourced reading comprehension questions. Li et al. (2019) casts entity-relation extraction as a multi-turn question answering task. Their questions lack diversity and naturalness. For example for the PART-WHOLE relation, the template questions is "find Y that belongs to X", where X is instantiated with the pre-given entity. The follow-up work from Li et al. (2020) propose better query strategies incorporating synonyms and examples for named entity recognition. Different from the works above, we focus on the more complex event extraction task, which involves both trigger detection and argument extraction. Our generated questions for extracting event arguments are more natural (incorporating descriptions from annotation guidelines) and leverage trigger information.

Question Generation To generate question templates 2&3 (Type + Role question and annotation guideline based question) which are more natural, we draw insights from literature of automatic rule-based question generation (Heilman and Smith, 2010). Heilman (2011) propose to use linguistically motivated rules for WH word (question phrase) selection. In their more general case of question generation from sentences, answer phrases can be noun phrases, prepositional phrases, or subordinate clauses. Complicated rules are designed with help from superTagger (Ciaramita and Altun, 2006). In our case, event arguments are mostly noun phrases and the rules are simpler – "who" for person, "where" for place and "what" for all other types of entities. We sample around 10 examples from the development set to determine the entity type of each argument role.

In the future, it is interesting to investigate how to utilize machine learning-based question generation method (Du et al., 2017), which would be more beneficial for schema/ontology containing a large number of event argument types.

6 Conclusion

In this paper, we introduce a new paradigm for event extraction based on question answering. We investigate how the question generation strategies affect the performance of our framework on both trigger detection and argument span extraction, and find that more natural questions lead to better performance. Our framework outperforms prior works on the ACE 2005 benchmark, and is capable of extracting event arguments of roles not seen at training time. For future work, it would be interesting to try incorporating broader context (e.g., paragraph/document-level context (Ji and Grishman, 2008; Huang and Riloff, 2011; Du and Cardie, 2020) in our methods to improve the accuracy of the predictions.

Acknowledgments

We thank the anonymous reviewers and Heng Ji for helpful suggestions. This research is based on work supported in part by DARPA LwLL Grant FA8750-19-2-0039.

References

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.

Massimiliano Ciaramita and Yasemin Altun. 2006. Broad-coverage sense disambiguation and information extraction with a supersense sequence tagger. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, pages 594–602.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. 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, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

George R Doddington, Alexis Mitchell, Mark A Przybocki, Lance A Ramshaw, Stephanie M Strassel, and Ralph M Weischedel. 2004. The automatic content extraction (ace) program-tasks, data, and evaluation. In *Lrec*, volume 2, page 1. Lisbon.

Xinya Du and Claire Cardie. 2020. Document-level event role filler extraction using multi-granularity contextualized encoding. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8010–8020, Online. Association for Computational Linguistics.

Xinya Du, Junru Shao, and Claire Cardie. 2017. Learning to ask: Neural question generation for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1342–1352, Vancouver, Canada. Association for Computational Linguistics.

Ralph Grishman and Beth Sundheim. 1996. Message Understanding Conference- 6: A brief history. In *COLING 1996 Volume 1: The 16th International Conference on Computational Linguistics*.

Michael Heilman. 2011. *Automatic factual question generation from text*. Ph.D. thesis, Ph. D. thesis, Carnegie Mellon University.

Michael Heilman and Noah A Smith. 2010. Good question! statistical ranking for question generation. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 609–617.

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. 2011. Peeling back the layers: Detecting event role fillers in secondary contexts. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 1137–1147, Portland, Oregon, USA. Association for Computational Linguistics.

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

Mandar Joshi, Eunsol Choi, Daniel Weld, and Luke Zettlemoyer. 2017. TriviaQA: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1601–1611, Vancouver, Canada. Association for Computational Linguistics.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, et al. 2019. Natural questions: a benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:453–466.

Omer Levy, Minjoon Seo, Eunsol Choi, and Luke Zettlemoyer. 2017. Zero-shot relation extraction via reading comprehension. In *Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017)*, pages 333–342, Vancouver, Canada. 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.

Xiaoya Li, Jingrong Feng, Yuxian Meng, Qinghong Han, Fei Wu, and Jiwei Li. 2020. A unified MRC framework for named entity recognition. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5849–5859, Online. Association for Computational Linguistics.

Xiaoya Li, Fan Yin, Zijun Sun, Xiayu Li, Arianna Yuan, Duo Chai, Mingxin Zhou, and Jiwei Li. 2019.

Entity-relation extraction as multi-turn question answering. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1340–1350, Florence, Italy. 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.

(LDC) Linguistic Data Consortium. 2005. English annotation guidelines for events. *https://www.ldc.upenn.edu/sites/ www.ldc.upenn.edu/files/ english-events-guidelines-v5.4.3.pdf*.

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.

Bryan McCann, Nitish Shirish Keskar, Caiming Xiong, and Richard Socher. 2018. The natural language decathlon: Multitask learning as question answering. *arXiv preprint arXiv:1806.08730*.

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.

Thien Huu Nguyen and Ralph Grishman. 2015. Event detection and domain adaptation with 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 2: Short Papers)*, pages 365–371.

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

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana. Association for Computational Linguistics.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, Austin, Texas. Association for Computational Linguistics.

Ellen Riloff. 1996. Automatically generating extraction patterns from untagged text. In *Proceedings of the national conference on artificial intelligence*, pages 1044–1049.

Lei Sha, Feng Qian, Baobao Chang, and Zhifang Sui. 2018. Jointly extracting event triggers and arguments by dependency-bridge rnn and tensor-based argument interaction. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Beth M Sundheim. 1992. Overview of the fourth message understanding evaluation and conference. In *Proceedings of the 4th conference on Message understanding*, pages 3–21. Association for Computational Linguistics.

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. *Linguistic Data Consortium, Philadelphia*, 57.

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.

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 28th International Joint Conference on Artificial Intelligence*, pages 5422–5428. AAAI Press.

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

Yue Zhang and Stephen Clark. 2011. Syntactic processing using the generalized perceptron and beam search. *Computational linguistics*, 37(1):105–151.

A Questions Based on Annotation Guidelines

Questions based on annotation guidelines for each argument role.

Event Type	Argument Role	Question
Business.Declare-Bankruptcy	Org Place	What declare bankruptcy? Where the event takes place?
Business.End-Org	Org Place	What is ended? Where the event takes place?
Business.Merge-Org	Org	What is merged?
Business.Start-Org	Org Place Agent	What is started? Where the event takes place? Who is the founder?
Conflict.Attack	Place Target Attacker Instrument Victim	Where the event takes place? Who is the target? Who is the attacking agent? What is the instrument used? Who is the victim?
Conflict.Demonstrate	Entity Place	Who is demonstrating agent? Where the event takes place?
Contact.Meet	Entity Place	Who is meeting? Where the event takes place?
Contact.Phone-Write	Entity Place	Who is communicating agents? Where the event takes place?
Justice.Acquit	Defendant Adjudicator	Who is the defendant? What is the judge?
Justice.Appeal	Adjudicator Plaintiff Place	What is the judge? What is the plaintiff? Where the event takes place?
Justice.Arrest-Jail	Person Agent Place	Who is jailed? Who is the jailor? Where the event takes place?
Justice.Charge-Indict	Adjudicator Defendant Prosecutor Place	What is the judge? Who is the defendant? Who is the prosecuting agent? Where the event takes place?
Justice.Convict	Defendant Adjudicator Place	Who is the defendant? What is the judge? Where the event takes place?
Justice.Execute	Place Agent Person	Where the event takes place? Who carry out the execution? Who was executed?
Justice.Extradite	Origin Destination Agent	What is original location of the person being extradited? Where the person is extradited to? Who is the extraditing agent?
Justice.Fine	Entity Adjudicator Place	What is fined? What is the judge? Where the event takes place?
Justice.Pardon	Adjudicator Place Defendant	What is the judge? Where the event takes place? Who is the defendant?
Justice.Release-Parole	Entity Person Place	Who will do the release? Who is released? Where the event takes place?
Justice.Sentence	Defendant Adjudicator Place	Who is the defendant? What is the judge? Where the event takes place?

Justice.Sue	Plaintiff	What is the plaintiff?
	Defendant	Who is the defendant?
	Adjudicator	What is the judge?
	Place	Where the event takes place?
Justice.Trial-Hearing	Defendant	Who is the defendant?
	Place	Where the event takes place?
	Adjudicator	What is the judge?
	Prosecutor	Who is the prosecuting agent?
Life.Be-Born	Place	Where the event takes place?
	Person	Who is born?
Life.Die	Victim	Who died?
	Agent	Who is the killer?
	Place	Where the event takes place?
	Instrument	What is the instrument used?
Life.Divorce	Person	Who are divorced?
	Place	Where the event takes place?
Life.Injure	Victim	Who is victim?
	Agent	Who is the attacking agent?
	Place	Where the event takes place?
	Instrument	What is the instrument used?
Life.Marry	Person	Who are married?
	Place	Where the event takes place?
Movement.Transport	Vehicle	What is the vehicle used?
	Artifact	What is being transported?
	Destination	Where the transporting is directed?
	Agent	Who is responsible for the transport event?
	Origin	Where the transporting originated?
Personnel.Elect	Person	Who is elected?
	Entity	Who voted?
	Place	Where the event takes place?
Personnel.End-Position	Entity	Who is the employer?
	Person	Who is the employee?
	Place	Where the event takes place?
Personnel.Nominate	Person	Who is nominated?
	Agent	Who is the nominating agent?
Personnel.Start-Position	Person	Who is the employee?
	Entity	Who is the employer?
	Place	Where the event takes place?
Transaction.Transfer-Money	Giver	Who is the donating agent?
	Recipient	Who is the recipient?
	Beneficiary	Who benefits from the transfer?
	Place	Where the event takes place?
Transaction.Transfer-Ownership	Buyer	Who is the buying agent?
	Artifact	What was bought?
	Seller	Who is the selling agent?
	Place	Where the event takes place?
	Beneficiary	Who benefits from the transaction?