# Query and Extract: Refining Event Extraction as Type-oriented Binary Decoding

**Sijia Wang**[1], **Mo Yu**[2], **Shiyu Chang**[3], **Lichao Sun**[4], **Lifu Huang**[1]
[1]Virginia Tech [2]WeChat AI [3]University of California Santa Barbara [4]Lehigh University
[1]{sijiawang,lifuh}@vt.edu, [2]moyumyu@tencent.com
[3]chang87@ucsb.edu, [4]lis221@lehigh.edu

## Abstract

Event extraction is typically modeled as a multi-class classification problem where event types and argument roles are treated as atomic symbols. These approaches are usually limited to a set of pre-defined types. We propose a novel event extraction framework that uses event types and argument roles as natural language queries to extract candidate triggers and arguments from the input text. With the rich semantics in the queries, our framework benefits from the attention mechanisms to better capture the semantic correlation between the event types or argument roles and the input text. Furthermore, the query-and-extract formulation allows our approach to leverage all available event annotations from various ontologies as a unified model. Experiments on ACE and ERE demonstrate that our approach achieves state-of-the-art performance on each dataset and significantly outperforms existing methods on zero-shot event extraction.[1]

## 1 Introduction

Event extraction (Grishman, 1997; Chinchor and Marsh, 1998; Ahn, 2006) is a task to identify and type event triggers and participants from natural language text. As shown in Figure 1, *married* and *left* are triggers of two event mentions of the *Marry* and *Transport* event types respectively. Two arguments are involved in the *left* event mention: *she* is an *Artifact*, and *Irap* is the *Destination*.

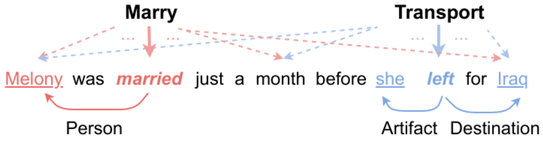

Figure 1: An example of event annotation.

Traditional studies usually model event extraction as a multi-class classification problem (McClosky et al., 2011; Li et al., 2013; Chen et al., 2015; Yang and Mitchell, 2016; Nguyen et al., 2016; Lin et al., 2020), where a set of event types are first defined, and then supervised machine learning approaches will detect and classify each candidate event mention or argument into one of the target types. However, each event type or argument role is treated as an atomic symbol, ignoring their rich semantics in these approaches. Several studies explore the semantics of event types by leveraging the event type structures (Huang et al., 2018), seed event mentions (Bronstein et al., 2015; Lai and Nguyen, 2019), or question answering (QA) (Du and Cardie, 2020; Liu et al., 2020). However, these approaches are still designed for and thus limited to a single target event ontology[2], such as ACE (Consortium, 2005) or ERE (Song et al., 2015).

With the existence of multiple ontologies and the challenge of handling new emerging event types, it is necessary to study event extraction approaches that are generalizable and can use all available training data from distinct event ontologies.[3]

To this end, we propose a new event extraction framework following a query-and-extract paradigm. Our framework represents event types and argument roles as natural language queries with rich semantics. The queries are then used to extract the corresponding event triggers and arguments by leveraging our proposed attention mechanism to capture their interactions with input texts. Specifically, (1) for trigger detection, we formulate each event type as a query based on its type name and a short list of prototype triggers, and make **binary decoding** of each token based on its query-aware

[1] Our code is open sourced at https://github.com/VT-NLP/Event_Query_Extract for reproduction purpose.

[2] An ontology is defined as a collection of event types and argument roles for a particular domain (Brown et al., 2017; Song et al., 2015).

[3] For argument extraction, the QA-based approaches have certain potential to generalize to new ontologies, but require high-quality template questions. As shown in our experiments, their generalizability is limited compared to ours.

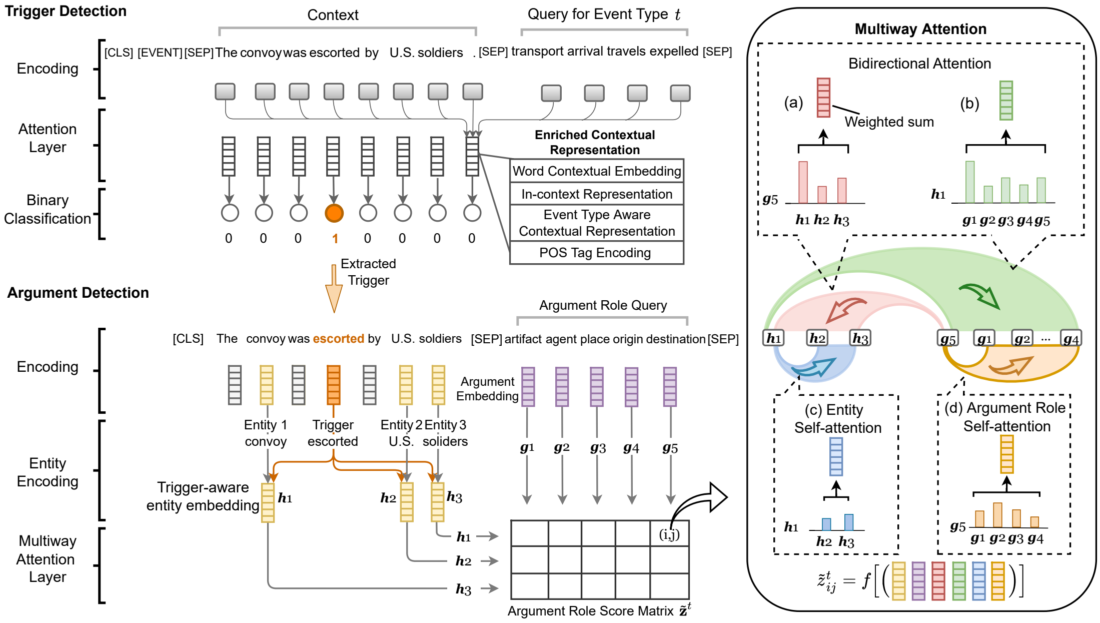
Trigger Detection
Context
Query for Event Type $t$
Encoding
[CLS] [EVENT] [SEP] The convoy was escorted by U.S. soldiers . [SEP] transport arrival travels expelled [SEP]
Attention Layer
Enriched Contextual Representation
Word Contextual Embedding
In-context Representation
Event Type Aware Contextual Representation
POS Tag Encoding
Binary Classification
0 0 0 1 0 0 0 0
Extracted Trigger
Argument Detection
Argument Role Query
Encoding
[CLS] The convoy was escorted by U.S. soldiers [SEP] artifact agent place origin destination [SEP]
Argument Embedding
Entity 1 convoy
Trigger escorted
Entity 2 U.S.
Entity 3 soliders
$g_1$ $g_2$ $g_3$ $g_4$ $g_5$
Entity Encoding
Trigger-aware entity embedding
$h_1$ $h_2$ $h_3$
Multiway Attention Layer
(i,j)
Argument Role Score Matrix $\tilde{z}^t$
Multiway Attention
Bidirectional Attention
(a)
Weighted sum
$g_5$
$h_1$ $h_2$ $h_3$
(b)
$h_1$
$g_1$ $g_2$ $g_3$ $g_4$ $g_5$
$h_1$ $h_2$ $h_3$ $g_5$ $g_1$ $g_2$ ... $g_4$
(c) Entity Self-attention
$h_1$
$h_2$ $h_3$
(d) Argument Role Self-attention
$g_5$
$g_1$ $g_2$ $g_3$ $g_4$
$\tilde{z}^t_{ij} = f[(\ )]$

Figure 2: Architecture overview. Each cell in Argument Role Score Matrix indicates the probabilities of an entity being labeled with an argument role. The arrows in Multiway Attention module show four attention mechanisms: (a) entity to argument roles, (b) argument role to entities, (c) entity to entities, (d) argument role to argument roles.

embedding; (2) for argument extraction, we put together all argument roles defined under each event type as a query, followed by a multiway attention mechanism to extract all arguments of each event mention with **one-time encoding**, with each argument predicted as **binary decoding**.

Our approach can naturally handle various ontologies as a unified model – compared to previous studies (Nguyen and Grishman, 2016; Wadden et al., 2019; Lin et al., 2020), our binary decoding mechanism directly works with any event type or argument role represented as natural language queries, thus effectively leveraging cross-ontology event annotations and making zero-shot predictions. Moreover, compared with the QA-based methods (Du and Cardie, 2020; Liu et al., 2020; Li et al., 2020a) that can also conduct zero-shot argument extraction, our approach does not require creating high-quality questions for argument roles or multi-time encoding for different argument roles separately, thus being more accurate and efficient.

We evaluate our approach on two public benchmark datasets, ACE and ERE, and demonstrate state-of-the-art performance in the standard supervised event extraction and the challenging transfer learning settings that generalize to new event types and ontologies. Notably, on zero-shot transfer to new event types, our approach outperforms a strong baseline by 16% on trigger detection and 26% on argument detection. The overall contributions of our work are:

- We refine event extraction as a query-and-extract paradigm, which is more generalizable and efficient than previous top-down classification or QA-based approaches.
- We design a new event extraction model that leverages rich semantics of event types and argument roles, improving accuracy and generalizability.
- We establish new state-of-the-art performance on ACE and ERE in supervised and zero-shot event extraction and demonstrate our framework as an effective unified model for cross ontology transfer.

## 2 Our Approach

As Figure 2 shows, given an input sentence, we first identify the candidate triggers for each event type by taking it as a query to the sentence. Each event type, such as *Attack*, is represented with a natural language text, including its type name and a shortlist of prototype triggers, such as *invaded* and *airstrikes*, which are selected from the training examples. Then, we concatenate the input sentence with the event type query, encode them with a pre-trained BERT encoder (Devlin et al., 2019), compute the attention distribution over the sequential representation of the event type query for each input token, and finally classify each token into a binary label, indicating it as a trigger candidate of the specific event type or not.

To extract the arguments for each candidate trigger, we follow a similar strategy and take the set of pre-defined argument roles for its corresponding event type as a query to the input sentence. We use another BERT encoder to learn the contextual representations for the input sentence and the query of the argument roles. Then, we take each entity of the input sentence as a candidate argument and compute the semantic correlation between entities and argument roles with multiway attention, and finally classify each entity into a binary label in terms of each argument role.

### 2.1 Trigger Detection

**Event Type Representation** A simple and intuitive way of representing an event type is to use the type name. However, the type name itself cannot accurately represent the semantics of the event type due to the ambiguity of the type name and the variety of the event mentions of each type. For example, *Meet* can refer to *an organized event* or an action of *getting together* or *matching*. Inspired by previous studies (Bronstein et al., 2015; Lai and Nguyen, 2019), we use a short list of prototype triggers to enrich the semantics of each event type.

Specifically, for each event type $t$, we collect a set of annotated triggers from the training examples. For each unique trigger word, we compute its frequency from the whole training dataset as $f_o$ and its frequency of being tagged as an event trigger of type $t$ as $f_t$, and then obtain a probability $f_t/f_o$, which will be used to sort all the annotated triggers for event type $t$. We select the top-$K$[4] ranked words as prototype triggers $\{\tau_1, \tau_2, \ldots, \tau_K\}$.

Finally, each event type will be represented with a natural language sequence of words, consisting of its type name and the list of prototype triggers $T = \{t, \tau_1^t, \tau_2^t, \ldots, \tau_K^t\}$. Taking the event type *Attack* as an example, we finally represent it as *Attack invaded airstrikes overthrew ambushed*.

**Context Encoding** Given an input sentence $W = \{w_1, w_2, \ldots, w_N\}$, we take each event type $T = \{t, \tau_1^t, \tau_2^t, \ldots, \tau_K^t\}$ as a query to extract the corresponding event triggers. Specifically, we first concatenate them into a sequence as follows:

$$\text{[CLS][EVENT][SEP]} \; w_1 \; ... \; w_N \; \text{[SEP]} \\ t \; \tau_1^t \; ... \; \tau_K^t \; \text{[SEP]}$$

where [SEP] is a separator from the BERT encoder (Devlin et al., 2019). Following (Liu et al., 2020), we use a special symbol [EVENT] to emphasis the trigger detection task.

Then we use a pre-trained BERT encoder to encode the whole sequence and get contextual representations for the input sentence $\boldsymbol{W} = \{\boldsymbol{w}_0, \boldsymbol{w}_2, ..., \boldsymbol{w}_N\}$ as well as the event type $\boldsymbol{T} = \{\boldsymbol{t}, \boldsymbol{\tau}_0^t, \boldsymbol{\tau}_1^t, ..., \boldsymbol{\tau}_K^t\}$.[5]

**Enriched Contextual Representation** Given a query of each event type, we aim to automatically extract corresponding event triggers from the input sentence. To achieve this goal, we need to capture the semantic correlation of each input token to the event type. Thus we apply attention mechanism to learn a weight distribution over the sequence of contextual representations of the event type query and get an event type aware contextual representation for each token:

$$\boldsymbol{A}_i^T = \frac{1}{T}\sum_{j=1}^{|T|} \alpha_{ij} \cdot \boldsymbol{T}_j \; ,$$

$$\alpha_{ij} = \cos(\boldsymbol{w}_i, \; \boldsymbol{T}_j) \; ,$$

where $\boldsymbol{T}_j$ is the contextual representation of the $j$-th token in the sequence $T = \{t, \tau_1^t, \tau_2^t, \ldots, \tau_K^t\}$. $\cos(\cdot)$ is the cosine similarity function between two vectors. $\boldsymbol{A}_i^T$ denotes the event type $t$ aware contextual representation of token $w_i$.

In addition, the prediction of event triggers also depends on the occurrence of a particular context. For example, according to ACE event annotation guidelines (Consortium, 2005), to qualify as a *Meet* event, the meeting must be known to be "*face-to-face and physically located somewhere*". To capture such context information, we further apply in-context attention to capture the meaningful contextual words for each input token:

$$\boldsymbol{A}_i^W = \frac{1}{N}\sum_{j=1}^{N} \tilde{\alpha}_{ij} \cdot \boldsymbol{w}_j \; ,$$

$$\tilde{\alpha}_{ij} = \rho(\boldsymbol{w}_i, \; \boldsymbol{w}_j) \; ,$$

where $\rho(.)$ is the attention function and is computed as the average of the self-attention weights from the last $m$ layers of BERT.[6]

[4] In our experiments, we set $K = 4$.

[5] We use bold symbols to denote vectors.

[6] We set $m$ as 3 as it achieved the best performance.

**Event Trigger Detection** With the event type oriented attention and in-context attention mechanisms, each token $w_i$ from the input sentence will obtain two enriched contextual representations $\boldsymbol{A}_i^W$ and $\boldsymbol{A}_i^T$. We concatenate them with the original contextual representation $\boldsymbol{w}_i$ from the BERT encoder, and classify it into a binary label, indicating it as a candidate trigger of event type $t$ or not:

$$\tilde{\boldsymbol{y}}_i^t = \boldsymbol{U}_o \cdot ([\boldsymbol{w}_i;\ \boldsymbol{A}_i^W;\ \boldsymbol{A}_i^T; \boldsymbol{P}_i]) \ ,$$

where $[;]$ denotes concatenation operation, $\boldsymbol{U}_o$ is a learnable parameter matrix for event trigger detection, and $\boldsymbol{P}_i$ is the one-hot part-of-speech (POS) encoding of word $w_i$. We optimize the following objective for event trigger detection

$$\mathcal{L}_1 = -\frac{1}{|\mathcal{T}||\mathcal{N}|} \sum_{t \in \mathcal{T}} \sum_{i=1}^{|\mathcal{N}|} \boldsymbol{y}_i^t \cdot \log \tilde{\boldsymbol{y}}_i^t \ ,$$

where $\mathcal{T}$ is the set of target event types and $\mathcal{N}$ is the set of tokens from the training dataset. $\boldsymbol{y}_i^t$ denotes the groundtruth label vector.

## 2.2 Event Argument Extraction

After detecting event triggers for each event type, we further extract their arguments based on the pre-defined argument roles of each event type.

**Context Encoding** Given a candidate trigger $r$ from the sentence $W = \{w_1, w_2, \ldots, w_N\}$ and its event type $t$, we first obtain the set of predefined argument roles for event type $t$ as $G^t = \{g_1^t, g_2^t, ..., g_D^t\}$. To extract the corresponding arguments for $r$, similar as event trigger detection, we take all argument roles $G^t$ as a query and concatenate them with the original input sentence

$$\text{[CLS] } w_1 \ w_2 \ ... \ w_N \text{ [SEP] } g_1^t \ g_2^t \ ... \ g_D^t \text{ [SEP]}$$

where we use the last [SEP] separator to denote *Other* category, indicating the entity is not an argument. Then, we encode the whole sequence with another pre-trained BERT encoder (Devlin et al., 2019) to get the contextual representations of the sentence $\tilde{\boldsymbol{W}} = \{\tilde{\boldsymbol{w}}_0, \tilde{\boldsymbol{w}}_2, ..., \tilde{\boldsymbol{w}}_N\}$, and the argument roles $\boldsymbol{G}^t = \{\boldsymbol{g}_0^t, \boldsymbol{g}_1^t, ..., \boldsymbol{g}_D^t, \boldsymbol{g}_{\text{[Other]}}^t\}$.

As the candidate trigger $r$ may span multiple tokens within the sentence, we obtain its contextual representation $\boldsymbol{r}$ as the average of the contextual representations of all tokens within $r$. In addition, as the arguments are usually detected from the entities of sentence $W$, we apply a BERT-CRF model, which is optimized on the same training set as event extraction to identify the entities $E = \{e_1, e_2, ..., e_M\}$. As each entity may also span multiple tokens, following the same strategy, we average the contextual representations of all tokens within each entity and obtain the entity contextual representations as $\boldsymbol{E} = \{\boldsymbol{e}_1, \boldsymbol{e}_2, ..., \boldsymbol{e}_M\}$.

**Multiway Attention** Given a candidate trigger $r$ of type $t$ and an entity $e_i$, for each argument role $g_j^t$, we need to determine whether the underlying relation between $r$ and $e_i$ corresponds to $g_j^t$ or not, namely, whether $e_i$ plays the argument role of $g_j^t$ in event mention $r$. To do this, for each $e_i$, we first obtain a trigger-aware entity representation as

$$\boldsymbol{h}_i = \boldsymbol{U}_h \cdot ([\boldsymbol{e}_i;\ \boldsymbol{r};\ \boldsymbol{e}_i \circ \boldsymbol{r}]) \ ,$$

where $\circ$ denotes element-wise multiplication operation. $\boldsymbol{U}_h$ is a learnable parameter matrix.

In order to determine the semantic correlation between each argument role and each entity, we first compute a similarity matrix $\boldsymbol{S}$ between the trigger-aware entity representations $\{\boldsymbol{h}_1, \boldsymbol{h}_2, ..., \boldsymbol{h}_M\}$ and the argument role representations $\{\boldsymbol{g}_0^t, \boldsymbol{g}_1^t, ..., \boldsymbol{g}_D^t\}$

$$S_{ij} = \frac{1}{\sqrt{d}} \sigma(\boldsymbol{h}_i,\ \boldsymbol{g}_j^t) \ ,$$

where $\sigma$ denotes dot product operator, $d$ denotes embedding dimension of $\boldsymbol{g}^t$, and $S_{ij}$ indicates the semantic correlation of entity $e_i$ to a particular argument role $g_j^t$ given the candidate trigger $r$.

Based on the correlation matrix $\boldsymbol{S}$, we further apply a bidirectional attention mechanism to get an argument role aware contextual representation for each entity and an entity-aware contextual representation for each argument role as follows:

$$\boldsymbol{A}_i^{e2g} = \sum_{j=1}^{D} \boldsymbol{S}_{ij} \cdot \boldsymbol{g}_j^t \ ,$$

$$\boldsymbol{A}_j^{g2e} = \sum_{i=1}^{M} \boldsymbol{S}_{ij} \cdot \boldsymbol{h}_i \ .$$

In addition, previous studies (Hong et al., 2011; Li et al., 2013; Lin et al., 2020) have revealed that the underlying relations among entities or argument roles are also important to extract the arguments. For example, if entity $e_1$ is predicted as *Attacker* of an *Attack* event and $e_1$ is *located in* another entity $e_2$, it's very likely that $e_2$ plays an argument role of *Place* for the *Attack* event. To capture the underlying relations among the entities, we further compute the self-attention among them

$$\mu_{ij} = \frac{1}{\sqrt{d}}\sigma(\boldsymbol{h}_i,\, \boldsymbol{h}_j)\ , \quad \tilde{\boldsymbol{\mu}}_i = \mathrm{Softmax}(\boldsymbol{\mu}_i)\ ,$$

$$\boldsymbol{A}_i^{e2e} = \sum_{j=1}^{M} \tilde{\mu}_{ij} \cdot \boldsymbol{h}_j\ .$$

Similarly, to capture the underlying relations among argument roles, we also compute the self-attention among them

$$v_{jk} = \frac{1}{\sqrt{d}}\sigma(\boldsymbol{g}_j^t,\, \boldsymbol{g}_k^t)\ , \quad \tilde{\boldsymbol{v}}_j = \mathrm{Softmax}(\boldsymbol{v}_j)\ ,$$

$$\boldsymbol{A}_j^{g2g} = \sum_{k=1}^{D} \tilde{v}_{jk} \cdot \boldsymbol{g}_k^t\ .$$

**Event Argument Predication** Finally, for each candidate event trigger $r$, we determine whether an entity $e_i$ plays an argument role of $g_j^t$ in the event mention by classifying it into a binary class:

$$\tilde{\boldsymbol{z}}_{ij}^t = \boldsymbol{U}_a \cdot ([\boldsymbol{h}_i;\, \boldsymbol{g}_j^t;\, \boldsymbol{A}_i^{e2g};\, \boldsymbol{A}_j^{g2e};\, \boldsymbol{A}_i^{e2e};\, \boldsymbol{A}_j^{g2g}]),$$

where $\boldsymbol{U}_a$ is a learnable parameter matrix for argument extraction. And $\tilde{z}^t$ is argument role score matrix for event type $t$. The training objective is to minimize the following loss function:

$$\mathcal{L}_2 = -\frac{1}{|\mathcal{A}||\mathcal{E}|}\sum_{j=1}^{|\mathcal{A}|}\sum_{i=1}^{|\mathcal{E}|} \boldsymbol{z}_{ij} \log \tilde{\boldsymbol{z}}_{ij}\ ,$$

where $\mathcal{A}$ denotes the collection of possible argument roles, and $\mathcal{E}$ is the set of entities we need to consider for argument extraction. $\boldsymbol{z}_{ij}$ denotes the ground truth label vector. During test, an entity will be labeled as a non-argument if the prediction for *Other* category is 1. Otherwise, it can be labeled with multiple argument roles.

# 3 Experiments

## 3.1 Experimental Setup

We perform experiments on two public benchmarks, ACE05-E$^{+}$[7] and ERE-EN (Song et al., 2015)[8]. ACE defines 33 event types while ERE includes 38 types, among which there are 31 overlapped event types. We use the same data split of ACE and ERE as (Wadden et al., 2019; Lin et al., 2020; Du and Cardie, 2020) for supervised event extraction. For zero-shot event extraction, we use the top-10 most popular event types in ACE as seen types for training and treat the remaining 23 event types as unseen for testing, following Huang et al. (2018). In our experiments, we use random seeds and report averaged scores of each setting. More details regarding the data statistics and evaluation are shown in Appendix A.

We further design two more challenging and practical settings to evaluate how well the approach could leverage resources from different ontologies: (1) *cross-ontology direct transfer*, where we only use the annotations from ACE or ERE for training and directly test the model on another event ontology. This corresponds to the *domain adaptation setting* in transfer learning literature; (2) *joint-ontology enhancement*, where we take the annotations from both ACE and ERE as the training set and test the approaches on ACE or ERE ontology separately. This corresponds to the *multi-domain learning setting* in transfer learning literature. Intuitively, an approach with good transferability should benefit more from the enhanced training data from other ontologies. We follow the same train/dev/test splits of ACE and ERE as supervised event extraction.

## 3.2 Supervised Event Extraction

Table 1 shows the supervised event extraction results of various approaches on ACE and ERE datasets. Though studies (Yang and Mitchell, 2016; Liu et al., 2020, 2018; Sha et al., 2018; Lai et al., 2020; Veyseh et al., 2020) have been conducted on the ACE dataset, they follow different settings[9], especially regarding whether the Time and Value arguments are considered and whether all Time-related argument roles are viewed as a single role. Following several recent state-of-the-art studies (Wadden et al., 2019; Lin et al., 2020; Du and Cardie, 2020), we do not consider Time and Value arguments. Our approach significantly outperforms most of the previous comparable baseline methods, especially on the ERE dataset[10]. Next, we take BERT_QA_Arg, a QA_based method, as the main baseline as it shares similar ideas to our approach to compare their performance.

[7] `https://catalog.ldc.upenn.edu/LDC2006T06`

[8] Following Lin et al. (2020), we merge LDC2015E29, LDC2015E68, and LDC2015E78 as the ERE dataset.

[9] Many studies did not describe their argument extraction setting in detail.

[10] Appendix E describes several remaining challenges identified from the prediction errors on ACE05 dataset.

| Model | ACE05-E$^{+}$ | | ERE-EN | |
|---|---|---|---|---|
| | Trigger Ext. | Argument Ext. | Trigger Ext. | Argument Ext. |
| DYGIE++ (Wadden et al., 2019) | 67.3$^{*}$ | 42.7$^{*}$ | - | - |
| BERT_QA_Arg (Du and Cardie, 2020) | 70.6$^{*}$ | 48.3$^{*}$ | 57.0 | 39.2 |
| OneIE (Lin et al., 2020) | 72.8 | 54.8 | 57.0 | 46.5 |
| Text2Event (Lu et al., 2021) | 71.8 | 54.4 | 59.4 | 48.3 |
| FourIE (Nguyen et al., 2021) | 73.3 | **57.5** | 57.9 | 48.6 |
| **Our Approach** | **73.6** (0.2) | 55.1 (0.5) | **60.4** (0.3) | **50.4** (0.3) |

Table 1: Event extraction results on ACE05-E$^{+}$ and ERE-EN datasets (F-score, %). $^{*}$ indicates scores obtained from their released codes. The performance of BERT_QA_Arg is lower than that reported in (Du and Cardie, 2020) as they only consider single-token event triggers. Each score of our approach is the mean of three runs and the variance is shown in parenthesis.

Specifically, for trigger detection, all the baseline methods treat the event types as symbols and classify each input token into one of the target types or *Other*. So they heavily rely on human annotations and do not perform well when the annotations are not enough. For example, there are only 31 annotated event mentions for *End_Org* in the ACE05 training dataset, so BERT_QA_Arg only achieves 35.3% F-score. In comparison, our approach leverages the semantic interaction between the input tokens and the event types. Therefore it still performs well when the annotations are limited, e.g., it achieves 66.7% F-score for *End_Org*. In addition, by leveraging the rich semantics of event types, our approach also successfully detects event triggers that are rarely seen in the training dataset, e.g., *ousting* and *purge* of *End-Position*, while BERT_QA_Arg misses all these triggers. A more detailed discussion about the impact of seed triggers is in Appendix B.

For argument extraction, our approach shows more consistent results than baseline methods. For example, in the sentence "*Shalom was to fly on to London for talks with British Prime Minister Tony Blair and Foreign Secretary Jack Straw*", the BERT_QA_Arg method correctly predicts *Tony Blair* and *Jack Straw* as *Entity* arguments of the *Meet* event triggered by *talks*, but misses *Shalom*. However, by employing multiway attention, especially the self-attention among all the entities, our approach can capture their underlying semantic relations, e.g., *Shalom* and *Tony Blair* are two persons to talk, so that it successfully predicts all the three *Entity* arguments for the *Meet* event.

### 3.3 Zero-Shot Event Extraction

As there are no fully comparable baseline methods for zero-shot event extraction, we adapt one of the most recent states of the arts, BERT_QA_Arg (Du and Cardie, 2020), which is expected to have specific transferability due to its QA formulation. However, the original BERT_QA_Arg utilizes a generic query, e.g., "*trigger*" or "*verb*", to classify each input token into one of the target event types or *Other*, thus is not capable of detecting event mentions for any new event types during the test. We adapt the BERT_QA_Arg framework by taking each event type instead of the generic words as a query for event detection. Note that our approach utilizes the event types as queries without prototype triggers for zero-shot event extraction.

| Model | Trigger Ext. | Arg Ext. (GT) |
|---|---|---|
| BERT_QA_Arg$^{\dagger}$ | 31.6 | 17.0 |
| **Our Approach** | **47.8** | **43.0** |

Table 2: Zero-shot F-scores on 23 unseen event types. †: adapted implementation from (Du and Cardie, 2020). GT indicates using gold-standard triggers as input.

As Table 2 shows, our approach significantly outperforms BERT_QA_Arg under zero-shot event extraction, with over 16% F-score gain on trigger detection and 26% F-score gain on argument extraction. Comparing with BERT_QA_Arg, which only relies on the self-attention from the BERT encoder to learn the correlation between the input tokens and the event types or argument roles, our approach further applies multiple carefully designed attention mechanisms over BERT contextual representations to better capture the semantic interaction between event types or argument roles and input tokens, yielding much better accuracy and generalizability.

We further pick 13 unseen event types and analyze our approach's zero-shot event extraction performance on each of them. As shown in Figure 3, our approach performs exceptionally well on

| Source | Target | BERT_QA_Arg$_{\text{multi}}$ | | BERT_QA_Arg$_{\text{binary}}$† | | **Our Approach** | |
|---|---|---|---|---|---|---|---|
| | | Trigger Ext. | Argument Ext. | Trigger Ext. | Argument Ext. | Trigger Ext. | Argument Ext. |
| ERE | ACE | 48.9 (48.9) | 18.5 (18.5) | 50.8 (50.8) | 20.9 (20.9) | 53.9 (52.6) | 30.2 (29.6) |
| ACE | ACE | 70.6 | 48.3 | 72.2 | 50.4 | 73.6 | 55.1 |
| ACE+ERE | ACE | 70.1 | 47.0 | 71.3 | 49.8 | 74.4 | 56.2 |
| ACE | ERE | 47.2 (47.2) | 18.0 (18.0) | 47.2 (45.0) | 17.9 (17.1) | 55.9 (46.3) | 31.9 (26.0) |
| ERE | ERE | 57.0 | 39.2 | 56.7 | 42.9 | 60.4 | 50.4 |
| ACE+ERE | ERE | 57.0 | 38.6 | 54.6 | 37.1 | 63.0 | 52.3 |

Table 3: Cross ontology transfer between ACE and ERE datasets (F-score %). The scores in parenthesis indicate the performance on the ACE and ERE shared event types.

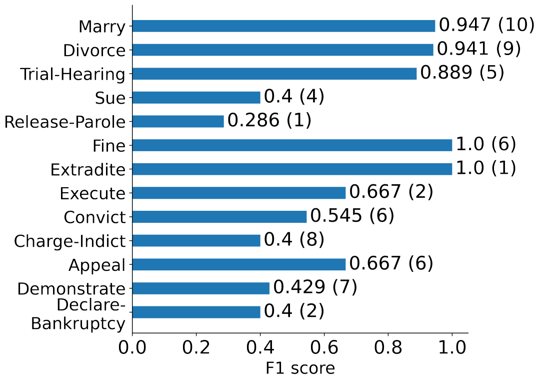

Figure 3: Zero-shot event extraction on each unseen event type. The number in parenthesis indicates # gold event mentions of each unseen type in the test set.

*Marry*, *Divorce*, *Trial-Hearing*, and *Fine*, but worse on *Sue*, *Release-Parole*, *Charge-Indict*, *Demonstrate*, and *Declare-Bankruptcy*, with two possible reasons: first, the semantics of event types, such as *Marry*, *Divorce*, is more straightforward and explicit than other types, such as *Charge-Indict*, *Declare-Bankruptcy*. Thus our approach can better interpret these types. Second, the diversity of the event triggers for some types, e.g., *Divorce*, is much lower than other types, e.g., *Demonstrate*. For example, among the 9 *Divorce* event triggers, there are only 2 unique strings, i.e., *divorce* and *breakdowns*, while there are 6 unique strings among the 7 event mentions of *Demonstrate*.

### 3.4 Cross Ontology Transfer

For cross-ontology transfer, we develop two variations of BERT_QA_Arg as baseline methods: (1) BERT_QA_Arg$_{\text{multi}}$, which is the same as the original implementation and use multi-classification to detect event triggers. (2) BERT_QA_Arg$_{\text{binary}}$, for which we apply the same query adaptation as Section 3.3 and use multiple binary-classification for event detection. For *joint-ontology enhancement*, we combine the training datasets of ACE and ERE and optimize the models from scratch.[11]

Table 3 shows the cross-ontology transfer results in both *direct transfer* and *enhancement* settings. Our approach significantly outperforms the baseline methods under all the settings. Notably, for *direct transfer*, e.g., from ERE to ACE, by comparing the F-scores on the whole test set with the performance on the ACE and ERE shared event types (F-scores shown in parenthesis), our approach not only achieves better performance on the shared event types but also extracts event triggers and arguments for the new event types in ACE. In contrast, the baseline methods hardly extract any events or arguments for the new event types. Moreover, by combining the training datasets of ACE and ERE for *joint-ontology enhancement*, our approach's performance can be further boosted compared with using the annotations of the target event ontology only, demonstrating the superior transfer capability across different ontologies. For example, ACE includes a *Transport* event type while ERE defines two more fine-grained types *Transport-Person* and *Transport-Artifact*. By adding the annotations of *Transport-Person* and *Transport-Artifact* from ERE into ACE, our approach can capture the underlying semantic interaction between *Transport*-related event type queries and the corresponding input tokens and thus yield 1.5% F-score gain on the *Transport* event type of ACE test set. In contrast, both baseline methods fail to be enhanced with additional annotations from a slightly different event ontology without explicitly capturing semantic interaction between event types and input tokens. Appendix C provides a more in-depth comparison between our approach and the baseline approaches.

[11] Another intuitive training strategy is to train the model on the source and target ontologies sequentially. Our pilot study shows that this strategy performs slightly worse.

### 3.5 Ablation Study

We further evaluate the impact of each attention mechanism on event trigger detection and argument extraction. As Table 4 shows, all the attention mechanisms show significant benefit to trigger or argument extraction, especially on the ERE dataset. The Event Type Attention and Multiway Attention show the most effects to trigger and argument extraction, which is understandable as they are designed to capture the correlation between the input texts and the event type or argument role-based queries. We also notice that, without taking entities detected by the BERT-CRF name tagging model as input, but instead considering all the tokens as candidate arguments[12], our approach still shows competitive performance for argument extraction compared with the strong baselines. More ablation studies are discussed in Appendix D.

| | Model | ACE | ERE |
|---|---|---|---|
| Trigger | Our Approach | 73.6 | 60.4 |
| | w/o Seed Trigger | 72.2 | 58.2 |
| | w/o In-Context Attention | 72.3 | 57.9 |
| | w/o Event Type Attention | 71.1 | 56.9 |
| Arg. | Our Approach | 55.1 | 50.4 |
| | w/o Entity Detection | 53.0 | 47.6 |
| | w/o Multiway Attention | 53.4 | 42.8 |
| | w/o Entity Self-attention | 53.7 | 48.3 |
| | w/o Arg Role Self-attention | 54.1 | 47.7 |

Table 4: Results of various ablation studies. Each score is the average of three runs for each experiment.

### 3.6 Pros and Cons of Type-oriented Decoding

The advantages of our type-oriented binary decoding include: (1) it allows the model to better leverage the semantics of event types which have been proved effective for both supervised and zero-shot event extraction; (2) it allows the approach to leverage all available event annotations from distinct ontologies, which is demonstrated in zero-shot event extraction and cross-ontology transfer; (3) in practice, new event types and annotations could emerge incessantly, and it is not possible to always train a model for all the event types. Our approach has the potential to be continuously updated and extract events for any desired event types.

We also admit that binary decoding usually increases the computation cost. We design two strategies to mitigate this issue: (1) More than 69% of the sentences in the training dataset do not contain any event triggers, so we randomly sample 20% of them for training. (2) Our one-time argument encoding and decoding strategies extract all arguments of each event trigger at once. It is more efficient than the previous QA-based approaches, which only extract arguments for one argument role at once. With these strategies, for trigger detection, our approach takes 80% more time for training and 19% less for inference compared with BERT_QA_Arg which relies on multi-class classification, while for argument extraction, our approach takes 36% less time for training and inference than BERT_QA_Arg. Even on a more fine-grained event ontology MAVEN (Wang et al., 2020), which consists of 168 event types, for trigger extraction, our approach roughly takes 20% more time for training and twice the time for inference compared with BERT_QA_Arg, with slightly better performance than the state of the art (Wang et al., 2021)[13].

## 4 Related Work

Traditional event extraction studies (McClosky et al., 2011; Li et al., 2013; Chen et al., 2015; Cao et al., 2015; Feng et al., 2016; Yang and Mitchell, 2016; Nguyen et al., 2016; Zhang et al., 2017; Wadden et al., 2019; Lin et al., 2020; Wang et al., 2021) usually detect event triggers and arguments with multi-class classifiers. Unlike all these methods that treat event types and argument roles as symbols, our approach considers them queries with rich semantics and leverages the semantic interaction between input tokens and each event type or argument role.

Several studies have explored the semantics of event types based on seed event triggers (Bronstein et al., 2015; Lai and Nguyen, 2019; Zhang et al., 2021), event type structures (Huang et al., 2016, 2018), definitions (Chen et al., 2019) and latent representations (Huang and Ji, 2020). However, they can hardly be generalized to argument extraction. Question answering based event extraction (Du and Cardie, 2020; Liu et al., 2020; Li et al., 2020a; Lyu et al., 2021) can take advantage of the semantics of event types and the large-scale question answering datasets. Compared with these methods, there are three different vital designs, making our approach perform and be generalized better than these

[12] We take consecutive tokens predicted with the same argument role as a single argument span.

[13] Our approach achieves 68.8% F-score on MAVEN. We do not discuss more as MAVEN only contains trigger annotations.

QA-based approaches: (1) our approach directly takes event types and argument roles as queries. In contrast, previous QA-based approaches rely on templates or generative modules to create natural language questions. However, it is difficult to find the optimal format of questions for each event type, and many studies (Du and Cardie, 2020; Li et al., 2020b) have shown that MRC or QA models are sensitive to the subtle change of the questions. (2) QA-based approaches can only detect arguments for one argument role at once, while our approach extracts all arguments of an event trigger with one-time encoding and decoding, which is more efficient and leverages the implicit relations among the candidate arguments or argument roles. (3) QA-based approaches rely on span prediction to extract arguments without requiring entity extraction, which could result in more entity boundary errors. Thus we choose to pre-train a name tagging model and use binary decoding over system detected entities.Moreover, it is pretty challenging to fully adapt the event extraction task to the span-based Question Answering. The main reason is that each sentence may contain multiple triggers for a particular event type. Even if we can formalize a question, e.g., "what is the trigger for Attack?" it is difficult for the model to return all the spans of event triggers correctly.

## 5 Conclusion and Future Work

We refine event extraction with a query-and-extract paradigm and design a new framework that leverages rich semantics of event types and argument roles and captures their interactions with input texts using attention mechanisms to extract event triggers and arguments. Experimental results demonstrate that our approach achieves state-of-the-art performance on supervised event extraction and shows prominent accuracy and generalizability to new event types and across ontologies. In the future, we will explore better representations of event types and argument roles and combine them prompt tuning approach further to improve the accuracy and generalizability of event extraction.

## Acknowledgements

We thank the anonymous reviewers and area chair for their valuable time and constructive comments, and the helpful discussions with Zhiyang Xu and Minqian Liu. We also thank the support from the Amazon Research Awards.

## References

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

Ofer Bronstein, Ido Dagan, Qi Li, Heng Ji, and Anette Frank. 2015. Seed-based event trigger labeling: How far can event descriptions get us? 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 372–376.

Susan Brown, Claire Bonial, Leo Obrst, and Martha Palmer. 2017. The rich event ontology. In *Proceedings of the Events and Stories in the News Workshop*, pages 87–97, Vancouver, Canada. Association for Computational Linguistics.

Kai Cao, Xiang Li, Miao Fan, and Ralph Grishman. 2015. Improving event detection with active learning. In *Proceedings of the International Conference Recent Advances in Natural Language Processing*, pages 72–77, Hissar, Bulgaria. INCOMA Ltd. Shoumen, BULGARIA.

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.

Yunmo Chen, Tongfei Chen, Seth Ebner, Aaron Steven White, and Benjamin Van Durme. 2019. Reading the manual: Event extraction as definition comprehension. *arXiv preprint arXiv:1912.01586*.

Nancy Chinchor and Elaine Marsh. 1998. Muc-7 information extraction task definition. In *Proceeding of the seventh message understanding conference (MUC-7), Appendices*, pages 359–367.

(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.

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.

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

Xiaocheng Feng, Lifu Huang, Duyu Tang, Heng Ji, Bing Qin, and Ting Liu. 2016. A language-independent neural network for event detection. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 66–71, Berlin, Germany. Association for Computational Linguistics.

Ralph Grishman. 1997. Information extraction: Techniques and challenges. In *International summer school on information extraction*, pages 10–27. Springer.

Yu Hong, Jianfeng Zhang, Bin Ma, Jianmin Yao, Guodong Zhou, and Qiaoming Zhu. 2011. Using cross-entity inference to improve event extraction. In *Proceedings of the 49th annual meeting of the association for computational linguistics: human language technologies*, pages 1127–1136.

Lifu Huang, Taylor Cassidy, Xiaocheng Feng, Heng Ji, Clare Voss, Jiawei Han, and Avirup Sil. 2016. Liberal event extraction and event schema induction. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 258–268.

Lifu Huang and Heng Ji. 2020. Semi-supervised new event type induction and event detection. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 718–724.

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.

John Lafferty, Andrew McCallum, and Fernando CN Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data.

Viet Dac Lai and Thien Huu Nguyen. 2019. Extending event detection to new types with learning from keywords. *arXiv preprint arXiv:1910.11368*.

Viet Dac Lai, Tuan Ngo Nguyen, and Thien Huu Nguyen. 2020. 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, Online. Association for Computational Linguistics.

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

Qi Li, Heng Ji, 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. 2020b. 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.

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

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

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

Yaojie Lu, Hongyu Lin, Jin Xu, Xianpei Han, Jialong Tang, Annan Li, Le Sun, Meng Liao, and Shaoyi Chen. 2021. Text2Event: Controllable sequence-to-structure generation for end-to-end event extraction. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 2795–2806, Online. Association for Computational Linguistics.

Qing Lyu, Hongming Zhang, Elior Sulem, and Dan Roth. 2021. Zero-shot Event Extraction via Transfer Learning: Challenges and Insights. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 322–332, Online. Association for Computational Linguistics.

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

Minh Van Nguyen, Viet Dac Lai, and Thien Huu Nguyen. 2021. Cross-task instance representation interactions and label dependencies for joint information extraction with graph convolutional networks.

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. 2016. Modeling skip-grams for event detection with convolutional neural networks. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 886–891, Austin, Texas. Association for Computational Linguistics.

Alexandre Passos, Vineet Kumar, and Andrew McCallum. 2014. Lexicon infused phrase embeddings for named entity resolution. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning*, pages 78–86, Ann Arbor, Michigan. Association for Computational Linguistics.

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 *AAAI*.

Zhiyi Song, Ann Bies, Stephanie Strassel, Tom Riese, Justin Mott, Joe Ellis, Jonathan Wright, Seth Kulick, Neville Ryant, and Xiaoyi Ma. 2015. From light to rich ere: annotation of entities, relations, and events. In *Proceedings of the the 3rd Workshop on EVENTS: Definition, Detection, Coreference, and Representation*, pages 89–98.

Amir Pouran Ben Veyseh, Tuan Ngo Nguyen, and Thien Huu Nguyen. 2020. Graph transformer networks with syntactic and semantic structures for event argument extraction. *CoRR*, abs/2010.13391.

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.

Xiaozhi Wang, Ziqi Wang, Xu Han, Wangyi Jiang, Rong Han, Zhiyuan Liu, Juanzi Li, Peng Li, Yankai Lin, and Jie Zhou. 2020. MAVEN: A massive general domain event detection dataset. In *Proceedings of EMNLP 2020*.

Ziqi Wang, Xiaozhi Wang, Xu Han, Yankai Lin, Lei Hou, Zhiyuan Liu, Peng Li, Juanzi Li, and Jie Zhou. 2021. CLEVE: Contrastive Pre-training for Event Extraction. In *Proceedings of ACL-IJCNLP*, pages 6283–6297, Online. Association for Computational Linguistics.

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

Hongming Zhang, Haoyu Wang, and Dan Roth. 2021. Zero-shot Label-aware Event Trigger and Argument Classification. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 1331–1340, Online. Association for Computational Linguistics.

Tongtao Zhang, Spencer Whitehead, Hanwang Zhang, Hongzhi Li, Joseph Ellis, Lifu Huang, Wei Liu, Heng Ji, and Shih-Fu Chang. 2017. Improving event extraction via multimodal integration. In *Proceedings of the 25th ACM international conference on Multimedia*, pages 270–278.

## A Data Statistics and Implementation Details

Table 5 shows the detailed data statics of the training, development and test sets of the ACE05-E+ and ERE datasets. The statistics for the ERE dataset is slightly different from previous work (Lin et al., 2020; Lu et al., 2021) as we consider the event triggers that are annotated with multiple types as different instances while the previous studies just keep one annotated type for each trigger span. For example, in the ERE-EN dataset, a token *"succeeded"* in the sentence *"Chun ruled until 1988 , when he was succeeded by Roh Tae - woo , his partner in the 1979 coup ."* triggers a *End-Position* event of *Chun* and a *Start-Position* of *Roh*. Previous classification based approaches did not predict multiple types for each candidate trigger.

| Dataset | Split | # Events | # Arguments |
|---|---|---|---|
| ACE05-E+ | Train | 4419 | 6605 |
| | Dev | 468 | 757 |
| | Test | 424 | 689 |
| ERE-EN | Train | 7394 | 11576 |
| | Dev | 632 | 979 |
| | Test | 669 | 1078 |

Table 5: Data statistics for ACE2005 and ERE datasets.

**Zero-Shot Event Extraction** To evaluate the transfer capability of our approach, we use the top-10 most popular event types in ACE05 as seen types for training and treat the remaining 23 event types as unseen for testing, following Huang et al. (2018). The top-10 training event types include *Attack*, *Transport*, *Die*, *Meet*, *Sentence*, *Arrest-Jail*, *Transfer-Money*, *Elect*, *Transfer-Ownership*, *End-Position*. We use the same data split as supervised event extraction but only keep the event annotations of the 10 seen types for training and development sets and sample 150 sentences with 120 annotated event mentions for the 23 unseen types from the test set for evaluation.

**Implementation** For fair comparison with previous baseline approaches, we use the same pre-trained `bert-large-uncased` model for fine-tuning and optimize our model with BertAdam. We optimize the parameters with grid search: training epoch 10, learning rate $\in [3e\text{-}6, 1e\text{-}4]$, training batch size $\in \{8, 12, 16, 24, 32\}$, dropout rate $\in \{0.4, 0.5, 0.6\}$. Our experiments run on one Quadro RTX 8000. For trigger detection, the average runtime is 3.0 hours. For argument detection, the average runtime is 1.3 hours. We use Spacy to generate POS tags.

**Evaluation Criteria** For evaluation of supervised event extraction, we use the same criteria as (Li et al., 2013; Chen et al., 2015; Nguyen et al., 2016; Lin et al., 2020) as follows:

- **Trigger**: A trigger mention is correct if its span and event type matches a reference trigger. Each candidate may act as triggers for multiple event occurrences.

- **Argument**: An argument prediction is correct only if the event trigger is correctly detected. Meanwhile, its span and argument role need to match a reference argument. An argument candidate can be involved in multiple events as different roles. Furthermore, within a single event extent, an argument candidate may play multiple roles.

## B Impact of Seed Triggers

To investigate the impact of seed triggers on event trigger extraction, we take the supervised event extraction ACE dataset as a case study, where we divide the triggers in the evaluation dataset into two groups: overlapped triggers with the seeds or non-overlapped ones with the seeds. Then, we compare the performance of our approach with and without using seed triggers as part of the event type representations. As Table 6 shows, by incorporating the seed triggers as part of the event type representations, our approach achieves better performance on both overlapped and non-overlapped triggers, demonstrating the benefit of seed triggers on representing event types. As the total number of overlapped triggers (34) is much lower than that of non-overlapped triggers (390), we view the impact of seed triggers on overlapped and non-overlapped triggers as comparable. On the other hand, by comparing our approach without using seed triggers with the BERT_QA_Arg baseline, our approach also achieves much better performance which is mostly due to the attention mechanism we used which can better capture the semantic consistency between the input tokens and the event type query which just consists of the event type name.

## C In-depth Comparison for Cross Ontology Transfer

To deeply investigate the reason that our approach performs better than QA-based baselines from cross ontology transfer, we conducted ablation study by removing the seed triggers from the event type queries of our approach, as shown in Table 7. The BERT_QA_Arg$_{\text{multi}}$ utilizes a generic query, e.g., *what's the trigger*, and classify each input token into one of the target types. It's essentially a multiclass classifier but just taking a query as the prompt. The BERT_QA_Arg$_{\text{binary}}$ utilizes each event type as the query to extract the corresponding event mentions. Comparing the two baseline methods, BERT_QA_Arg$_{\text{binary}}$ works slightly better than BERT_QA_Arg$_{\text{multi}}$, especially on ACE, demonstrating the benefit of type-oriented binary decoding mechanism. The only difference between BERT_QA_Arg$_{\text{binary}}$ and our approach without seed triggers is the learning of enriched contextual representations. The comparison of their scores demonstrates the effectiveness of the attention mechanisms designed for trigger extraction. Finally, by incorporating the seed triggers into event type representations, our approach is further improved significantly for all the settings. These in-depth comparisons demonstrate the effectiveness of both seed triggers and the attention mechanisms in our approach for transferring annotations from old types to the new types.

| | Overlapped Triggers | Non-overlapped Triggers |
|---|---|---|
| OneIE (Lin et al., 2020) | 88.2 | 71.0 |
| BERT_QA_Arg (Du and Cardie, 2020) | 72.2 | 70.9 |
| **Our Approach w/o Seed Triggers** | 88.9 | 70.8 |
| **Out Approach w/ Seed Triggers** | 97.2 | 71.3 |

Table 6: Impact of seed triggers on supervised trigger extraction on ACE (F-score, %)

| Source | Target | BERT_QA_Arg$_{\text{multi}}$ † | BERT_QA_Arg$_{\text{binary}}$ † | **Our Approach** | |
|---|---|---|---|---|---|
| | | | | w/o Seed Triggers | w/ Seed Triggers |
| ERE | ACE | 48.9 | 50.8 | 53.8 | 53.9 |
| ACE | ACE | 70.6 | 72.2 | 72.2 | 73.6 |
| ACE+ERE | ACE | 70.1 | 71.3 | 72.2 | 74.4 |
| ACE | ERE | 47.2 | 47.2 | 48.7 | 55.9 |
| ERE | ERE | 57.0 | 56.7 | 58.2 | 60.4 |
| ACE+ERE | ERE | 57.0 | 54.6 | 56.2 | 63.0 |

Table 7: Cross ontology transfer results for queries without seed triggers, between ACE and ERE datasets (F-score %)

## D More Ablation Studies of Supervised Event Extraction

The entity recognition model is based on a pre-trained BERT (Devlin et al., 2019) encoder with a CRF (Lafferty et al., 2001; Passos et al., 2014) based prediction network. It's trained on the same training dataset from ACE05 before event extraction, and the predictions are taken as input to argument extraction to indicate the candidate argument spans. Table 8 shows the comparison of the entity extraction performance between our BERT-CRF approach and the baselines.

| **Model** | F1 |
|---|---|
| OneIE | 89.6 |
| FourIE | 91.1 |
| BERT+CRF | 89.3 |

Table 8: Performance of Entity Extraction (F-score, %)

To understand the factors that affect argument extraction and decompose the errors propagated along the learning process (from predicted triggers or predicted entities), we conduct experiments that condition on given ground truth labels for those factors. Specifically, we investigate three settings: 1) given gold entity, 2) given gold event trigger, and 3) given both gold entity and event trigger. The experimental results is shown in Table 9.

| Given Information | ACE | ERE |
|---|---|---|
| None | 55.1 | 50.2 |
| GE | 59.7 (+4.6) | 59.5 (+9.3) |
| GT | 68.7 (+13.6) | 67.2 (+17.0) |
| GT & GE | 74.2 (+19.1) | 72.2 (+22.0) |

Table 9: Performance of argument extraction conditioning on various input information: gold trigger (GT), and gold entities (GE). (F-score, %)

## E Remaining Challenges for Supervised Event Extraction

We sample 200 supervised trigger detection and argument extraction errors from the ACE test dataset and identify the remaining challenges.

**Lack of Background Knowledge** Background knowledge, as well as human commonsense knowledge, sometimes is essential to event extraction. For example, from the sentence "*since the intifada exploded in September 2000, the source said*", without knowing that *intifada* refers to a resistance movement, our approach failed to detect it as an *Attack* event mention.

**Pronoun Resolution** Extracting arguments by resolving coreference between entities and pronouns is still challenging. For example, in the following sentence "*Attempts by Laleh and Ladan to have their operation elsewhere in the world were rejected, with doctors in Germany saying one or both of them could die*", without pronoun resolution, our approach mistakenly extracted *one*, *both* and *them* as *Victims* of the *Die* event triggered by *die*, while the actual *Victims* are *Ladan* and *Laleh*.

**Ambiguous Context** The ACE annotation guidelines (Consortium, 2005) provide detailed rules and constraints for annotating events of all event types. For example, a *Meet* event must be specified by the context as *face-to-face and physically located somewhere*. Though we carefully designed several attention mechanisms, it is difficult for the machines to capture such context features accurately. For example, from the sentence "*The admission came during three-day talks in Beijing which concluded Friday, the first meeting between US and North Korean officials since the nuclear crisis erupted six months ago.*", our approach failed to capture the context features that *the talks is not an explicit face-to-face meet event*, and thus mistakenly identified it as a *Meet* event mention.