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

Anonymous ACL submission

Abstract

Event extraction is typically modeled as a multi-class classification problem where both 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 takes both 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 two public benchmark datasets, ACE and ERE, demonstrate that our approach achieves the state-of-the-art performance on each dataset and significantly outperforms existing methods on zero-shot event extraction. We will make all the programs publicly available once the paper is accepted.

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.

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 firstly defined and then supervised machine learning approaches will detect and classify each candidate event mention or argument into one of the target types. However, in these approaches, each event type or argument role is treated as an atomic symbol, ignoring their rich semantics. 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, thus limited to a single target event ontology, such as ACE 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.1 To this end, we propose a new event extraction framework following a query-and-extract paradigm. Our framework represents both 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 shortlist of prototype triggers, and make binary decoding of each token based on its query-aware embedding; (2) for argument extraction, we put together all argument roles defined under each event type as a query, followed by a multiway attention

\[\text{Figure 1: An example of event annotation.}\]
Our proposed 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., 2020) 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 is more accurate and efficient.

We evaluate our approach on two public benchmark datasets, ACE and ERE. We demonstrate state-of-the-art performance in both the standard supervised event extraction and the challenging transfer learning settings that generalize to new event types and new ontologies. Specifically, equipped with the cross-ontology transferability, our approach can make use of both datasets and achieve 1.1% and 3.6% F-score gain on trigger detection compared with the previous state of the arts on ACE and ERE, respectively. 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, leading to both improved 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 short list 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 sequen-
tial 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 as well as 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 as well as 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 \( f_0 \) and its frequency of being tagged as an event trigger of type \( t \) as \( f_t \), and then obtain a probability \( f_t / f_0 \), which will be used to sort all the annotated triggers for event type \( t \). We select the top-\( K \) \(^2\) 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 \( T = \{ t, \tau_1, \tau_2, \ldots, \tau_K \} \). Taking the event type \( \text{Attack} \) as an example, we finally represent it as \( \text{Attack invaded airstrikes overthrew ambushed} \).

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

\[
[\text{CLS}][\text{EVENT}][\text{SEP}] w_1 \ldots w_N [\text{SEP}] t \tau_1^t \ldots \tau_K^t [\text{SEP}]
\]

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 emphasize 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 \( W = \{ w_0, w_1, \ldots, w_N \} \) as well as the event type \( T = \{ t, \tau_1^t, \tau_2^t, \ldots, \tau_K^t \} \).

**Enriched Contextual Representation** Given a query of each event type, we aim to extract corresponding event triggers from the input sentence automatically. 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:

\[
A^T_i = \sum_{j=1}^{\left| T \right|} \alpha_{ij} \cdot T_j, \quad \text{where} \quad \alpha_{ij} = \cos(w_i, T_j),
\]

where \( T_j \) is the contextual representation of the \( j \)-th token in the sequence \( T = \{ t, \tau_1^t, \tau_2^t, \ldots, \tau_K^t \} \). \( \cos(\cdot, \cdot) \) is the cosine similarity function between two vectors. \( A^T_i \) 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 certain context. For example, according to ACE event annotation guidelines (Linguistic Data 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:

\[
A^W_i = \sum_{j=1}^{N} \bar{\alpha}_{ij} \cdot w_j, \quad \text{where} \quad \bar{\alpha}_{ij} = \rho(w_i, w_j),
\]

where \( \rho(\cdot, \cdot) \) is the attention function and is computed as the average of the self-attention weights from the last \( m \) layers of BERT.\(^4\)

\(^2\)In our experiments, we set \( K = 4 \).

\(^3\)We use bold symbols to denote vectors.

\(^4\)We set \( m \) as 3 as it achieved the best performance.
Event Trigger Detection With the aforementioned event type oriented attention and in-context attention mechanisms, each token $w_i$ from the input sentence will obtain two enriched contextual representations $A_i^W$ and $A_i^T$. We concatenate them with the original contextual representation $w_i$ from the BERT encoder, and classify it into a binary label, indicating it as a candidate trigger of event type $t$ or not:

$$\hat{y}_i^t = U_\theta \cdot ([w_i; A_i^W; A_i^T; P_i]),$$

where $[;]$ denotes concatenation operation, $U_\theta$ is a learnable parameter matrix for event trigger detection, and $P_i$ is the one-hot part-of-speech (POS) encoding of word $w_i$. We optimize the following objective for event trigger detection

$$L_1 = - \frac{1}{|T||N|} \sum_{t \in T} \sum_{i \in N} y_i^t \cdot \log \hat{y}_i^t,$$

where $T$ is the set of target event types and $N$ is the set of tokens from the training dataset. $y_i^t$ 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 pre-defined argument roles for event type $t$ as $G_t = \{g_1^t, g_2^t, \ldots, g_D^t\}$. 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

[CLS] $w_1 \ w_2 \ \ldots \ \ w_N$ [SEP] $g_1^t \ g_2^t \ \ldots \ g_D^t$ [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 $\tilde{W} = \{\tilde{w}_0, \tilde{w}_2, \ldots, \tilde{w}_N\}$, and the argument roles $\tilde{G}_t = \{\tilde{g}_0^t, \tilde{g}_1^t, \ldots, \tilde{g}_D^t, \tilde{g}_{Other}^t\}$.

As the candidate trigger $r$ may span multiple tokens within the sentence, we obtain its contextual representation $r$ 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 $E = \{e_1, e_2, \ldots, e_M\}$. 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 $\tilde{E} = \{e_1, e_2, \ldots, e_M\}$.

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

$$h_i = U_\theta \cdot ([e_i; r; e_i \circ r]),$$

where $\circ$ denotes element-wise multiplication operation, $U_\theta$ 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 $S$ between the trigger-aware entity representations $\{h_1, h_2, \ldots, h_M\}$ and the argument role representations $\{g_0^t, g_1^t, \ldots, g_D^t\}$

$$S_{ij} = \frac{1}{\sqrt{d}} \sigma(h_i, g_j^t),$$

where $\sigma$ denotes dot product operator, $d$ denotes embedding dimension of $g_i^t$, 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 $S$, 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:

$$A_i^{2g} = \sum_{j=1}^{D} \sum_{j=1}^{M} S_{ij} \cdot g_j^t, \quad A_i^{2e} = \sum_{i=1}^{M} S_{ij} \cdot h_i,$$

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
\[
\mu_{ij} = \rho(h_i, h_j), \quad \tilde{\mu}_i = \text{Softmax} (\mu_i),
\]
\[
A_i^{\text{2e}} = \sum_{j=1}^{M} \tilde{\mu}_{ij} \cdot h_j,
\]
where \(\rho\) denotes the averaged self-attention weights obtained from the last \(m\) layers of BERT encoder.

Similarly, to capture the underlying relations among argument roles, we also compute the self-attention among them
\[
v_{jk} = \frac{1}{\sqrt{d}} \sigma(g_j^t, g_k^t), \quad \tilde{v}_j = \text{Softmax} (v_j),
\]
\[
A_j^{\text{2g}} = \sum_{k=1}^{D} \tilde{v}_{jk} \cdot g_k^t,
\]
where \(\sigma\) denotes the dot product operator, and \(d\) denotes embedding dimension of \(g^t\).

**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:
\[
\tilde{z}_{ij}^t = U_a \cdot ([h_i; g_j^t]; A_i^{\text{2e}}; A_j^{\text{2g}}),
\]
where \(U_a\) is a learnable parameter matrix for argument extraction. The training objective is to minimize the following loss function:
\[
\mathcal{L}_2 = -\frac{1}{|A| |E|} \sum_{j=1}^{|A|} \sum_{i=1}^{|E|} \tilde{z}_{ij} \log \tilde{z}_{ij},
\]
where \(A\) denotes the collection of possible argument roles, and \(E\) is the set of entities we need to consider for argument extraction. \(z_{ij}\) 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, Automatic Content Extraction 2005 (ACE05-E+)\(^5\) and Entity Relation Event (ERE-EN) (Song et al., 2015)\(^6\). ACE defines 33 event types while ERE includes 38 types, among which there are 31 overlapping event types. Following previous studies (Wadden et al., 2019; Du and Cardie, 2020; Lin et al., 2020), we only consider the arguments from the 7 entity types, including Facility, Geo-Political Entity, Location, Organization, Person, Vehicle, Weapon, and ignore Time and Value related arguments. We use the same data split of ACE and ERE as (Li et al., 2013; Wadden et al., 2019; Lin et al., 2020; Du and Cardie, 2020; Lin et al., 2020; Nguyen et al., 2021) 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). 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 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 many other event extraction studies (Li et al., 2013; Yang and Mitchell, 2016; Liu et al., 2020, 2018; Sha et al., 2018; Lai et al., 2020; Veyseh et al., 2020; Nguyen et al., 2021) 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). More details regarding the data statistics and evaluation are shown in Appendix A.

---

\(^5\) https://catalog.ldc.upenn.edu/LDC2006T06

\(^6\) Following Lin et al. (2020), we merge LDC2015E29, LDC2015E68, and LDC2015E78 as the ERE dataset.
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.

<table>
<thead>
<tr>
<th>Model</th>
<th>ACE05-E+</th>
<th>ERE-EN</th>
</tr>
</thead>
<tbody>
<tr>
<td>DYGIE++ (Wadden et al., 2019)</td>
<td>67.3*</td>
<td>42.7*</td>
</tr>
<tr>
<td>BERT_QA_Arg (Du and Cardie, 2020)</td>
<td>70.6*</td>
<td>48.3*</td>
</tr>
<tr>
<td>OneIE (Lin et al., 2020)</td>
<td>72.8</td>
<td>54.8</td>
</tr>
<tr>
<td>Text2Event (Lu et al., 2021)</td>
<td>71.8</td>
<td>54.4</td>
</tr>
<tr>
<td>FourIE (Nguyen et al., 2021)</td>
<td>73.3</td>
<td>57.5</td>
</tr>
<tr>
<td>Our Approach</td>
<td>73.7</td>
<td>55.1</td>
</tr>
</tbody>
</table>

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.

<table>
<thead>
<tr>
<th>Model</th>
<th>Trigger Ext.</th>
<th>Arg Ext. (GT)</th>
</tr>
</thead>
<tbody>
<tr>
<td>BERT_QA_Arg†</td>
<td>31.6</td>
<td>17.0</td>
</tr>
<tr>
<td>Our Approach</td>
<td>47.8</td>
<td>43.0</td>
</tr>
</tbody>
</table>

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 any prototype triggers for zero-shot event extraction.

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
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.

performance on each of them. As shown in Figure 3, our approach performs exceptionally well on 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_Argmulti, which is the same as the original implementation and use multi-classification to detect event triggers. (2) BERT_QA_Argbinary, 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.9

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.

3.5 Ablation Study

We further evaluate the impact of each attention mechanism to event trigger detection and argument extraction. As Table 4 shows, all the attention

---

9Another intuitive training strategy is to sequentially train the model on the source and target ontologies. Our pilot study shows that this strategy performs slightly worse.
mechanisms show significant benefit to trigger or argument extraction, especially on ERE dataset. Among them, the Event Type Attention and Multitask 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, our approach still shows competitive performance for argument extraction comparing with the strong baselines. More ablation studies are discussed in Appendix C.

<table>
<thead>
<tr>
<th>Model</th>
<th>ACE</th>
<th>ERE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Trigger</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Our Approach</td>
<td>73.7</td>
<td>60.4</td>
</tr>
<tr>
<td>w/o In-Context Attention</td>
<td>71.9</td>
<td>58.2</td>
</tr>
<tr>
<td>w/o Event Type Attention</td>
<td>70.7</td>
<td>56.9</td>
</tr>
<tr>
<td>Arg.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Our Approach</td>
<td>55.1</td>
<td>50.2</td>
</tr>
<tr>
<td>w/o Entity Detection</td>
<td>53.0</td>
<td>47.9</td>
</tr>
<tr>
<td>w/o Multiway Attention</td>
<td>54.0</td>
<td>42.2</td>
</tr>
<tr>
<td>w/o Entity Self-attention</td>
<td>53.6</td>
<td>48.3</td>
</tr>
<tr>
<td>w/o Arg Role Self-attention</td>
<td>54.1</td>
<td>47.7</td>
</tr>
</tbody>
</table>

Table 4: Results of various ablation studies.

3.6 Computational and Time Cost

Despite the performance improvement via extending from multi-class classification to multiple binary classifications, these approaches usually increase the time cost. We thus 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 comparing with BERT_QA_Arg (Du and Cardie, 2020) which relies on multi-class classification for trigger extraction, while for argument extraction, our approach takes 36% less time for training and inference than BERT_QA_Arg.

4 Related Work

Traditional event extraction studies (Ji and Grishman, 2008; Liao and Grishman, 2010; 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; 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) or event type structures (Huang et al., 2018). However, they can hardly be generalized to argument extraction. Recent studies that model event extraction as question answering (Du and Cardie, 2020; Liu et al., 2020; Li et al., 2020; 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 two different vital designs, making our approach perform and be generalized better than these 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. (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.

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 to further improve the the accuracy and generalizability of event extraction.

We take consecutive tokens predicted with the same argument role as a single argument span.
References


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.


A Data Statistics and Implementation Details

Table 5 shows the detailed data statistics 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.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Split</th>
<th># Events</th>
<th># Arguments</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACE05-E+</td>
<td>Train</td>
<td>4419</td>
<td>7932</td>
</tr>
<tr>
<td></td>
<td>Dev</td>
<td>468</td>
<td>892</td>
</tr>
<tr>
<td></td>
<td>Test</td>
<td>424</td>
<td>898</td>
</tr>
<tr>
<td>ERE-EN</td>
<td>Train</td>
<td>7232</td>
<td>12832</td>
</tr>
<tr>
<td></td>
<td>Dev</td>
<td>619</td>
<td>1100</td>
</tr>
<tr>
<td></td>
<td>Test</td>
<td>652</td>
<td>1228</td>
</tr>
</tbody>
</table>

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 $\in [3e-6, 1e-4]$, 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.

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 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 (Linguistic Data 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.

C More Ablation Studies of Supervised Event Extraction

The entity recognition model is based on a pretrained 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 6 shows the comparison of the entity extraction performance between our BERT-CRF approach and the baselines.

<table>
<thead>
<tr>
<th>Model</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>OneIE</td>
<td>89.6</td>
</tr>
<tr>
<td>FourIE</td>
<td>91.1</td>
</tr>
<tr>
<td>BERT+CRF</td>
<td>89.3</td>
</tr>
</tbody>
</table>

Table 6: 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 7.

<table>
<thead>
<tr>
<th>Given Information</th>
<th>ACE</th>
<th>ERE</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td>55.1</td>
<td>50.2</td>
</tr>
<tr>
<td>GE</td>
<td>59.7 (+4.6)</td>
<td>59.5 (+9.3)</td>
</tr>
<tr>
<td>GT</td>
<td>68.7 (+13.6)</td>
<td>67.2 (+17.0)</td>
</tr>
<tr>
<td>GT &amp; GE</td>
<td>74.2 (+19.1)</td>
<td>72.2 (+22.0)</td>
</tr>
</tbody>
</table>

Table 7: Performance of argument extraction conditioning on various input information: gold trigger (GT), and gold entities (GE). (F-score, %)