Zero-Shot Transfer Learning for Event Extraction

Lifu Huang[1], **Heng Ji**[1], **Kyunghyun Cho**[2],
Ido Dagan[3], **Sebastian Riedel**[4], **Clare R. Voss**[5]
[1] Rensselaer Polytechnic Institute, [2] New York University,
[3] Bar-Ilan University, [4] University College London,
[5] US Army Research Lab
[1] {huangl7, jih}@rpi.edu, [2] kyunghyun.cho@nyu.edu
[3] dagan@cs.biu.ac.il, [4] s.riedel@ucl.ac.uk
[5] clare.r.voss.civ@mail.mil

Abstract

Most previous supervised event extraction methods have relied on features derived from manual annotations, and thus cannot be applied to new event types without extra annotation effort. We take a fresh look at event extraction and model it as a generic grounding problem: mapping each event mention to a specific type in a target event ontology. We design a transferable architecture of structural and compositional neural networks to jointly represent and map event mentions and types into a shared semantic space. Based on this new framework, we can select, for each event mention, the event type which is semantically closest in this space as its type. By leveraging manual annotations available for a small set of existing event types, our framework can be applied to new unseen event types without additional manual annotations. When tested on 23 unseen event types, this zero-shot framework, without manual annotations, achieves performance comparable to a supervised model trained from 3,000 sentences annotated with 500 event mentions.[1]

1 Introduction

The goal of event extraction is to identify event triggers and their arguments in unstructured text data, and then to assign an event type to each trigger and a semantic role to each argument. An example is shown in Figure 1. Traditional supervised methods have typically modeled this task of event extraction as a classification problem, by assigning event triggers to event types from a pre-defined fixed set. These methods rely heavily on manual annotations and features specific to each event type, and thus are not easily adapted to new event types without extra annotation effort. Handling new event types may even entail starting over, without being able to re-use annotations from previous event types.

To make event extraction effective as new real-world scenarios emerge, we take a look at this task from the perspective of zero-shot learning, ZSL (Frome et al., 2013; Norouzi et al., 2013; Socher et al., 2013a). ZSL, as a type of transfer learning, makes use of separate, pre-existing classifiers to build a semantic, cross-concept space that maps between their respective classes. The resulting shared semantic space then allows for building a novel "zero-shot" classifier, i,e,, requiring no (zero) additional training examples, to handle *unseen* cases. We observe that each event mention has a structure consisting of a candidate trigger and arguments, with corresponding predefined name labels for the event type and argument roles. We propose to enrich the semantic representations of each event mention and event type with rich structures, and determine the type based on the semantic similarity between an event mention and each event type defined in a target ontology. Let's consider two example sentences:

E1. The Government of *China* has ruled Tibet since 1951 after **dispatching** *troops* to the *Himalayan* region in *1950*.

E2. Iranian state television stated that the **conflict** between the *Iranian police* and the drug *smugglers* took place near the town of *mirjaveh*.

In E1, as also diagrammed in Figure 1, **dis-**

[1]The programs are publicly available for research purpose at: https://github.com/wilburOne/ZeroShotEvent

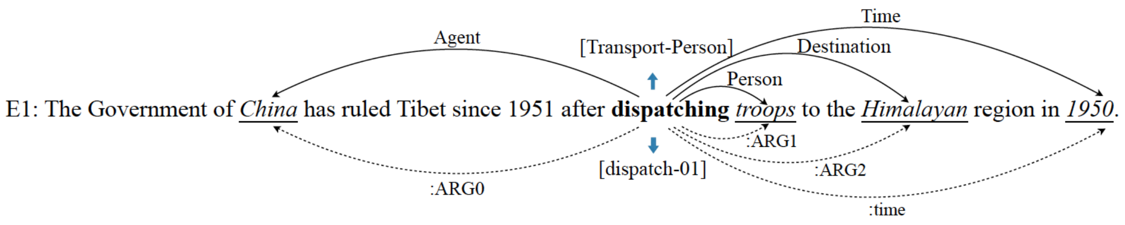
Agent
[Transport-Person]
Time
Destination
Person
E1: The Government of China has ruled Tibet since 1951 after dispatching troops to the Himalayan region in 1950.
:ARG1
:ARG2
[dispatch-01]
:ARG0
:time

Figure 1: Event Mention Example: **dispatching** is the trigger of a *Transport-Person* event with four arguments: the solid lines show the event annotations for the sentence while the dotted lines show the Abstract Meaning Representation parsing output.

patching is the trigger for the event mention of type *Transport_Person* and in E2, **conflict** is the trigger for the event mention of type *Attack*. We make use of Abstract Meaning Representations (AMR) (Banarescu et al., 2013) to identify the candidate arguments and construct event mention structures as shown in Figure 2 (top). Figure 2 (bottom) also shows event type structures defined in the Automatic Content Extraction (ACE) guideline.[2] We can see that a trigger and its event type name usually have some shared meaning. Furthermore, their structures also tend to be similar: a *Transport_Person* event typically involves a *Person* as its *patient* role, while an *Attack* event involves a *Person* or *Location* as an *Attacker*. This observation matches the theory by Pustejovsky (1991): "the semantics of an event structure can be generalized and mapped to event mention structures in a systematic and predictable way".

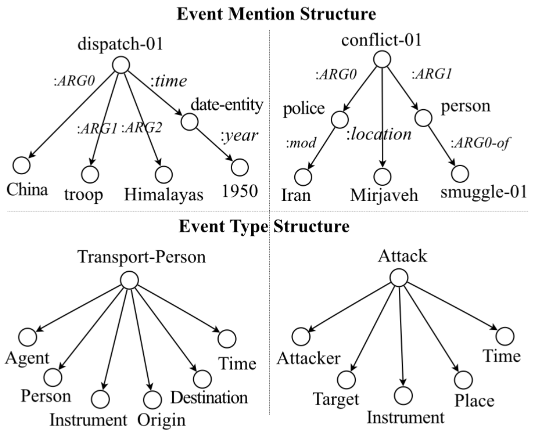

Figure 2: Examples of Event Mention Structures and Type Structures from ACE.

Inspired by this theory, for the first time, we model event extraction as a generic grounding problem, by mapping each mention to its semantically closest event type. Given an event ontology, where each event type structure is well-defined, we will refer to the event types for which we have annotated event mentions as *seen* types, while those without annotations as *unseen* types. Our goal is to learn a generic mapping function independent of event types, which can be trained from annotations for a limited number of seen event types and then used for any new unseen event types. We design a transferable neural architecture, which jointly learns and maps the structural representations of event mentions and types into a shared semantic space, by minimizing the distance between each event mention and its corresponding type. For event mentions with *unseen* types, their structures will be projected into the same semantic space using the same framework and assigned types with top-ranked similarity values.

To summarize, to apply our new zero-shot transfer learning framework to any new unseen event types, we only need (1) a structured definition of the unseen event type (its type name along with role names for its arguments, from the event ontology); and (2) some annotations for one or a few seen event types. Without requiring any additional manual annotations for the new unseen types, our ZSL framework achieves performance comparable to supervised methods trained from a substantial amount of training data for the same types.

2 Approach Overview

Briefly here, we overview the phases involved in building our framework's shared semantic space that, in turn, is the basis for the ZSL framework. Given a sentence s, we start by identifying candidate triggers and arguments based on AMR parsing (Wang et al., 2015b). For the example shown in Figure 1, we identify *dispatching* as a trigger, and its candidate arguments: *China*, *troops*, *Himalayan* and *1950*. The details will be described in Section 3.

[2]https://en.wikipedia.org/wiki/Automatic_content_extraction

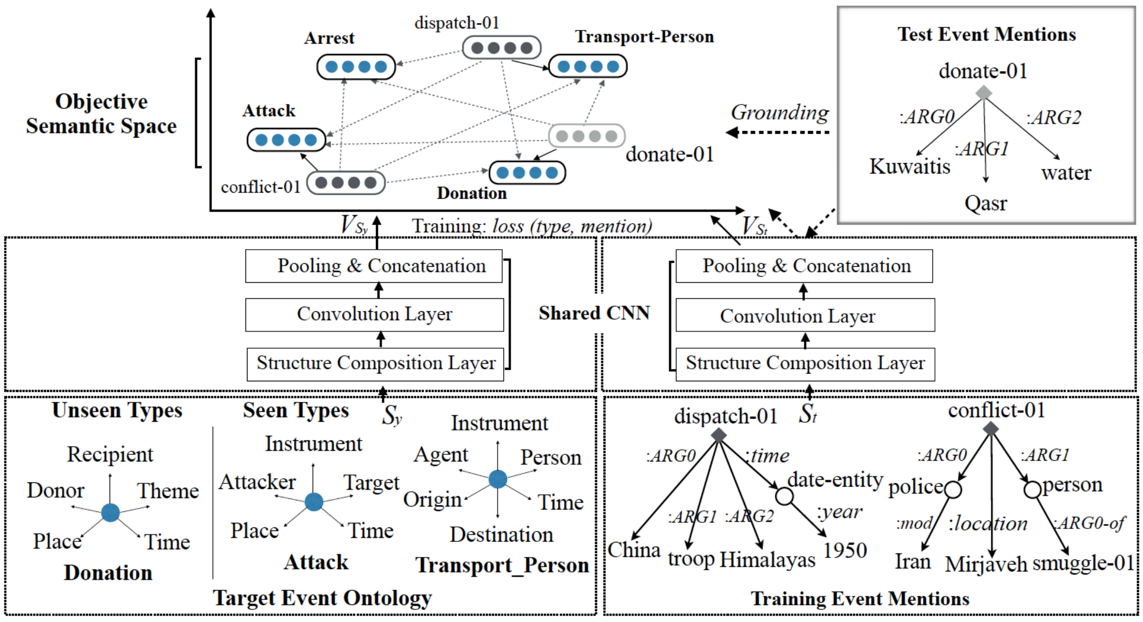

Figure 3: Architecture Overview. The blue circles denote event types and event type representations. The dark grey diamonds and circles denote triggers and trigger representations from training set. The light grey diamonds and circles denote triggers and trigger representations from testing set.

After this identification phase, we use our new neural architecture, as depicted in Figure 3, to classify triggers into event types. (The classification of arguments into roles follows the same pipeline.) For each trigger t, e.g., *dispatch-01*, we determine its type by comparing its semantic representation with that of any event type in the event ontology. In order to incorporate the contexts into the semantic representation of t, we build a structure S_t using AMR as shown in Figure 3. Each structure is composed of a set of tuples, e.g, ⟨*dispatch-01*, *:ARG0*, *China*⟩. We use a matrix to represent each AMR relation, composing its semantics with two concepts for each tuple (in Section 4), and feed all tuple representations into a CNN to generate a dense vector representation V_{S_t} for the event mention structure (in Section 5.1).

Given a target event ontology, for each type y, e.g., *Transport_Person*, we construct a type structure S_y consisting of its predefined roles, and use a tensor to denote the implicit relation between any type and argument role. We compose the semantics of type and argument role with the tensor for each tuple, e.g., ⟨*Transport_Person*, *Destination*⟩ (in Section 4). Then we generate the event type structure representation V_{S_y} using the same CNN (in Section 5.1). By minimizing the semantic distance between *dispatch-01* and *Transport_Person* using their dense vectors, V_{S_t} and V_{S_y} respectively, we jointly map the representations of event mention and event types into a shared semantic space, where each mention is closest to its annotated type.

After training that completes the construction of the semantic space, the compositional functions and CNNs are then used to project any new event mention (e.g., *donate-01*) into the semantic space and find its closest event type (e.g., *Donation*) (in Section 5.3). In the next sections we will elaborate each step in great detail.

3 Trigger and Argument Identification

Similar to Huang et al. (2016), we identify candidate triggers and arguments based on AMR Parsing (Wang et al., 2015b) and apply the same word sense disambiguation (WSD) tool (Zhong and Ng, 2010) to disambiguate word senses and link each sense to OntoNotes, as shown in Figure 1.

Given a sentence, we consider all noun and verb concepts that can be mapped to OntoNotes senses by WSD as candidate event triggers. In addition, the concepts that can be matched with verbs or nominal lexical units in FrameNet (Baker et al., 1998) are also considered as candidate triggers. For each candidate trigger, we consider any concepts that are involved in a subset of AMR relations as candidate arguments [3]. We manually select this subset of AMR relations that are useful for capturing generic relations between event triggers and arguments, as shown in Table 1.

Categories	Relations
Core roles	ARG0, ARG1, ARG2, ARG3, ARG4
Non-core roles	mod, location, instrument, poss, manner, topic, medium, prep-X
Temporal	year, duration, decade, weekday, time
Spatial	destination, path, location

Table 1: Event-Related AMR Relations.

4 Trigger and Type Structure Composition

As Figure 3 shows, for each candidate trigger t, we construct its event mention structure S_t based on its candidate arguments and AMR parsing. For each type y in the target event ontology, we construct a structure S_y by including its pre-defined roles and using its type as the root.

Each S_t or S_y is composed of a collection of tuples. For each event mention structure, a tuple consists of two AMR concepts and an AMR relation. For each event type structure, a tuple consists of a type name and an argument role name. Next we will describe how to compose semantic representations for event mention and event type respectively based on these structures.

Event Mention Structure For each tuple $u = \langle w_1, \lambda, w_2 \rangle$ in an event mention structure, we use a matrix to represent each AMR relation λ, and compose the semantics of λ between two concepts w_1 and w_2 as:

$$V_u = [V_{w_1}^{'}; V_{w_2}^{'}] = f([V_{w_1}; V_{w_2}] \cdot M_\lambda)$$

where V_{w_1}, $V_{w_2} \in \mathbb{R}^d$ are the vector representations of words w_1 and w_2. d is the dimension size of each word vector. $[\,;\,]$ denotes the concatenation of two vectors. $M_\lambda \in \mathbb{R}^{2d \times 2d}$ is the matrix representation for AMR relation λ. V_u is the composition representation of tuple u, which consists of two updated vector representations $V_{w_1}^{'}$, $V_{w_2}^{'}$ for w_1 and w_2 by incorporating the semantics of λ.

Event Type Structure For each tuple $u^{'} = \langle y, r \rangle$ in an event type structure, where y denotes the event type and r denotes an argument role, following Socher et al. (2013b), we assume an implicit relation exists between any pair of type and argument, and use a single and powerful tensor to represent the implicit relation:

$$V_{u^{'}} = [V_y^{'}; V_r^{'}] = f([V_y; V_r]^T \cdot U^{[1:2d]} \cdot [V_y; V_r])$$

where V_y and V_r are vector representations for y and r. $U^{[1:2d]} \in \mathbb{R}^{2d \times 2d \times 2d}$ is a 3-order tensor. $V_u^{'}$ is the composition representation of tuple $u^{'}$, which consists of two updated vector representations $V_y^{'}$, $V_r^{'}$ for y and r by incorporating the semantics of their implicit relation $U^{[1:2d]}$.

5 Trigger and Argument Classification

5.1 Trigger Classification for Seen Types

Both event mention and event type structures are relatively simple and can be represented with a set of tuples. CNNs have been demonstrated effective at capturing sentence level information by aggregating compositional n-gram representations. In order to generate structure-level representations, we use CNN to learn to aggregate all edge and tuple representations.

Input layer is a sequence of tuples, where the order of tuples is from top to bottom in the structure. Each tuple is represented by a $d \times 2$ dimensional vector, thus each mention structure and each type structure are represented as a feature map of dimensionality $d \times 2h^*$ and $d \times 2p^*$ respectively, where h^* and p^* are the maximal number of tuples for event mention and type structures. We use zero-padding to the right to make the volume of all input structures consistent.

Convolution layer Take S_t with h^* tuples: $u_1, u_2, ..., u_{h^*}$ as an example. The input matrix of S_t is a feature map of dimensionality $d \times 2h^*$. We make c_i as the concatenated embeddings of n continuous columns from the feature map, where n is the filter width and $0 < i < 2h^* + n$. A convolution operation involves a filter $W \in \mathbb{R}^{nd}$, which is applied to each sliding window c_i:

$$c_i^{'} = \tanh(W \cdot c_i + b)$$

where $c_i^{'}$ is the new feature representation, and $b \in \mathbb{R}^d$ is a biased vector. We set filter width as 2 and stride as 2 to make the convolution function operate on each tuple with two input columns.

[3]On the whole ACE2005 corpus, using the AMR parser (Wang et al., 2015b), the coverage for trigger identification is 89.4% and the coverage for argument candidate identification is 66.0%.

Max-Pooling: All tuple representations $c_i^{'}$ are used to generate the representation of the input sequence by max-pooling.

Learning: For each event mention t, we name the correct type as *positive* and all the other types in the target event ontology as *negative*. To train the composition functions and CNN, we first consider the following hinge ranking loss:

$$L_1(t, y) = \sum_{j \in Y,\, j \neq y} \max\{0, m - C_{t,y} + C_{t,j}\}$$

$$C_{t,y} = \cos([V_t; V_{S_t}], [V_y; V_{S_y}])$$

where y is the positive event type for t. Y is the type set of the target event ontology. $[V_t; V_{S_t}]$ denotes the concatenation of representations of t and S_t. j is a negative event type for t from Y. m is a margin. $C_{t,y}$ denotes the cosine similarity between t and y.

The hinge loss is commonly used in zero-shot visual object classification task. However, it tends to overfit the seen types in our experiments. While clever data augmentation can help alleviate overfitting, we design two strategies: (1) we add "negative" event mentions into the training process. Here a "negative" event mention means that the mention has no positive event type among all seen types, namely it belongs to *Other*. (2) we design a new loss function as follows:

$$L_1^d(t, y) =$$
$$\begin{cases} \max\limits_{j \in Y, j \neq y} \max\{0, m - C_{t,y} + C_{t,j}\}, & y \neq \mathit{Other} \\ \max\limits_{j \in Y^{'}, j \neq y^{'}} \max\{0, m - C_{t,y^{'}} + C_{t,j}\}, & y = \mathit{Other} \end{cases}$$

where Y is the type set of the event ontology. $Y^{'}$ is the seen type set. y is the annotated type. $y^{'}$ is the type which ranks the highest among all event types for event mention t, while t belongs to *Other*.

By minimizing L_1^d, we can learn the optimized model which can compose structure representations and map both event mention and types into a shared semantic space, where the positive type ranks the highest for each mention.

5.2 Argument Classification for Seen Types

For each mention, we map each candidate argument to a specific role based on the semantic similarity of the argument path. Take E1 as an example. *China* is matched to *Agent* based on the semantic similarity between *dispatch-01→ :ARG0→ China* and *Transport-Person→Agent*.

Given a trigger t and a candidate argument a, we first extract a path $S_a = (u_1, u_2, ..., u_p)$, which connects t and a and consists of p tuples. Each predefined role r is also represented as a structure by incorporating the event type, $S_r = \langle y, r \rangle$. We apply the same framework to take the sequence of tuples contained in S_a and S_r into a weight-sharing CNN to rank all possible roles for a.

$$L_2^d(a, r) =$$
$$\begin{cases} \max\limits_{j \in R_y, j \neq r} \max\{0, m - C_{a,r} + C_{a,j}\} & r \neq \mathit{Other} \\ \max\limits_{j \in R_{Y^{'}}, j \neq r^{'}} \max\{0, m - C_{a,r^{'}} + C_{a,j}\} & r|y = \mathit{Other} \end{cases}$$

where R_y and $R_{Y^{'}}$ are the set of argument roles which are predefined for trigger type y and all seen types $Y^{'}$. r is the annotated role and $r^{'}$ is the argument role which ranks the highest for a when a or y is annotated as *Other*.

In our experiments, we sample various size of "negative" training data for trigger and argument labeling respectively. In the following section, we describe how the negative training instances are generated.

5.3 Zero-Shot Classification for Unseen Types

During test, given a new event mention $t^{'}$, we compute its mention structure representation for $S_{t^{'}}$ and all event type structure representations for $S_Y = \{S_{y_1}, S_{y_2}, ..., S_{y_n}\}$ using the same parameters trained from seen types. Then we rank all event types based on their similarity scores with mention $t^{'}$. The top ranked prediction for $t^{'}$ from the event type set, denoted as $\widehat{y}(t^{'}, 1)$, is given by:

$$\widehat{y}(t^{'}, 1) = \arg\max_{y \in Y} \cos([V_{t^{'}}; V_{S_{t^{'}}}], [V_y; V_{S_y}])$$

Moreover, $\widehat{y}(t^{'}, k)$ denotes the k^{th} most probable event type predicted for $t^{'}$. We will investigate the event extraction performance based on the top-k predicted event types.

After determining the type $y^{'}$ for mention $t^{'}$, for each candidate argument, we adopt the same ranking function to find the most appropriate role from the role set defined for $y^{'}$.

6 Experiments

6.1 Hyper-Parameters

We used the English Wikipedia dump to learn trigger sense and argument embeddings based on the Continuous Skip-gram model (Mikolov et al., 2013). Table 2 shows the hyper-parameters we used to train models.

Parameter Name	Value
Word Sense Embedding Size	200
Initial Learning Rate	0.1
# of Filters in Convolution Layer	500
Maximal # of Tuples for Mention Structure	10
Maximal # of Tuples for Argument Path	5
Maximal # of Tuples for Event Type Structure	5
Maximal # of Tuples for Argument Role Path	1

Table 2: Hyper-parameters.

6.2 ACE Event Classification

Setting	N	Seen Types for Training/Dev
A	1	Attack
B	3	Attack, Transport, Die
C	5	Attack, Transport, Die, Meet, Arrest-Jail
D	10	Attack, Transport, Die, Meet, Sentence, Arrest-Jail, Transfer-Money, Elect, Transfer-Ownership, End-Position

Table 3: Seen Types in Each Experiment Setting.

We first used the ACE event schema [4] as our target event ontology and assumed the boundaries of triggers and arguments as given. Of the 33 ACE event types, we selected the top-N most popular event types from ACE05 data as "seen" types, and used 90% event annotations of these for training and 10% for development. We set N as 1, 3, 5, 10 respectively. We tested the zero-shot classification performance on the annotations for the remaining 23 unseen types. Table 3 shows the types that we selected for training in each experiment setting.

The negative event mentions and arguments that belong to *Other* were sampled from the output of the system developed by Huang et al. (2016) based on ACE05 training sentences, which groups all candidate triggers and arguments into clusters based on semantic representations and assigns a type/role name to each cluster. We sampled the negative event mentions from the clusters (e.g., *Build*, *Threaten*) which do not map to ACE event types. We sampled the negative arguments from the arguments of negative event mentions. Table 4 shows the statistics of the training, development and testing data sets.

To show the effectiveness of structural similarity in our approach, we designed a baseline, WSD-Embedding, which directly maps event mentions and arguments to their candidate types and roles using our pre-trained word sense embeddings. Table 5 makes the contrast clear: structural similarity (our approach) is much more effective than lexical similarity (baseline) for both trigger and argument classification. Also, as the number of seen types in training increases, the performance of the transfer model improves.

We further evaluated the performance of our transfer approach on similar and distinct unseen types. The 33 subtypes defined in ACE fall within 8 coarse-grained main types, such as *Life* and *Justice*. Each subtype belongs to one main type. Subtypes that belong to the same main type tend to have similar structures. For example, *Trial-Hearing* and *Charge-Indict* have the same set of argument roles. For training our transfer model, we selected 4 subtypes of *Justice*: **Arrest-Jail**, **Convict**, **Charge-Indict**, **Execute**. For testing, we selected 3 other subtypes of *Justice*: *Sentence, Appeal, Release-Parole*. Additionally, we selected one subtype from each of the other seven main types for comparison. Table 6 shows that, when testing on a new unseen type, the more similar it is to the seen types, the better performance is achieved.

6.3 ACE Event Identification & Classification

The ACE2005 corpus includes the richest event annotations currently available for 33 types. However, in real-world scenarios, there may be thousands of event types of interest. To enrich the target event ontology and assess our transferable neural architecture on a large number of unseen types, when trained on limited annotations of seen types, we manually constructed a new event ontology which combined 33 ACE event types and argument roles, and 1,161 frames from FrameNet, except for the most generic frames such as *Entity* and *Locale*. Some ACE event types were easily aligned to frames, e.g., *Die* aligned to *Death*. Some frames were instead more accurately treated as inheritors of ACE types, such as *Suicide-Attack*, which inherits from *Attack*. We manually mapped the selected frames to ACE types.

We then compared our approach with the following state-of-the-art *supervised* methods:

- LSTM: A long short-term memory neural network (Hochreiter and Schmidhuber, 1997) based on distributed semantic features, similar

[4]ACE event schema specification is at: https://www.ldc.upenn.edu/sites/www.ldc.upenn.edu/files/english-events-guidelines-v5.4.3.pdf

Setting	Training			Development		Test		
	# of Types, Roles	# of Events	# of Arguments	# of Events	# of Arguments	# of Types/Roles	# of Events	# of Arguments
A	1, 5	953/900	894/1,097	105/105	86/130	23/59	753	879
B	3, 14	1,803/1,500	2,035/1,791	200/200	191/237			
C	5, 18	2,033/1,300	2,281/1,503	225/225	233/241			
D	10, 37	2537/700	2,816/879	281/281	322/365			

Table 4: Statistics for Positive/Negative Instances in Training, Dev, and Test Sets for Each Experiment.

Setting	Method	Hit@k Trigger Classification (%)			Hit@k Argument Classification (%)		
		k=1	k=3	k=5	k=1	k=3	k=5
	WSD-Embedding	1.7	13.0	22.8	2.4	2.8	2.8
A	Our Approach	4.0	23.8	32.5	1.3	3.4	3.6
B		7.0	12.5	36.8	3.5	6.0	6.3
C		20.1	34.7	46.5	9.6	14.7	15.7
D		33.5	51.4	68.3	14.7	26.5	27.7

Table 5: Comparison between Structural Representation (Our Approach) and Word Sense Embedding based Approaches on Hit@K Accuracy (%) for Trigger and Argument Classification.

Type	Subtype	Hit@k Trigger Classification		
		1	3	5
Justice	Sentence	68.3	68.3	69.5
Justice	Appeal	67.5	97.5	97.5
Justice	Release-Parole	73.9	73.9	73.9
Conflict	Attack	26.5	44.5	46.7
Transaction	Transfer-Money	48.4	68.9	79.5
Business	Start-Org	0	33.3	66.7
Movement	Transport	2.6	3.7	7.8
Personnel	End-Position	9.1	50.4	53.7
Contact	Phone-Write	60.8	88.2	90.2
Life	Injure	87.6	91.0	91.0

Table 6: Performance on Various Types Using Justice Subtypes for Training

to (Feng et al., 2016).

- Joint: A structured perceptron model based on symbolic semantic features (Li et al., 2013).

For our approach, we followed the experiment setting *D* in the previous section, using the same training and development data sets for the 10 seen types, but targeted all 1,194 event types in our new event ontology, instead of just the 33 ACE event types. For evaluation, we sampled 150 sentences from the remaining ACE05 data, including 129 annotated event mentions for the 23 unseen types. For both LSTM and Joint approaches, we used the entire ACE05 annotated data for 33 ACE event types for training except for the held-out 150 evaluation sentences.

We first identified the candidate triggers and arguments, then mapped each of these to the target event ontology. We evaluated our model on their extracting of event mentions which were classified into 23 testing ACE types. Table 7 shows the performance.

To further demonstrate the effectiveness of zero-shot learning in our framework and its impact in saving human annotation effort, we used the supervised LSTM approach for comparison. The training data of LSTM contained 3,464 sentences with 905 annotated event mentions for the 23 unseen event types. We divided these event annotations into 10 subsets and successively added one subset at a time (10% of annotations) into the training data of LSTM. Figure 4 shows the LSTM learning curve. By contrast, without any annotated mentions on the 23 unseen test event types in its training set, our transfer learning approach achieved performance comparable to that of the LSTM, which was trained on 3,000 sentences[5] with 500 annotated event mentions.

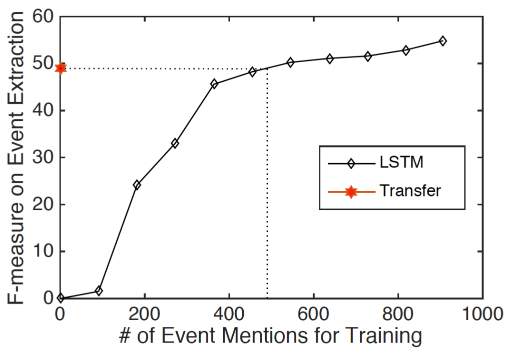

Figure 4: Comparison between Our Approach and Supervised LSTM model on 23 Unseen Event Types.

[5]The 3,000 sentences included all the sentences which even have not any event annotations.

Method	Trigger Identification			Trigger Identification + Classification			Arg Identification			Arg Identification + Classification		
	P	R	F	P	R	F	P	R	F	P	R	F
Supervised LSTM	94.7	41.8	58.0	89.4	39.5	54.8	47.8	22.6	30.6	28.9	13.7	18.6
Supervised Joint	55.8	67.4	61.1	50.6	61.2	55.4	36.4	28.1	31.7	33.3	25.7	29.0
Transfer	85.7	41.2	55.6	75.5	36.3	49.1	28.2	27.3	27.8	16.1	15.6	15.8

Table 7: Event Trigger and Argument Extraction Performance (%) on Unseen ACE Types.

6.4 Impact of AMR

Recall that we used AMR parsing output to identify triggers and arguments in constructing event structures. To assess the impact of the AMR parser (Wang et al., 2015a) on event extraction, we chose a subset of the ERE (Entity, Relation, Event) corpus (Song et al., 2015) which has ground-truth AMR annotations. This subset contains 304 documents with 1,022 annotated event mentions of 40 types. We selected the top-6 most popular event types (*Arrest-Jail, Execute, Die, Meet, Sentence, Charge-Indict*) with manual annotations of 548 event mentions as seen types. We sampled 500 negative event mentions from distinct types of clusters generated from the system (Huang et al., 2016) based on ERE training sentences. We combined the annotated events for seen types and the negative event mentions, and used 90% for training and 10% for development. For evaluation, we selected 200 sentences from the remaining ERE subset, which contains 128 *Attack* event mentions and 40 *Convict* event mentions. Table 8 shows the event extraction performances based on ground-truth AMR and system AMR respectively.

We also compared AMR analyses with Semantic Role Labeling (SRL) output (Palmer et al., 2010) by keeping only the core roles (e.g., *:ARG0, :ARG1*) from AMR annotations. As Table 8 shows, comparing the full AMR (top row) to this SRL proxy (middle row), the fine-grained AMR semantic relations such as *:location*, *:instrument* appear to be more informative for inferring event argument role labeling.

Method	Trigger Labeling			Argument Labeling		
	P	R	F_1	P	R	F_1
Perfect AMR	79.1	47.1	59.1	25.4	21.4	23.2
Perfect AMR with Core Roles only (SRL)	77.1	47.0	58.4	19.7	16.9	18.2
System AMR	85.7	32.0	46.7	22.6	15.8	18.6

Table 8: Impact of AMR and Semantic Roles on Trigger and Argument Extraction (%).

7 Related Work

Most previous event extraction methods have been based on supervised learning, using either symbolic features (Ji and Grishman, 2008; Miwa et al., 2009; Liao and Grishman, 2010; Liu et al., 2010; Hong et al., 2011; McClosky et al., 2011; Riedel and McCallum, 2011; Li et al., 2013; Liu et al., 2016) or distributional features (Chen et al., 2015; Nguyen and Grishman, 2015; Feng et al., 2016; Nguyen et al., 2016) derived from a large amount of training data, and treating event types and argument role labels as symbols. These approaches can achieve high quality for known event types, but cannot be applied to new types without additional annotation effort. In contrast, we provide a new angle on event extraction, modeling it as a generic grounding task by taking advantage of rich semantics of event types.

Some other IE paradigms such as Open IE (Etzioni et al., 2005; Banko et al., 2007, 2008; Etzioni et al., 2011; Ritter et al., 2012), Preemptive IE (Shinyama and Sekine, 2006), On-demand IE (Sekine, 2006), Liberal IE (Huang et al., 2016, 2017), and semantic frame-based event discovery (Kim et al., 2013) can discover many events without pre-defined event schema. These paradigms however rely on information redundancy, and so they are not effective when the input data only consists of a few sentences. Our work can discover events from any size of input corpus and can also be complementary with these paradigms.

Our event extraction paradigm is similar to the task of entity linking (Ji and Grishman, 2011) in semantic mapping. However, entity linking aims to map entity mentions to the same concept, while our framework maps each event mention to a specific category. In addition, Bronstein et al. (2015) and Peng et al. (2016) employ an event-independent similarity-based function for event trigger detection, which follows few-shot learning setting and requires some trigger examples as seeds. Lu and Roth (2012) design a structure preference modeling framework, which can automatically predict argument roles without any annotated data, but it relies on manually constructed patterns.

Zero-Shot learning has been widely applied in visual object classification (Frome et al., 2013; Norouzi et al., 2013; Socher et al., 2013a; Chen et al., 2017; Li et al., 2017; Xian et al., 2017; Changpinyo et al., 2017), fine-grained name tagging (Ma et al., 2016; Qu et al., 2016), relation extraction (Verga et al., 2016; Levy et al., 2017), semantic parsing (Bapna et al., 2017) and domain adaptation (Romera-Paredes and Torr, 2015; Kodirov et al., 2015; Peng et al., 2017). In contrast to these tasks, for our case, the number of seen types in event extraction with manual annotations is quite limited. The most popular event schemas, such as ACE, define 33 event types while most visual object training sets contain more than 1,000 types. Therefore, methods proposed for zero-shot visual-object classification cannot be directly applied to event extraction due to overfitting. In this work, we designed a new loss function by creating "negative" training instances to avoid overfitting.

8 Conclusions and Future Work

In this work, we take a fresh look at the event extraction task and model it as a generic grounding problem. We propose a transferable neural architecture, which leverages existing human-constructed event schemas and manual annotations for a small set of seen types, and transfers the knowledge from the existing types to the extraction of unseen types, to improve the scalability of event extraction as well as to save human effort. To the best of our knowledge, this work is the first time that zero-shot learning has been applied to event extraction. Without any annotation, our approach can achieve performance comparable to state-of-the-art supervised models trained on a large amount of labeled data. In the future, we will extend this framework to other Information Extraction problems.

Acknowledgments

This material is based upon work supported by United States Air Force under Contract No. FA8650-17-C-7715 and ARL NS-CTA No. W911NF-09-2-0053. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the United States Air Force. or the United States Government. The United States Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation here on.

References

Collin F Baker, Charles J Fillmore, and John B Lowe. 1998. The berkeley framenet project. In *Proc. COLING1998*.

L. Banarescu, C. Bonial, S. Cai, M. Georgescu, K. Griffitt, U. Hermjakob, K. Knight, P. Koehn, M. Palmer, and N. Schneider. 2013. Abstract meaning representation for sembanking. In *Proc. ACL2013 Workshop on Linguistic Annotation and Interoperability with Discourse*.

M. Banko, M. Cafarella, S. Soderland, M. Broadhead, and O. Etzioni. 2007. Open information extraction for the web. In *Proc. IJCAI2007*.

M. Banko, O. Etzioni, and T. Center. 2008. The tradeoffs between open and traditional relation extraction. In *Proc. ACL-HLT2008*.

Ankur Bapna, Gokhan Tur, Dilek Hakkani-Tur, and Larry Heck. 2017. Towards zero-shot frame semantic parsing for domain scaling. *arXiv preprint arXiv:1707.02363* .

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 *Proc. ACL2015*.

Soravit Changpinyo, Wei-Lun Chao, and Fei Sha. 2017. Predicting visual exemplars of unseen classes for zero-shot learning. In *Proc. ICCV2017*.

Long Chen, Hanwang Zhang, Jun Xiao, Wei Liu, and Shih-Fu Chang. 2017. Zero-shot visual recognition using semantics-preserving adversarial embedding network. *arXiv preprint arXiv:1712.01928* .

Y. Chen, L. Xu, K. Liu, D. Zeng, and J. Zhao. 2015. Event extraction via dynamic multi-pooling convolutional neural networks. In *Proc. ACL2015*.

O. Etzioni, M. Cafarella, D. Downey, A. Popescu, T. Shaked, S. Soderland, D. Weld, and A. Yates. 2005. Unsupervised named-entity extraction from the web: An experimental study. *Artificial Intelligence* .

O. Etzioni, A. Fader, J. Christensen, S. Soderland, and M. Mausam. 2011. Open information extraction: The second generation. In *Proc. IJCAI2011*.

X. Feng, L. Huang, D. Tang, B. Qin, H. Ji, and T. Liu. 2016. A language-independent neural network for event detection. In *Proc. ACL2016*.

A. Frome, G. Corrado, J. Shlens, S. Bengio, J. Dean, and T. Mikolov. 2013. Devise: A deep visual-semantic embedding model. In *Proc. NIPS2013*.

S. Hochreiter and J. Schmidhuber. 1997. Long short-term memory. *Neural computation* .

Y. Hong, J. Zhang, B. Ma, J. Yao, G. Zhou, and Q. Zhu. 2011. Using cross-entity inference to improve event extraction. In *Proc. ACL2011*.

L. Huang, T. Cassidy, X. Feng, H. Ji, C. Voss, J. Han, and A. Sil. 2016. Liberal event extraction and event schema induction. In *Proc. ACL2016*.

L. Huang, J. May, X. Pan, H. Ji, X. Ren, J. Han, L. Zhao, and J. Hendler. 2017. Liberal entity extraction: Rapid construction of fine-grained entity typing systems. *Big Data* .

H. Ji and R. Grishman. 2008. Refining event extraction through cross-document inference. In *Proc. ACL2008*.

Heng Ji and Ralph Grishman. 2011. Knowledge base population: Successful approaches and challenges. In *Proc. ACL-HLT2011*.

H. Kim, X. Ren, Y. Sun, C. Wang, and J. Han. 2013. Semantic frame-based document representation for comparable corpora. In *Proc. ICDM2013*.

Elyor Kodirov, Tao Xiang, Zhenyong Fu, and Shaogang Gong. 2015. Unsupervised domain adaptation for zero-shot learning. In *Proc. ICCV2015*.

Omer Levy, Minjoon Seo, Eunsol Choi, and Luke Zettlemoyer. 2017. Zero-shot relation extraction via reading comprehension. *arXiv preprint arXiv:1706.04115* .

Q. Li, H. Ji, and L. Huang. 2013. Joint event extraction via structured prediction with global features. In *Proc. ACL2013*.

Yanan Li, Donghui Wang, Huanhang Hu, Yuetan Lin, and Yueting Zhuang. 2017. Zero-shot recognition using dual visual-semantic mapping paths. *arXiv preprint arXiv:1703.05002* .

S. Liao and R. Grishman. 2010. Using document level cross-event inference to improve event extraction. In *Proc. ACL2010*.

B. Liu, L. Qian, H. Wang, and G. Zhou. 2010. Dependency-driven feature-based learning for extracting protein-protein interactions from biomedical text. In *Proc. COLING2010*.

S. Liu, Y. Chen, S. He, K. Liu, and J. Zhao. 2016. Leveraging framenet to improve automatic event detection. In *Proc. ACL2016*.

Wei Lu and Dan Roth. 2012. Automatic event extraction with structured preference modeling. In *Proc. ACL2012*.

Y. Ma, E. Cambria, and S. Gao. 2016. Label embedding for zero-shot fine-grained named entity typing. In *Proc. COLING2016*.

D. McClosky, M. Surdeanu, and C. D. Manning. 2011. Event extraction as dependency parsing. In *Proc. ACL2011*.

T. Mikolov, K. Chen, G. Corrado, and J. Dean. 2013. Efficient estimation of word representations in vector space. *CoRR* abs/1301.3781.

M. Miwa, R. Stre, Y. Miyao, and J. Tsujii. 2009. A rich feature vector for protein-protein interaction extraction from multiple corpora. In *Proc. EMNLP2009*.

T. Nguyen, K. Cho, and R. Grishman. 2016. Joint event extraction via recurrent neural networks. In *Proc. NAACL-HLT2016*.

T. Nguyen and R. Grishman. 2015. Event detection and domain adaptation with convolutional neural networks. In *Proc. ACL2015*.

M. Norouzi, T. Mikolov, S. Bengio, Y. Singer, J. Shlens, A. Frome, G. Corrado, and J. Dean. 2013. Zero-shot learning by convex combination of semantic embeddings. *arXiv preprint arXiv:1312.5650* .

M. Palmer, D. Gildea, and N. Xue. 2010. Semantic role labeling. *Synthesis Lectures on Human Language Technologies* .

Haoruo Peng, Yangqiu Song, and Dan Roth. 2016. Event detection and co-reference with minimal supervision. In *Proc. EMNLP2016*.

Kuan-Chuan Peng, Ziyan Wu, and Jan Ernst. 2017. Zero-shot deep domain adaptation. *arXiv preprint arXiv:1707.01922* .

J. Pustejovsky. 1991. The syntax of event structure. *Cognition* .

L. Qu, G. Ferraro, L. Zhou, W. Hou, and T. Baldwin. 2016. Named entity recognition for novel types by transfer learning. In *Proc. ACL2016*.

S. Riedel and A. McCallum. 2011. Fast and robust joint models for biomedical event extraction. In *Proc. EMNLP2011*.

A. Ritter, O. Etzioni, and S. Clark. 2012. Open domain event extraction from twitter. In *Proc. SIGKDD2012*.

Bernardino Romera-Paredes and Philip Torr. 2015. An embarrassingly simple approach to zero-shot learning. In *Proc. ICML2015*.

S. Sekine. 2006. On-demand information extraction. In *Proc. COLING-ACL2006*.

Y. Shinyama and S. Sekine. 2006. Preemptive information extraction using unrestricted relation discovery. In *Proc. HLT-NAACL2006*.

R. Socher, M. Ganjoo, C. Manning, and A. Ng. 2013a. Zero-shot learning through cross-modal transfer. In *Proc. NIPS2013*.

R. Socher, A. Perelygin, J. Wu, J. Chuang, C. Manning, A. Ng, and C. Potts. 2013b. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proc. EMNLP2013*.

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 *Proc. NAACL-HLT2015 Workshop on on EVENTS*.

P. Verga, D. Belanger, E. Strubell, B. Roth, and A. McCallum. 2016. Multilingual relation extraction using compositional universal schema. In *Proc. NAACL2016*.

C. Wang, N. Xue, and S. Pradhan. 2015a. Boosting transition-based amr parsing with refined actions and auxiliary analyzers. In *Proc. ACL2015*.

Chuan Wang, Nianwen Xue, Sameer Pradhan, and Sameer Pradhan. 2015b. A transition-based algorithm for amr parsing. In *HLT-NAACL*.

Yongqin Xian, Bernt Schiele, and Zeynep Akata. 2017. Zero-shot learning-the good, the bad and the ugly. *arXiv preprint arXiv:1703.04394* .

Z. Zhong and H. T. Ng. 2010. It makes sense: A wide-coverage word sense disambiguation system for free text. In *Proc. ACL2010*.