Joint Event Extraction with Hierarchical Policy Network

Peixin Huang[1] **Xiangzhao**[1,2,*] **Ryuichi Takanobu**[3] **Zhen Tan**[1] **Weidong Xiao**[1,2,*]
[1]National University of Defense Technology, Changsha, China
[2]Collaborative Innovation Center of Geospatial Technology, Wuhan, China
[3]Tsinghua University, Beijing, China
{huangpeixin15, xiangzhao, tanzhen08a, wdxiao}@nudt.edu.cn;
gxly19@mails.tsinghua.edu.cn

Abstract

Most existing work on event extraction (EE) either follows a pipelined manner or uses a joint structure but is pipelined in essence. As a result, these efforts fail to utilize the information interactions among event triggers, event arguments and argument roles, and also causes information redundancy. In view of this, we propose to exploit the role information of the arguments in an event and devise a Hierarchical Policy Network (*HPNet*) to perform joint EE. The whole EE process is fulfilled through a two-level hierarchical structure which consists of two policy networks for event detection and argument detection respectively, so that the deep information interactions among the subtasks are realized and it is more natural to deal with multiple events issue. Extensive experiments on ACE2005 and TAC2015 demonstrate the superiority of *HPNet*, leading to the state-of-the-art performance and is more powerful for sentences with multiple events.

1 Introduction

Event extraction (EE) plays an important role in various real-life applications, such as information retrieval and news summarization (Glavas and Snajder, 2014; Daniel et al., 2003). It aims to discover events with triggers and their corresponding arguments. Typically, EE contains several subtasks: trigger identification, trigger classification, event argument identification and argument role classification.

Some researchers handle these subtasks in a pipelined manner, i.e., perform trigger prediction and then identify arguments in separate stages, assuming that gold standard entities are provided (McClosky et al., 2011; Chen et al., 2015; Nguyen and Grishman, 2018; Yang et al., 2019). However, this staged manner has no strategy to make advantage of the deeper information interactions among these subtasks, so the upstream and the downstream subtasks cannot interfere with each other to improve their decisions. Although there have been joint approaches which aim to build a joint extractor (Yang and Mitchell, 2016; Nguyen and Nguyen, 2019; Zhang et al., 2019), they still follow a pipelined framework by first jointly predicting entities and triggers, and then scanning every entity-event pair for arguments and argument roles. One of the same limitations they share is that they produce redundant entity-event pairs information, thus bringing in possible errors. Another is the mismatch between arguments and triggers when the sentence contains multiple events. For example, consider the following sentence:

*In Baghdad, a cameraman **died** when an American tank **fired** on the Palestine hotel.*

In this sentence, "cameraman" is not only a Victim argument of event Die (trigger "died"), but also a Target argument of event Attack (trigger "fired"). However, as "cameraman" is far away from the trigger "fired" in the sentence, the extractor may fail to identify "cameraman" as one argument for event Attack.

In view of this, we propose a Hierarchical Policy Network (*HPNet*) to jointly solve the subtasks of EE, where 1) trigger identification and trigger classification are solved together, and 2) event argument identification and argument role classification are solved together, by a hierarchy of an event-level policy network (PN) and an argument-level PN. This two-level hierarchical structure works as follows. During the scanning from the beginning to the end of a sentence, the event-level PN first detects the trigger and classifies its event type at each word. Once a certain event is detected, an argument-level PN is triggered to detect participating arguments and classify the roles they play in the current event. When the argument-level process under this event is finished, the event-level PN continues its scan to search for

* Corresponding author

other events in the sentence until it scans to the end. The learning of the EE subtasks is formulated as a sequential decision problem, which can be addressed by policy gradient method (Sutton et al., 1999).

HPNet realizes deep information interactions among the subtasks by passing state representations and rewards in the two-level hierarchical structure: the event-level PN passes fine-grained semantic information to the argument-level PN through state representations when triggering it, and the argument-level PN passes rewards back to the event-level PN to convey how well it is completed. Besides, the event scheme information from the event-level decision also benefits the argument-level decision. As HPNet uses hierarchical structure to detect the event first and then identify its participating arguments, redundant information and the mismatch issues can be potentially avoided.

To sum up, our main contribution is at least three-fold: (1) To our knowledge, this is the first work applying policy network, a deep reinforcement learning method, to extract event. (2) We utilize a two-level hierarchical structure to realize joint EE. With well designed state representations, rewards and event scheme retrieval table, this structure fully explores the deep information interactions among EE subtasks and addresses the multiple events and mismatch issues. (3) We design comprehensive experiments on the widely used ACE2005 and TAC2015 datasets. The experimental results show that our method *HPNet* significantly outperforms other state-of-the-art methods, especially in dealing with multiple events.

This paper proceeds as follows. First we discuss the work that is related to pipelined EE, joint EE and policy network (§2). Next we present *HPNet*, a two-level policy network based structure that contains a hierarchy of an event-level PN and an argument-level PN, followed with their hierarchical training details (§3). Then we describe the experiments on two benchmark datasets ACE2005 and TAC2015, followed by the results and discussions (§4). We finish with a conclusion of the paper (§5).

2 Related Work

Pipelined Event Extraction The early work on EE has focused on the pipelined approach which performs the subtasks of EE in separate stages (McClosky et al., 2011; Chen et al., 2015; Nguyen and Grishman, 2018). They either rely on manually designed features or use convolutional neural networks (CNNs) to construct completely separate classifiers for trigger labeling and argument role classification, regarding entities as being provided by external annotators.

Recently, there are methods that extract entities and events to utilize the dependency of these two subtasks. Above work exploits different structured prediction methods, including Markov Logic Networks (Poon and Vanderwende, 2010), dual decomposition (Riedel and McCallum, 2011), structured perceptron (Li et al., 2013) and attention-based graph CNN (Liu et al., 2018). There are also methods that develop inference models for events and argument roles, which include structured predictions with Markov Logic (Riedel et al., 2009) and parameter sharing (Sha et al., 2018). However, the above studies are limited to the modeling of only two subtasks, either assuming the golden annotations for other subtasks or simply ignoring them.

Joint Event Extraction There have been efforts towards joint modeling of event triggers, event types, arguments and argument roles. Yang and Mitchell (2016) consider information interactions among these subtasks through a two-stage framework, in which the k-best outputs of triggers and entities are first selected, and re-ranking is then used for joint inference. Nguyen and Nguyen (2019) propose to share common encoding layers to enable the information sharing, and decode the event triggers, arguments and roles separately. Zhang et al. (2019) use transition systems for the dependency parsing of the input sentence, and decode the labels for subtasks with score ranking. However, as mentioned in Section 1, these methods follow a pipelined decoding order, which makes them face the information redundancy and the mismatch problem. Moreover, they all strongly rely on the annotations of training data.

Policy Network Policy network is one of the most important deep reinforcement learning methods, which has been frequently witnessed in recent information extraction work. Zhang et al. (2018) employ PN for structure discovery in sentence representations prepared for the downstream text classification. Qin et al. (2018) propose a policy network-based distant supervision relation extraction method in which they use PN to define a policy for redistribution of false positives. Feng et al. (2018) use PN to construct an instance selector to obtain a weak supervision signal for relation classification from noisy data. Takanobu et al. (2019) propose to apply hierarchical reinforcement learning to relation extraction task, which gives highly competitive results. We are inspired by these methods. To the best of our knowledge, we are the first to incorporate hierarchical policy network for event extraction task.

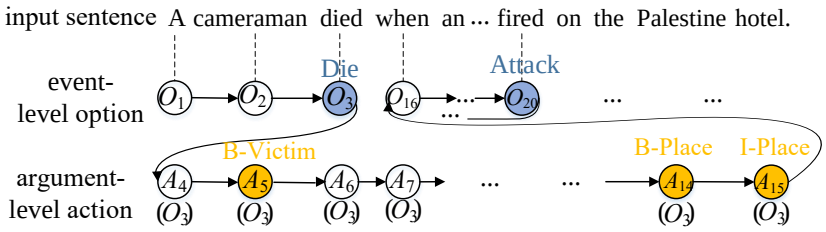

Figure 1: An example of EE process by *HPNet* on a sentence containing two events (Die and Attack). The straight arrow indicates the event-level and the argument-level process. The curved arrow marks a transition between the event-level and the argument-level process. O_3 indicates the event-level option (Die) for word "died" at time step 3, while A_5 is the argument-level action (B-Victim) for word "cameraman" at time step 5 under the event-level option O_3.

3 Hierarchical Policy Network

3.1 Overview

We introduce a reward-driven, policy-based method *HPNet* to hierarchically detect events and their participating arguments. As shown in Figure 1, given a sentence, *HPNet* realizes the entire extraction process as follows. As the agent scans this sentence sequentially, the event-level PN keeps following a policy to sample an option (indicating a non-trigger word or a trigger with a specific event type) at each time step. Once a certain event is detected, the agent transfers to the argument-level PN and follows a policy to select an action (assigning an exact argument tag). When the agent finishes its scanning for participating arguments under the current event, it continues to scan the rest of the sentence for other events. Since a reward can be computed once the option/action is sampled, the process can be naturally addressed by policy gradient method (Sutton et al., 1999). By detecting an event first, and then detecting participating arguments for that event, our method can achieve effective EE even when multiple events exist.

3.2 Event-level Policy Network

Given the input sentence $S = w_1, w_2, ..., w_L$, the event-level PN aims to detect the event type that the trigger word w_i triggers. Specifically, at current word/step t, the policy network adopts a stochastic policy μ over options and uses rewards to guide the policy learning. It samples an option with the probability at current state whose representation is history-dependent. We briefly introduce option, state, policy and reward as follows:

Option. The event-level option o_t^e is sampled from an option set $O^e = \{NE\} \cup \mathcal{E}$, where NE indicates non-trigger word, and $\mathcal{E}$ is the event type set which is predefined in the dataset, indicating which type the current trigger is triggering.

State. The state $\mathbf{s_t}^e \in S^e$ of the event-level process is history-dependent, encoding the previous environment as well as the current input. $\mathbf{s_t}^e$ is the concatenation of 1) the state $\mathbf{s_{t-1}}$ from the last time step (where $\mathbf{s_{t-1}} = \mathbf{s_{t-1}}^e$ if the agent launches an event-level PN at time step $t-1$, otherwise $\mathbf{s_{t-1}} = \mathbf{s_{t-1}}^r$), 2) the event type vector $\mathbf{v_t}^e$ which is learned from the latest option $o_{t'}^e$ that satisfies $o_{t'}^e \in \mathcal{E}$ and 3) the hidden state vector $\mathbf{h_t}$ over the current input word embedding $\mathbf{w_t}$, represented by:

$$\mathbf{s_t}^e = \mathbf{s_{t-1}} \oplus \mathbf{v_t}^e \oplus \mathbf{h_t}, \tag{1}$$

We obtain the hidden state vector $\mathbf{h_t}$ through a sequential word-level Bi-LSTM:

$$\begin{aligned} \mathbf{h_t} &= \overrightarrow{\mathbf{h_t}} \oplus \overleftarrow{\mathbf{h_t}}, \\ \overrightarrow{\mathbf{h_t}} &= \overrightarrow{LSTM}(\overrightarrow{\mathbf{h_{t-1}}}, \mathbf{w_t}), \\ \overleftarrow{\mathbf{h_t}} &= \overleftarrow{LSTM}(\overleftarrow{\mathbf{h_{t+1}}}, \mathbf{w_t}). \end{aligned} \tag{2}$$

Finally, we use MLP to represent the state as a continuous real-valued vector $F^e(\mathbf{s_t}^e)$.

Policy. The stochastic policy for event detection over the possible options $\mu : S^e \rightarrow O^e$ samples an option $o_t^e \in O^e$ according to a probability distribution:

$$\mu(o_t^e|\mathbf{s_t}^e) = softmax(\mathbf{W}^e F^e(\mathbf{s_t}^e) + \mathbf{b}^e), \tag{3}$$

Event Types	Event Argument Roles
Attack	Attacker, Target, Instrument, Place, Time-*
Die	Agent, Victim, Instrument, Place, Time-*
Phone-Write	Entity, Place, Time-*
Sue	Adjudicator, Crime, Defendant, Plaintiff, Prosecutor, Place, Time-*

Table 1: Event scheme table which specifies the possible argument roles for each event type. Take 4 event types as examples. Time-* means argument roles related to time, such as Time-Within, Time-Before.

where $\mathbf{W}^e$ and $\mathbf{b}^e$ are the parameters, $F^e(\mathbf{s_t}^e)$ is the state feature vector. During training, the option is sampled according to the probability in Eq 3. During test, the option with the maximal probability will be chosen (i.e.,$o_t^e* = \arg\max_{o^e} \mu(o^e|\mathbf{s_t}^e)$).

Reward. An intuition of rewards is that the event-level PN ultimately aims to recognize and categorize events, viewing triggers as intermediate results. Once an event-level option o_t^e is sampled, the agent will receive an instant reward which estimates the short-term return under option o_t^e. The instant reward is computed by consulting the gold-standard event type annotations S_t^e of sentence S:

$$r_t^e = sgn(o_t^e = S_t^e) \cdot I(NE) + \alpha \cdot sgn(o_t^e = S_t^e) \cdot (1 - I(NE)), \tag{4}$$

where $sgn(\cdot)$ is the sign function, $I(NE)$ is a switching function which distinguishes the reward of the trigger and the non-trigger word:

$$I(NE) = \begin{cases} 1 & \text{if} \quad o_t^e \neq NE \\ 0 & \text{if} \quad o_t^e = NE, \end{cases} \tag{5}$$

and α is a bias weight. The smaller α ($\alpha < 1$) is, the less reward on non-trigger words, which will prevent the model from learning a trivial policy to predict all words as NE (non-trigger words).

The transition of the event-level PN depends upon option o_t^e. If $o_t^e = NE$ at a time step, the agent will continue at a new event-level PN state. Otherwise a specific event is detected and the agent will launch a new subtask and transfer to the argument-level PN to detect participating arguments of this event. Hence, the agent starts the argument-level options with an event-initialized state, and it will not transfer to the event-level PN until all the argument-level options under the current event o_t^e are finished. The event-level options are sampled until the option of the last word in S. When the agent finishes all the event-level options, it receives a terminal reward r_{ter}^e. This delayed reward at the terminal state is defined by the sentence-level event detection performance:

$$r_{ter}^e = F_1(S), \tag{6}$$

where $F_1(\cdot)$ represents $F1$ scores which is the harmonic mean of sentence-level *precision* and *recall* of event detection results.

3.3 Argument-level Policy Network

Once a specific event is detected at a time step $t^{'}$ (i.e., $o_{t^{'}}^e \in \mathcal{E}$), the agent transfers to the argument-level PN to predict the role that each argument plays under the event $o_{t^{'}}^e$. Specifically, at each word/step t, the argument-level PN adopts a stochastic policy π over actions and uses rewards to guide the learning for participating arguments as well as their roles under the current event. To pass more fine-grained event information for the argument decision, the option $o_{t^{'}}^e$ as well as the state representation $\mathbf{s_{t^{'}}}^e$ from the event-level process are taken as additional inputs throughout the whole argument-level process. We introduce action, state, policy and reward as follows:

Action. The argument-level action a_t^r is to assign an argument tag to the current word. a_t^r is selected from an action space $A^r = (\{B, I\} \times \mathcal{R}) \cup \{O, NR\}$, where B/I represents the position information (Begin, Inside) of a word in an argument, O tags arguments unrelated to the current event, $\mathcal{R}$ is a predefined argument role set in the dataset and NR tags non-argument words. Note that, the same argument may be assigned with different tags depending on distinct event types concerned at the moment. In this way, multiple events and mismatch issues can be naturally solved. We use Figure 2 to illuminate.

Then we take advantage of the event scheme information which specifies the possible argument roles for each event type to filter the action space. Event schema describes the specific argument roles for each event type, as shown in Table 1. Ding et al. (2018) use event scheme to build a mask matrix to filter out irrelevant argument roles. Inspired by their work, we devise a predefined event scheme

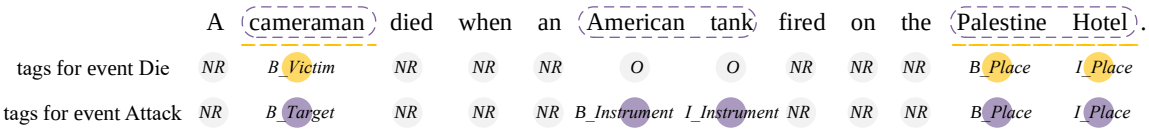

Figure 2: Examples of argument tags under different moments for the sentence in Section 1. "cameraman" and "Palestine hotel" are both participating arguments in event Die and Attack, while "cameraman" has different roles under distinct events. Also, "American tank" is a participating argument of event Attack but is irrelevant with event Die. As a result, both "cameraman" and "American tank" have different tags under different moments.

retrieval table $R^{|\mathcal{E}|*|\mathcal{R}|}$ to narrow down the argument role set $\mathcal{R}$. Finally, the filtered action space $A^r = (\{B, I, O\} \times R[o^e_{t'}]) \cup \{NR\}$. A motivation for this event scheme retrieval table is that the implicit argument information from the event-level decision can be fully utilized. Besides, by this retrieval table, the action space is narrowed down, enabling more efficient agent.

State. The state $\mathbf{s_t}^r \in S^r$ of the argument-level process is also history-dependent, encoding the previous environment, the initial event environment and the current input. $\mathbf{s_t}^r$ is represented by the concatenation of 1) the state $\mathbf{s_{t-1}}$ from previous time step (where $\mathbf{s_{t-1}}$ may be a state either from the event-level PN or the argument-level PN), 2) the argument tag vector $\mathbf{v_t}^r$ which is a learnable embedding of the action a^r_{t-1}, 3) the event state representation $G^e(\mathbf{s_{t'}}^e)$ and 4) the hidden vector $\mathbf{h_t}$ obtained from the similar Bi-LSTM in Eq. 2, as follows:

$$\mathbf{s_t}^r = \mathbf{s_{t-1}} \oplus \mathbf{v_t}^r \oplus G^e(\mathbf{s_{t'}}^e) \oplus \mathbf{h_t}, \tag{7}$$

The state is then represented as a real-valued vector $F^r(\mathbf{s_t}^r)$ with MLP.

Policy. With event type $o^e_{t'}$ as an additional input, the stochastic policy for argument detection over actions $\pi : S^r \rightarrow A^r$ selects an action a^r_t according to a probability distribution:

$$\pi(a^r_t|\mathbf{s_t}^r; o^e_{t'}) = softmax(\mathbf{o_{t'}}^e \mathbf{W}^r F^r(\mathbf{s_t}^r) + \mathbf{b}^r), \tag{8}$$

$$\mathbf{o_{t'}}^e = \mathbf{W}_\mu[o^e_{t'}], \tag{9}$$

where $\mathbf{W}^r$ and $\mathbf{b}^r$ are the parameters, $F^r(\mathbf{s_t}^r)$ is the argument-level state feature vector, $\mathbf{o_{t'}}^e$ is the representation of the event $o^e_{t'}$ and $\mathbf{W}_\mu$ is an array of $|\mathcal{E}|$ matrices. During training and test, the actions will be sampled in the similar way as the options of event-level PN.

Reward. Once an argument-level action a^r_t is selected, the agent will receive an instant reward r^r_t, provided by consulting the gold argument annotation $S^r_t(o^e_{t'})$ conditioned on the predicted event type $o^e_{t'}$ [1]. The reward is computed as below:

$$r^r_t = sgn(a^r_t = S^r_t(o^e_{t'})) \cdot I(NR) + \beta \cdot sgn(a^r_t = S^r_t(o^e_{t'})) \cdot (1 - I(NR)), \tag{10}$$

where $I(NR)$ is a switching function which distinguishes the reward of argument and non-argument word:

$$I(NR) = \begin{cases} 1 & \text{if } a^r_t \neq NR \\ 0 & \text{if } a^r_t = NR, \end{cases} \tag{11}$$

and β is a bias weight. The smaller β ($\beta < 1$) means the less reward on the non-argument word, preventing the agent from learning a trivial policy to select all the actions as NR.

The actions are selected until the action of the last word. When the agent finishes all the argument-level actions under the current event $o^e_{t'}$, it receives a terminal reward r^r_{ter}:

$$r^r_{ter} = \begin{cases} 1 & \text{if } a^r_t = S^r_t(o^e_{t'}), \forall t \; under \; o^e_{t'} \\ -1 & \text{else.} \end{cases} \tag{12}$$

3.4 Hierarchical Training

To optimize the event-level PN and the argument-level PN, we aim to maximize the expected cumulative discounted rewards from the option and action that the agent samples following the event-level policy $\mu(*)$ and the argument-level policy $\pi(*|o_{t'})$ at each time step t, which can be computed as follows:

[1] There we assume that the event $o^e_{t'}$ is correctly predicted. Otherwise, all the argument-level rewards will be set to 0.

Dataset	ACE2005	TAC2015
Event types	33	38
Role types	28	-
Training documents	529	132
Development documents	30	26
Test documents	40	202

Table 2: Statistics of the datasets.

$$
\begin{aligned}
J(\Theta^e) &= \mathbb{E}_{\mathbf{s}^e, o^e, r^e \sim \mu(o^e|\mathbf{s}^e)}[\Sigma_{k=t}^{T^e} \gamma^{k-t} r_k^e], \\
J(\Theta^r; o_{t'}^e) &= \mathbb{E}_{\mathbf{s}^r, a^r, r^r \sim \pi(a^n|\mathbf{s}^n; o_{t'}^e)}[\Sigma_{k=t}^{T^r} \gamma^{k-t} r_k^n],
\end{aligned} \tag{13}
$$

where $\gamma \in [0, 1]$ is a discount rate, T^e is the total time steps that the event-level process takes before it terminates and T^r is the ended time step of the argument-level process.

We then decompose the cumulative rewards as:

$$
\begin{aligned}
R^e(\mathbf{s_t}^e, o_t^e) &= \mathbb{E}[\gamma^N R^e(\mathbf{s_{t+N}}^e, o_{t+N}^e) + \Sigma_{j=0}^{N-1} \gamma^j r_{t+j}^e | \mathbf{s_t}^e, o_t^e], \\
R^r(\mathbf{s_t}^r, a_t^r; o_{t'}^e) &= \mathbb{E}[\gamma R^r(\mathbf{s_{t+1}}^r, a_{t+1}^r; o_{t'}^e) + r_t^r | \mathbf{s_t}^r, a_t^r],
\end{aligned} \tag{14}
$$

where N is the time steps that the argument-level process continues under option o_t^e, so the next option of the agent is o_{t+N}. If $o_t^e = NE$, then $N = 1$.

The optimization is achieved by using policy gradient method (Sutton et al., 1999) and the REINFORCE algorithm (Williams, 1992), which updates parameters with the following stochastic gradients:

$$
\begin{aligned}
\nabla_{\Theta^e} J(\Theta^e) &= \mathbb{E}_{\mathbf{s^e}, o^e, r^e \sim \mu(o^e|\mathbf{s}^e)}[R^e(\mathbf{s_t}^e, o_t^e) \nabla_{\Theta^e} \log \mu(o^e|\mathbf{s}^e)], \\
\nabla_{\Theta^r} J(\Theta^r; o_{t'}^e) &= \mathbb{E}_{\mathbf{s^r}, a^r, r^r \sim \pi(a^r|\mathbf{s}^r; o_{t'}^e)}[R^r(\mathbf{s_t}^r, a_t^r; o_{t'}^e) \nabla_{\Theta^r} \log \pi(a^r|\mathbf{s}^r; o_{t'}^e)].
\end{aligned} \tag{15}
$$

We describe the entire training process of *HPNet* in Appendix A.

4 Experiments

4.1 Experimental Setup

Datasets and Evaluations We perform evaluation on two standard datasets: Automatic Content Extraction program of 2005 (ACE2005) and Event Nugget data of TAC 2015 (TAC2015) (Mitamura et al., 2015). They contain text documents from a variety of sources, such as newswire reports and discussion forums. While ACE2005 provides annotations for all subtasks involved (event triggers and event arguments), TAC2015 provides only annotations for event triggers. For ACE2005, we use the same setup as Liu et al. (2018). For TAC2015, we use the official training and test sets, and manually reserve some documents in the training set for development. Descriptive statistics of two datasets are provided in detail in Table 2. We utilize the Stanford CoreNLP toolkit [2] to do the preprocessing for the sentences (i.e., POS tagging, chunking and dependency parsing).

We follow standard evaluation criteria for events and arguments (Ji and Grishman, 2008). A trigger is correct if its span and event type match a reference trigger. An argument is correct if its span, event type and role type match a reference argument. Our evaluations report identification and classification of event triggers and arguments. An event trigger or argument is correctly identified if its offsets match a reference trigger or argument, and correctly classified if there is an additional match in event type or role. We report micro-averaged *Precision* (*P*), *Recall* (*R*) and *F1 score* (*F1*) in all evaluations.

Parameters and Training Details All hyper-parameters are tuned on the development set. The word embeddings are pre-trained by Glove algorithm (Pennington et al., 2014). The vectors of event type and argument tag are initialized randomly. The bias weight $\alpha = 0.05$ in Eq. 4, $\beta = 0.1$ in Eq. 10 and the discount rate $\gamma = 0.9$.

Baselines We compare our method with several state-of-the-art methods, which can be divided into two categories: the pipelined methods and the jointly learning methods. For the pipelined methods, the entity candidates are obtained through a CRF entity extractor (Yang and Mitchell, 2016). These methods include: (1) *StagedMaxent* is a typical feature-based two-stage approach that first detects event triggers

[2] http://stanfordnlp.github.io/CoreNLP

Model	ACE2005												TAC2015					
	Event Trigger Identification (%)			Event Trigger Classification (%)			Event Argument Identification (%)			Argument Role Classification (%)			Event Trigger Identification (%)			Event Trigger Classification (%)		
	P	*R*	F_1	*P*	*R*	F_1	*P*	*R*	F_1	*P*	*R*	F_1	*P*	*R*	F_1	*P*	*R*	F_1
StagedMaxent	73.1	65.4	69.0	70.1	63.3	66.5	75.0	20.3	31.9	71.0	19.3	30.3	69.7	46.8	56.0	65.4	44.5	53.0
TwoStageBeam	74.8	60.7	67.0	72.8	53.7	61.8	73.9	27.1	39.7	66.9	25.5	36.9	72.1	43.8	54.5	66.0	41.6	51.0
DMCNN	79.6	67.2	72.9	74.3	62.9	68.1	69.1	51.8	59.2	62.8	45.0	52.4	77.4	48.7	59.8	71.3	45.8	55.8
dbRNN*	-	-	-	-	-	69.6	-	-	57.2	-	-	50.1	-	-	-	-	-	-
JMEE-NP	78.5	71.3	74.7	74.1	69.1	71.5	62.3	53.5	57.6	58.9	47.3	52.5	74.3	51.4	60.8	69.7	47.0	56.1
JointTransition	76.2	75.7	75.9	74.0	73.7	73.8	59.6	56.3	57.9	54.3	51.8	53.0	73.8	56.4	63.9	68.3	53.0	59.7
TAC2015Best*	-	-	-	-	-	-	-	-	-	-	-	-	82.0	52.0	63.7	75.2	47.7	58.4
HPNet	81.3	77.2	**79.2**	80.1	75.7	**77.8**	70.2	53.8	**60.9**	64.6	50.7	**56.8**	78.2	55.6	**65.0**	70.9	54.8	**61.8**

Table 3: Main results on the ACE2005 and TAC2015. The comparison between *HPNet* and *JointTransition* is significant with $p < 0.05$. * indicates the results adapted from the original paper. For TAC2015, "Event Trigger Identification" corresponds to the "Span" metric and "Event Trigger Classification" corresponds to the "Type" metric reported in official evaluation (cf. Section 4.2 for detailed analysis).

and then event arguments (Yang and Mitchell, 2016). (2) *TwoStageBeam* is a pipelined model with structure perception and global features (Li et al., 2013). (3) *DMCNN* is the most successful pipelined model for EE, which uses dynamic multi-pooling convolutional neural networks (Chen et al., 2015). The joint methods include: (4) *dbRNN* adds dependency bridges over Bi-LSTM for joint EE. It uses Stanford constituency parser [3] to parse every sentence and takes predicted NP nodes as candidate arguments (Sha et al., 2018). (5) *JMEE-NP* is our reimplementation of *JMEE* (Liu et al., 2018) method, which also takes NP nodes from Stanford constituency parser as candidate arguments. (6) *JointTransition* performs joint decoding for the subtasks using a neural transition-based method (Zhang et al., 2019). This method currently has the state-of-the-art performance on the ACE2005 dataset. We also include: (7) *TAC2015Best* corresponds to reported results for Hong et al. (2015) including semi-supervised learning, achieving the best performance in TAC2015 Evaluation.

4.2 Main Results

Table 3 shows the results on ACE2005 and TAC2015. From the table, we observe that: (1) *HPNet* steadily outperforms all the baselines significantly. Compared with them, *HPNet* gains at least 4.0% absolute *F1 score* improvement in triggers, 3.8% in arguments on ACE2005, and at least 2.1 % absolute *F1 score* improvement in triggers on TAC2015, respectively. (2) Regarding argument detection, the joint system with predicted entity mentions (i.e., *dbRNN* with 57.2% and 50.1% for arguments and argument roles respectively) performs worse than that with perfect entity mentions (i.e., 67.7% and 58.7% for arguments and argument roles respectively in Sha et al. (2018)). This is consistent with the significant performance drop of *JMEE-NP* compared to *JMEE* (Liu et al., 2018) with gold entity mentions. Such evidences reveal that they rely heavily on entity annotations from external tools. However, our *HPNet* is significantly better than both methods with respect to all the *F1 scores* in Table 3, with no need for gold entity tagging. (3) In general, the joint systems outperform the pipelined systems on both datasets for all the subtasks, which proves the advancement of joint modeling to some extend. (4) Compared with the current state-of-the-art method *JointTransition*, our *HPNet* gives a lower *Recall* in argument detection. The reason may be that the event type information from the event-level process limits the decision of arguments to some extend, especially with the event scheme retrieval table. Nevertheless, our HPNet can balance the *Precision* and *Recall*, and gains the highest *F1 score* with the hierarchical structure and policy network.

4.3 Effect of Hierarchical Structure

In this section, we prove the effectiveness of our two-level hierarchical structure for building the deep interactions between events and arguments, we design a variant of *HPNet*, *HPNet-Argument* which removes the argument-level PN and only keeps the event-level PN. We investigate the performance on event detection and report the results of event trigger classification from the event-level PN. Moreover,

[3] https://nlp.stanford.edu/software/lex-parser.shtml

Model	1/1(%)	1/N(%)	all(%)
Embedding+T	68.1	25.5	59.8
CNN	72.5	43.1	66.3
DMCNN	74.3	50.9	68.1
JMEE	75.2	72.7	73.7
HPNet-Argument	**85.8**	43.8	72.0
HPNet	83.7	**73.6**	**77.8**

Table 4: Event detection results on single event sentences (1/1) and multiple event sentences (1/N) (cf. Section 4.3 for detailed analysis).

Stage	Model	A(%)	B(%)	all(%)
Event	*Traditional*	68.8	14.3	61.2
	Reduced DMCNN	70.7	33.1	64.9
	HPNet	**87.7**	**69.9**	**77.8**
Argument	*Traditional*	58.5	22.2	34.6
	Reduced DMCNN	59.5	30.7	40.2
	HPNet	**66.5**	**50.4**	**56.8**

Table 5: Event detection and argument detection results on clear sample (A) and new sample (B) (cf. Section 4.4 for detailed analysis).

to prove that HPNet could alleviate multiple events issue through hierarchical structure, we divide the test data (all) of ACE2005 into two parts (1/1 and 1/N) following the previous work (Chen et al., 2015), where 1/1 means that only one event or one argument plays a role in one sentence, and 1/N contains all other cases. We perform evaluations on two test sets separately. Statistically, 27.3% of the sentences have multiple events and 23.2% of arguments attend multiple events within one sentence in the test set.

Table 4 shows the performance (*F1 score*) of *HPNet*, the variant, *DMCNN* and two baseline systems *Embedding+T* and *CNN* (Chen et al., 2015). *Embedding+T* uses word embeddings and the traditional sentence-level features (Li et al., 2013), while *CNN* is similar to DMCNN, except that it applies the standard pooling mechanism instead of the dynamic multi-pooling method. For comparison, we also include *JMEE* with gold entity mentions, which is the state-of-the-art system dealing with multiple events issue. From Table 4, we can see that our method achieves the best performance on both 1/N dataset and all. There is a significant performance drop (2.1%) of *HPNet* compared with *HPNet-Argument* on 1/1. The reason may be that, in 1/1 test set, the interactions between two policy networks have almost no influence on event detection as 1/1 has only sentences containing one event. However, comparing to the variant *HPNet-Argument*, our *HPNet* yields at least 29.8% improvement on 1/N, indicating that our two-level structure captures the interactions among the subtasks and event-level PN also benefits from the argument-level process. Moreover, our *HPNet* yields a substantial improvement over all baselines as well as the variant on 1/N dataset, implying that our structure can better capture the event type information and is more powerful in extracting multiple events. Therefore, our two-level hierarchical structure indeed enhances the information interactions among the subtasks and alleviates multiple events issue.

4.4 Effect of Policy Network

In this section, we prove the effectiveness of policy network for improving the generalization ability of *HPNet*. We divide the triggers and arguments in the test data (all) of ACE2005 into two parts (A and B) following previous work (Chen et al., 2015), where situation A represents the triggers or arguments appearing in both the training set and the test set, and situation B denotes all other cases. Statistically, 34.9% of the triggers and 83.1% of the arguments in test set never appear to be the same event type in the training set. We perform evaluations and report the results on event detection and argument detection.

Table 5 gives the results (*F1 score*) of *HPNet*, two baselines *Traditional* and *Reduced DMCNN*. *Traditional* is described in previous work (Li et al., 2013) as the traditional method, and *Reduced DMCNN* is simplified DMCNN which only uses word embedding as lexical feature (Chen et al., 2015). In Table 5, our *HPNet* makes significant improvements compared with the above methods in the classification of both events and arguments under all situations. Noticeably, there is a significant gap between the performance on situation A and that on situation B as all the methods suffer from data sparsity and could not adequately handle a situation where a trigger or an argument does not appear in the training data. However, significant improvements (at least 36.8% for events and 19.7% for arguments) of *HPNet* can be observed on situation B. This occurs because policy network adaptively communicates with the environments (also new environments) to obtain the knowledge and uses rewards to guide the leaning for the subtasks. As a result, it is more powerful in dealing with generalized data and improving the generalization ability of *HPNet*.

4.5 Error Analysis

To analyze the operation of our method on event detection, we notice from Table 3 that the event trigger classification performance is quite close to that of the trigger identification, suggesting that the main errors lie in the event trigger identification. So we examine the predicted results of the event to determine the contributions of each event type to the trigger identification errors. There are two types of errors

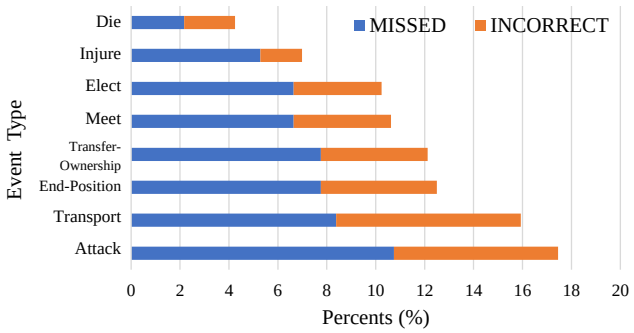

Figure 3: Top eight event type for trigger identification errors (cf. Section 4.5 for detailed analysis).

(MISSED and INCORRECT) where MISSED denotes that a trigger is missed during evaluation and INCORRECT indicates that a trigger is incorrectly detected. We report the top eight event types which appear in the errors and their corresponding percents over the total numbers of errors in Figure 3. These top eight event types account for more than 90% of all errors.

As we can see from the Figure 3, the majority of errors relates to missing triggers. The rest INCORRECT errors involve spurious triggers (e.g., a non-trigger word is classified to a specific event type) and misclassified triggers (e.g., a trigger of event End-Position is mistaken for event Attack). Attack and Transport are the event types that account for a large percent of both errors. A closer look at the errors reveals that the MISSED errors are mainly due to lexical sparsity. For example, in the following sentence:

*Vivendi earlier this week ... that it planned to **shed** its entertainment assets ...*

Our model fails to decide the "shed" as an event Transfer-Ownership as it never saw "shed" with this sense during training and rarely saw that in testing. Although using policy network can improve the generalization ability and alleviate the data sparsity issue to some extend, some instances are also necessary for policy network to guide the policy learning. The INCORRECT errors are mainly resulted from ambiguous context. For example, in the following sentence:

*... Davies is scheduled to **leave**, to become the chairman of the London.*

The word "leave" (event End-Position) can be easily misinterpreted as an event Movement due to its ambiguous context words "scheduled" and "London". The ambiguity of contexts requires better modeling of the input sentence. Other reasons of both types of event detection errors include the existence of complex language such as metaphor, idioms and sarcasm, which require lavish background knowledge and deeper reasoning.

For argument detection, we notice from Table 3 that the performance of event argument identification (60.9%) is better than that of argument role classification (56.8%). This suggests that a large number of correctly identified arguments cannot be classified properly. Among the misclassified arguments, a large portion of errors (87.3%) comes from incorrect roles (e.g., an argument with role Target is misclassified as that with role Agent) while the remaining errors (12.7%) are associated with incorrect event type (i.e., an argument role is correctly classified but is assigned to an incorrect event type). A closer look at the errors shows that the majority incorrect roles belongs to co-occurring antonymous roles, such as the argument "Hassan" (with role Target) in the following sentence:

*British officials say they believe **Hassan** was a blindfolded woman seen being **shot** in the head ...*

The model wrongly detects that "Hassan" refers to the Attacker who conducted the "shot" while "woman" was the Target of the "shot". This is mainly due to the complex context between the words "Hassan" and "shot". Similar confusable roles include Attacker/Victim, Origin/Destination, assigner/receiver and so on.

5 Conclusion

In this paper, we present a novel model *HPNet* which approaches event extraction via hierarchical policy network. In *HPNet*, the extraction process follows a two-level hierarchical structure: an event-level PN for events and an argument-level PN for participating arguments. Thanks to the hierarchical structure, *HPNet* is good at modeling deep information interactions among the subtasks, and particularly better in dealing with the multiple events issue. Comprehensive experiments demonstrate that our *HPNet* outperforms the state-of-the-art methods. As future work, this *HPNet* will be applied in cross-event or

cross-document EE.

Acknowledgements. This work was partially supported by NSFC under grants Nos. 61872446, 61902417, and 71971212, and NSF of Hunan Province under grant No. 2019JJ20024.

References

Yubo Chen, Liheng Xu, Kang Liu, Daojian Zeng, and Jun Zhao. 2015. Event extraction via dynamic multi-pooling convolutional neural networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL 2015, July 26-31, 2015, Beijing, China, Volume 1: Long Papers*, pages 167–176.

Naomi Daniel, Dragomir Radev, and Timothy Allison. 2003. Sub-event based multi-document summarization. In *the HLT-NAACL 03 on Text summarization workshop-Volume 5. Association for Computational Linguistics, 9–16*.

Ling Ding, Shengbin Jia, Xiaojun Chen, and Yang Xiang. 2018. Joint chinese event extraction via gated hybrid representation and mask-based information filtering. In *Transactions on Asian Language Information Processing*.

Jun Feng, Minlie Huang, Li Zhao, Yang Yang, and Xiaoyan Zhu. 2018. Reinforcement learning for relation classification from noisy data. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, pages 5779–5786.

Goran Glavas and Jan Snajder. 2014. Event graphs for information retrieval and multi-document summarization. *Expert Syst. Appl.*, 41(15):6904–6916.

Yu Hong, Di Lu, Dian Yu, Xiaoman Pan, Xiaobin Wang, Yadong Chen, Lifu Huang, and Heng Ji. 2015. RPI BLENDER TAC-KBP2015 system description. In *Proceedings of the 2015 Text Analysis Conference, TAC 2015, Gaithersburg, Maryland, USA, November 16-17, 2015, 2015*.

Heng Ji and Ralph Grishman. 2008. Refining event extraction through cross-document inference. In *ACL 2008, Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics, June 15-20, 2008, Columbus, Ohio, USA*, pages 254–262.

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, ACL 2013, 4-9 August 2013, Sofia, Bulgaria, Volume 1: Long Papers*, pages 73–82.

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, Brussels, Belgium, October 31 - November 4, 2018*, pages 1247–1256.

David McClosky, Mihai Surdeanu, and Christopher D. Manning. 2011. Event extraction as dependency parsing. In *The 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, Proceedings of the Conference, 19-24 June, 2011, Portland, Oregon, USA*, pages 1626–1635.

Teruko Mitamura, Zhengzhong Liu, and Eduard H. Hovy. 2015. Overview of TAC KBP 2015 event nugget track. In *Proceedings of the 2015 Text Analysis Conference, TAC 2015, Gaithersburg, Maryland, USA, November 16-17, 2015, 2015*.

Thien Huu Nguyen and Ralph Grishman. 2018. Graph convolutional networks with argument-aware pooling for event detection. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, pages 5900–5907.

Trung Minh Nguyen and Thien Huu Nguyen. 2019. One for all: Neural joint modeling of entities and events. In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019*, pages 6851–6858.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1532–1543.

Hoifung Poon and Lucy Vanderwende. 2010. Joint inference for knowledge extraction from biomedical literature. In *Human Language Technologies: Conference of the North American Chapter of the Association of Computational Linguistics, Proceedings, June 2-4, 2010, Los Angeles, California, USA*, pages 813–821.

Pengda Qin, Weiran Xu, and William Yang Wang. 2018. Robust distant supervision relation extraction via deep reinforcement learning. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 1: Long Papers*, pages 2137–2147.

Sebastian Riedel and Andrew McCallum. 2011. Fast and robust joint models for biomedical event extraction. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, EMNLP 2011, 27-31 July 2011, John McIntyre Conference Centre, Edinburgh, UK, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1–12.

Sebastian Riedel, Hong-Woo Chun, Toshihisa Takagi, and Jun'ichi Tsujii. 2009. A markov logic approach to bio-molecular event extraction. In *Proceedings of the BioNLP 2009 Workshop Companion Volume for Shared Task, BioNLP@HLT-NAACL 2009 - Shared Task, Boulder, Colorado, USA, June 5, 2009*, pages 41–49.

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 *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, pages 5916–5923.

Richard S. Sutton, David A. McAllester, Satinder P. Singh, and Yishay Mansour. 1999. Policy gradient methods for reinforcement learning with function approximation. In *Advances in Neural Information Processing Systems 12, [NIPS Conference, Denver, Colorado, USA, November 29 - December 4, 1999]*, pages 1057–1063.

Ryuichi Takanobu, Tianyang Zhang, Jiexi Liu, and Minlie Huang. 2019. A hierarchical framework for relation extraction with reinforcement learning. In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019*, pages 7072–7079.

Ronald J. Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Mach. Learn.*, 8:229–256.

Bishan Yang and Tom M. Mitchell. 2016. Joint extraction of events and entities within a document context. In *NAACL HLT 2016, The 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, San Diego California, USA, June 12-17, 2016*, pages 289–299.

Sen Yang, Dawei Feng, Linbo Qiao, Zhigang Kan, and Dongsheng Li. 2019. Exploring pre-trained language models for event extraction and generation. In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 5284–5294.

Tianyang Zhang, Minlie Huang, and Li Zhao. 2018. Learning structured representation for text classification via reinforcement learning. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, pages 6053–6060.

Junchi Zhang, Yanxia Qin, Yue Zhang, Mengchi Liu, and Donghong Ji. 2019. Extracting entities and events as a single task using a transition-based neural model. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10-16, 2019*, pages 5422–5428.

Appendix A. Training Process of *HPNet*

Algorithm 1: OVERALL TRAINING PROCEDURE

Input: the word embedding $\mathbf{w_t}$ for each word in the sentence

Calculate $\mathbf{h_t}$ for each input word embedding with Bi-LSTM;
Initiate state ${\mathbf{s_0}}^e = 0$ and time step $t = 0$;
for $i = 1$ *to* L **do**
    $t = t + 1$;
    Calculate ${\mathbf{s_t}}^e$ through Eq. 1;
    Sample o_t^e from ${\mathbf{s_t}}^e$ through Eq. 3;
    Obtain instant reward r_t^e through Eq. 4;
    if $o_t^e \in \mathcal{E}$ **then**
        for $j = 1$ *to* L **do**
            t=t+1;
            Calculate ${\mathbf{s_t}}^r$ through Eq. 7;
            Select a_t^r from ${\mathbf{s_t}}^r$ through Eq. 8;
            Obtain instant reward r_t^r through Eq. 10;
        Obtain the argument-level terminal reward r_{ter}^r through Eq. 12;
Obtain the event-level terminal reward r_{ter}^e through Eq. 6;
Optimize the HPNet with Eq. 15;