Exploiting the Ground-Truth: An Adversarial Imitation Based Knowledge Distillation Approach for Event Detection

Jian Liu,[1,2] **Yubo Chen,**[1] **Kang Liu**[1,2]
[1]National Laboratory of Pattern Recognition, Institute of Automation
Chinese Academy of Sciences, Beijing, 100190, China
[2]University of Chinese Academy of Sciences
{jian.liu, yubo.chen, kliu}@nlpr.ia.ac.cn

Abstract

The ambiguity in language expressions poses a great challenge for event detection. To disambiguate event types, current approaches rely on external NLP toolkits to build knowledge representations. Unfortunately, these approaches work in a pipeline paradigm and suffer from error propagation problem. In this paper, we propose an adversarial imitation based knowledge distillation approach, for the first time, to tackle the challenge of acquiring knowledge from raw-sentences for event detection. In our approach, a teacher module is first devised to learn the knowledge representations from the ground-truth annotations. Then, we set up a student module that only takes the raw-sentences as the input. The student module is taught to imitate the behavior of the teacher under the guidance of an adversarial discriminator. By this way, the process of knowledge distillation from raw-sentence has been implicitly integrated into the feature encoding stage of the student module. To the end, the enhanced student is used for event detection, which processes raw texts and requires no extra toolkits, naturally eliminating the error propagation problem faced by pipeline approaches. We conduct extensive experiments on the ACE 2005 datasets, and the experimental results justify the effectiveness of our approach.

Introduction

With the rapid development in the fields of Text Mining (TM) and Natural Language Processing (NLP), the study on Event Detection (ED) has gained great popularity (Aggarwal and Zhai 2012; Hirschberg and Manning 2015). Concretely, ED is a specialized Information Extraction (IE) technology which aims to identify event instances of specified types in unstructured texts, and it has shown to be beneficial for a big variety of real-world applications including Information Retrieval (IR) (Allan 2002; Campos et al. 2014), Question Answering (QA) (Saurí et al. 2005; Kuchmann-Beauger, Aufaure, and Thollot 2015), Automatic Text Summarization (Ge et al. 2016; Marujo et al. 2017) and others.

Amongst, a major challenge faced by ED is the ambiguity in natural language expressions (Poon and Vanderwende 2010; Ritter et al. 2012; Li, Ji, and Huang 2013). On the one hand, the same event can be expressed in a wide variation; on the other hand, depending on the context, the same expression might refer to entirely different events. To illustrate, consider the following two sentences (adapted from the corpus of ACE 2005[1]):

$S1$: <u>*The European Unit*</u> is set to **release** <u>*20 million*</u> euros to Iraq.

$S2$: The government reports that <u>*Anwar*</u> 's earliest **release** date is <u>*April 14*</u>.

It is note that each of the sentences contains a token **release**, however, according to the ACE Annotation Guidelines[2], **release** expresses completely different events in the two sentences — in $S1$, **release** evokes a *Transfer-Money* event; while in $S2$, **release** expresses a *Release-Parole* event. According to (Liu et al. 2018), 57% of the event triggers are ambiguous and trigger different events (in the ACE 2005 corpus). This ambiguity might confuse many event detection systems and lead to a considerable amount of errors.

Previous studies have shown that the chunk knowledge corresponding to the sentences can provide evidence for event type disambiguation (Hong et al. 2011; Li, Ji, and Huang 2013; Feng et al. 2016; Liu et al. 2017). Taking the above two examples again, if an event detector is aware that in $S1$: 1) *The European Unit* is an *ORGANIZATION* entity, and 2) *20 million* designates a *NUMBER* value, then the detector can guess that $S1$ might express a financial event which is related to an organization. These clues help to recognize the *Transfer-money* event without ambiguity. As for $S2$, if an event detector knows that: 1) *Anwar* means a *PERSON* entity, and 2) *April 14* indicates a *DATE* value, then the detector can correctly identify the *Release-Parole* event by considering the prior knowledge that — if a sentence matches the pattern "*[PERSON]* **release** *[DATE]*", then the token **release** is highly probable to express a *Release-Parole* event in the sentence.

Unfortunately, how to acquire this chunk knowledge from the raw-sentences for ED remains a challenge, especially in the real test scenario where the ground-truth annotations are missing. Current works usually first use external NLP toolkits (e.g., a named entity recognizer) to predict chunk labels and then transform the predicted labels into vectorized knowledge representations. These methods operate in

[1]https://catalog.ldc.upenn.edu/LDC2006T06

[2]https://www.ldc.upenn.edu/sites/www.ldc.upenn.edu/files/english-events-guidelines-v5.4.3.pdf

a pipeline paradigm, and consequently, the errors derived from the NLP toolkits could propagate and harm their final performance. In our knowledge, there is no current literature explicitly studies this practice problem in the context of event detection.

In this paper, to acquire knowledge from raw-sentences for ED, we propose an adversarial based knowledge distillation approach. In the training stage, we first learn the knowledge representations on the ground-truth annotations, and then we incorporate the knowledge distillation process directly into the feature encoding procedure, through an adversarial imitation strategy. Consequently, in the testing stage, our model processes raw-sentences and requires no external NLP toolkits any more, which naturally eliminates the error propagation problem faced by pipeline approaches.

Methodologically, in our approach, we first build a teacher module, which cat take full advantages of ground-truth annotations to learn knowledge representations. Note that we cannot directly use the teacher module for testing since usually the golden annotations are missing in the real test scenario. Then, we set up a student module, which only takes raw texts as the input and will be used for testing in the future time. The plain "student" performs relatively poorly since it cannot utilize any chunk knowledge. The idea behind our approach is to enhance the "student" by forcing it to imitate the behavior of the teacher module, by proposing an adversarial imitation strategy. To achieve this goal, we employ a discriminator, which measures the similarity between the "student" and the "teacher". During training, the discriminator manages to discriminate between the "student" and the "teach" by examining their outputs, meanwhile, the "student" tries to "fool" the discriminator by producing vectorized outputs that appear to come from the "teacher" (i.e., to imitate the "teacher"). The contest between the discriminator and the "student" composes a dynamic adversarial game, which helps the two modules to improve each other. We assume at a time point, a skilled discriminator still cannot identify the "student" out, then using the current "student" for testing can yield rival performance as using the teacher module. Since the student module only processes raw-sentences and is not aware of any explicit forms of chunk knowledge, we say the process of knowledge distillation from raw-sentence has been implicitly integrated into its feature encoding stage. To the end, our final model requires no external NLP toolkits in the testing phase and avoids the error propagation problem faced by current pipeline approaches naturally.

Our contributions are summarized as follows:

1) In this paper, for the first time, we design a new adversarial imitation based knowledge distillation approach, to tackle the challenge of acquiring knowledge from raw-sentences for event detection task.

2) Our approach learns to incorporate the knowledge distillation process into feature encoding stage in a implicit manner, which requires no external NLP toolkits for testing and avoids the error propagation problem faced by current pipeline approaches naturally.

3) The extensive experiments on the ACE 2005 datasets justify the effectiveness of our approach.

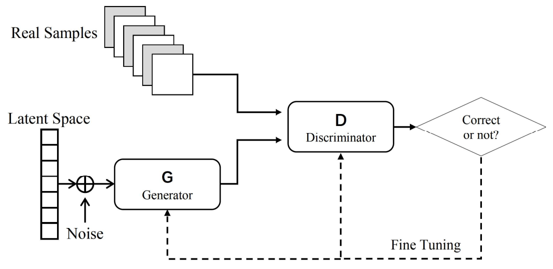

Figure 1: The illustration of Generative Adversarial Networks (GANs) and adversarial training.

Preliminaries

Task Description

The evaluation of event detection is defined in the Automatic Content Extraction (ACE) program[3]. We next introduce the definitions of several ACE terminologies to facilitate the understanding of the task.

- **Entity**: an object in one of the interested semantic categories (e.g., PERSON, ORGANIZATION).
- **Entity mention**: a reference to an **Entity**, which is typically a noun phrase.
- **Event mention**: a phrase or sentence within which a certain event is described. An event mention contains an **Event trigger** and some **Event argument**s if any.
- **Event trigger**: the word that most clearly expresses the event mention, which is most often a single verb or noun.
- **Event argument**: an entity mention, temporal expression or value (e.g., Job-Title) that serves as a participant with a specific role in the event mention.

For a sentence: "*The boy* **died** in the *hospital*", the overall Event Extraction (EE) evaluation defined in ACE program requires to extract a *Die* event along with the event trigger **died** and the two event arguments: *The boy* (Role=*Victim*) and *the hospital* (Role=*Place*). Unlike the overall EE evaluation, the ED evaluation only cares about event type and event trigger. That is, for the above sentence, ED requires to locate the event trigger **died** and identify the event type *Die*.

Adversarial Training

The most influential idea of adversarial training is the Generative Adversarial Networks (GANs) (Goodfellow et al. 2014). As shown in Figure 2, GANs are usually implemented as a hybrid system that consists of two neural networks contesting each other: one generative network (referred to as G) generates candidates and the other discriminative network (referred to as D) evaluates them. During training, G aims to increase the error rate of D, i.e., to "fool" D by producing novel synthesized instances that appear to come from the true data distribution, while D manages to discriminate the synthesized data out. Back-propagation is applied to both networks for optimization, and in the end,

[3]http://projects.ldc.upenn.edu/ace/

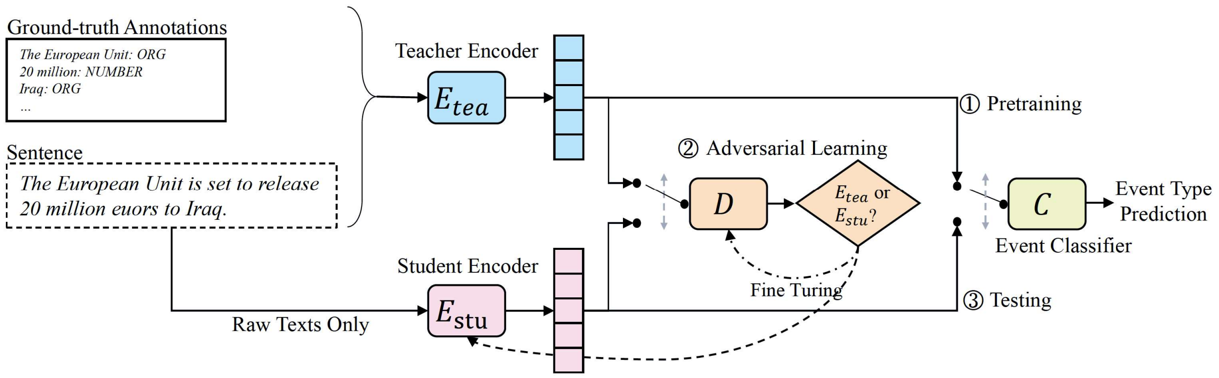

Figure 2: The overall architecture of our approach. The model consists of four main components: the teacher encoder E_{tea}, the student encoder E_{stu}, the discriminator D and the event classifier C. During training, we first concatenate E_{tea} and C to learn discriminative features for event detection. Next, D and E_{stu} contest with each other under the adversarial imitation strategy. In the final testing stage, we concatenate the enhanced E_{stu} with C to construct the final event detector, which only takes the raw texts as the input and requires no additional NLP toolkits.

G can produce instances that close to the true data, while D becomes more skilled at flagging synthetic data.

GANs have been applied to many different domains such as image generation (Goodfellow et al. 2014), representative learning (Radford, Metz, and Chintala 2015) and others. In the field of NLP, the GAN-style adversarial learning has been applied to many tasks including text classification (Liu, Qiu, and Huang 2017), Chinese word segmentation (Chen et al. 2017), neural response generation (Li et al. 2017) and others. In particular, our method is highly motivated by the work of (Qin et al. 2017) using adversarial mechanism to exploit the annotated connectives for implicit discourse relation extraction. In the context of event detection, we design an adversarial imitation approach to acquire knowledge from raw-sentences for event type disambiguation, which eliminates the error propagation problem faced by pipeline approaches naturally.

Methodology

Figure 2 presents the overall architecture of our approach. In the following, we will introduce the details of our model as well as the training strategy.

Model Component

Our model consists of four main components: the teacher encoder, the student encoder, the adversarial discriminator and the event classifier. Specifically,

- **The Teacher Encoder** takes the full advantage of the ground-truth chunk annotations and encodes them into vectorized knowledge representations for feature learning. During training, the teacher encoder serves as the role of "teacher" to teach a student module to learn. We denote the teacher encoder as E_{tea}.
- **The Student Encoder** takes only raw texts to build representations. In the training stage, the student encoder is forced to produce vectorized outputs that are similar to the outputs of E_{tea}, under the guidance of an adversarial discriminator. We denote the student encoder as E_{std}.
- **The Adversarial Discriminator** aims to distinguish between E_{tea} and E_{stu}. If a skilled adversarial discriminator still cannot tell the difference between E_{stu} and E_{tea}, then using E_{stu} for feature learning will yield similar performance as using E_{tea}. We denote the adversarial discriminator as D.
- **The Event Classifier** takes the vectorized outputs (features) produced by either E_{tea} or E_{stu} as the input to make the final event type classification. We denote the event classifier as C.

Theoretically, each component can be instantiated with any learning structure in the machine learning literature. We present our particular implementations in the following.

Implementation Structures

The implementation structure of each component is presented in this subsection.

Attention Based Encoder E_{tea} and E_{stu} are implemented with (self-)attention based neural architectures (Bahdanau, Cho, and Bengio 2014; Vaswani et al. 2017). The structure of E_{tea} is presented in Figure 3.

A N-token sentence is denoted as $s = \{w_1, ..., w_t, ..., w_N\}$, where we use w_t to designate the concerning token that we want to check whether it is an event trigger and which event type it evokes. To encode the sentence, we first give each token in s a real-valued vector as its word embedding (Mikolov et al. 2013; Pennington, Socher, and Manning 2014), and then:

1) For E_{tea}, we further incorporate the manually annotated entity/event-argument labels into the representation. For each label, we give it a real-valued vector as its embedding. We concatenate the entity/argument embeddings with the word embedding as the representation of each token.

2) For E_{stu}, we take only the word embedding as the representation of each token.

We next employ bidirectional GRU (BiGRU) (Chung et al. 2014) to encode the entire sentence. After encoding, s is

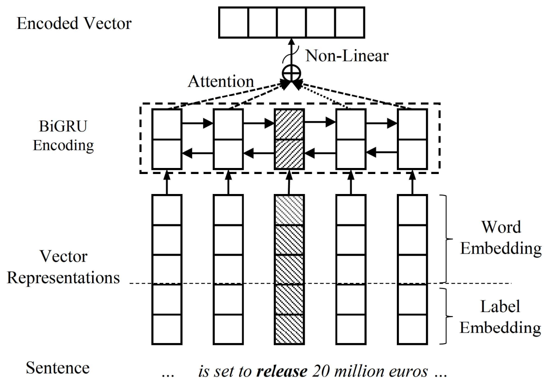

Figure 3: The architecture of E_{tea}. In the sentence, ***release*** is the concerning token for assigning an event type.

transferred to a state sequence $\{h_1, ..., h_N\}$, where h_i is the state representation of each token w_i. We adopt attention-based composition strategy to construct the context representation $ctx^{(w_t)}$ with respect to the concerning token w_t:

$$ctx^{(w_t)} = \sum\nolimits_{i=1}^{N} \alpha_i^{(w_t)} * h_i \tag{1}$$

$$\alpha_i^{(w_t)} = \frac{exp(m_i^{(w_t)})}{\sum_{j=1}^{N} exp(m_j^{(w_t)})} \tag{2}$$

where $\alpha_i^{(w_t)}$ means the attention weight of h_i with respect to w_t; j ranges over the number of tokens in s; $m_i^{(w_t)}$ indicates the semantic correlation between w_t and w_i, which is computed by:

$$m_i^{(w_t)} = tanh(W_{att} \cdot (h_i \oplus h_t) + b_{att}) \tag{3}$$

where $\oplus$ stands for the concatenation operation; W_{att} and b_{att} are attention parameters; h_i and h_t are state representations of w_i and w_t respectively. In the end, we concatenate $ctx^{(w_t)}$ and h_t as the input of a non-linear transformation layer to construct the final representation of the concerning token w_t. Note the non-linear transformation makes E_{tea} and E_{stu} have the same output dimension. The outputs of E_{tea} and E_{stu} are denoted as $f_{tea}^{(w_k)}$ and $f_{stu}^{(w_k)}$ respectively.

Binary Classification-Based Discriminator D is implemented as a binary classifier that equips feed-forward neural networks with a sigmoid output layer. It takes a vectorized feature $f^{(w_t)}$ (either $f_{tea}^{(w_t)}$ or $f_{stu}^{(w_t)}$) as the input and yields a probability p that indicates the probability that D thinks $f^{(w_t)}$ comes from E_{tea}. The probability is computed by:

$$p = D(f^{(w_t)}) = \sigma(W_h(tanh(W_x f^{(w_t)} + b_x)) + b_h) \tag{4}$$

where σ is the *multivariate sigmoid function* that maps a real-valued vector to a float number between 0 and 1; W_h, W_x, b_x and b_h are model parameters. Ideally, an perfect discriminator would always output 1s for outputs of E_{tea} and 0s for outputs of E_{stu}.

Multi-class Event Classifier The event classifier C is implemented as a multi-class classifier. It also accepts a a vectorized feature $f^{(w_t)}$ (either $f_{tea}^{(w_t)}$ or $f_{stu}^{(w_t)}$) as the input, and it computes a vector out that indicates the prediction probabilities of different event types for the concerning token w_t. The prediction probability for the lth event type, $P(l|f^{(w_t)}, \Theta)$, is computed as:

$$out = softmax(W_o \cdot f^{(w_t)} + b_o)) \tag{5}$$

$$P(l|f, \Theta) = out_{(l)} \tag{6}$$

where W_o and b_o are model parameters of C; Θ indicates the overall parameters; $out_{(l)}$ indicates the lth element of out.

The Adversarial Imitation Strategy

This subsection illustrates the adversarial imitation strategy to train our model. The training process contains one pretraining stage and one adversarial learning stage.

In the Pretraining Stage:

(1) We concatenate E_{tea} and C to form an event detector that is aware of ground-truth annotations. The loss function is defined as:

$$J_{tea}(\Theta_U) = -\sum\nolimits_{k=1}^{K} P(y^{(w_k)}|f_{tea}^{(w_k)}, \Theta_U) \tag{7}$$

where Θ_U=$\{\Theta_{tea}, \Theta_C\}$ means the union set of parameters of E_{tea} and C; k ranges over the number of tokens in the train set; w_k indicates the kth token and $f_{tea}^{(w_k)}$ is the output of E_{tea} for w_k; $y^{(w_k)}$ means the ground-truth label of w_k. This substage jointly trains E_{tea} and C.

(2) We next freeze the event classifier C, and we concatenate E_{stu} and C to build a raw-sentences event detector. The loss function of this substage is defined as:

$$J_{stu}(\Theta_{stu}) = -\sum\nolimits_{k=1}^{K} P(y^{(w_k)}|f_{stu}^{(w_k)}, \Theta_{stu}) \tag{8}$$

where Θ_{stu} indicates only the parameters of E_{stu}; $f_{stu}^{(w_k)}$ means the vectorized feature of w_k computed by E_{stu}. Note this substage only trains E_{stu} and does not update C.

(3) We next freeze both E_{tea} and E_{stu}, and we treat the outputs of E_{tea} as positive examples (labeled as 1s) and the outputs of E_{stu} as negative examples (labeled as 0s) to pretrain D. Cross-entropy loss is adopted:

$$\begin{aligned} J_D(\Theta_D) = -\sum\nolimits_{k=1}^{K} [log(D(f_{tea}^{(w_k)}))+ \\ log(1 - D(f_{stu}^{(w_k)}))] \end{aligned} \tag{9}$$

where Θ_D denotes the parameters of D.

In the end of the pretraining stage, a leave-out test shows that D achieves an accuracy of 91.7%, which means D has a 91.7 percent chance to correctly distinguish E_{stu} and E_{tea}.

In the Adversarial learning Stage:

In this substage, E_{stu} and D contest with each other to formulate an adversarial game, where E_{stu} tries to "fool" D by generating outputs that resemble the outputs of E_{tea}, meanwhile, D manages to discriminate E_{stu} out.

In this substage, the loss function of D is still $J_D(\Theta_D)$ (defined in Eq.(9)). However, the loss function with respect to E_{stu} should include two aspects: 1) The first part

Algorithm 1 Adaptive Learning Process

```
Input: Training data $\{(w_k, y^{(w_k)})_{k=1,...,K}\}$; Pretrained modules $E_{tea}$, $E_{stu}$, $D$ and $C$.
1: Keep $E_{tea}$ and $C$ fixed.
2: repeat
3:     Update $E_{stu}$ using Eq.(11)
4:     if the accuracy of $D$ is lower than a threshold then
           Update $D$ using Eq.(9)
5:     end if
6: until convergence
Output: The adversarially enhanced $E_{stu}$.
```

corresponds to the final classification error, which is still $J_{stu}(\Theta_{stu})$ (in Eq.(8)); 2) The second part corresponds to whether E_{stu} has successfully fooled D, and the loss corresponds to this part is defined as:

$$J_{adv}(\Theta_{stu}) = \sum_{k=1}^{K} log(1 - D(f_{stu}^{(w_k)})) \\ = -\sum_{k=1}^{K} log(D(f_{stu}^{(w_k)})) \tag{10}$$

$J_{adv}(\Theta_{stu})$ gets the minimum value when D just cannot discern whether the input vector comes from E_{stu} or E_{stu}.

The overall loss function of E_{stu} in this stage is thus:

$$J_{stu\_adv}(\Theta_{stu}) = J_{stu}(\Theta_{stu}) + \lambda * J_{adv}(\Theta_{stu}) \tag{11}$$

where λ is a hyper parameter balancing J_{stu} and J_{adv}, whose value is decided by a grid search on the dev set.

In practice, we note that the alternative updating strategy proposed by (Goodfellow et al. 2014) in fact leads to an uneven game in our approach: D tends to be too powerful, and the accuracy of D always keeps over 85%, i.e., the E_{stu} has difficulties in cheating D and the imitating cannot proceed. We come up two heuristic rules to smooth the learning process: (1) We set the learning rate of E_{stu} as twice as the learning rate of D (to strength E_{stu}). (2) We only update the parameters of D when we notice its accuracy is lower than a threshold (0.7 in our approach, to weaken D). **Algorithm 1** summarizes the modified updating process.

After the training process converges, we concatenate the enhanced E_{stu} with the pre-trained event classifier C as the final event detector, which only takes plain texts as the input to identify events. For optimization, we adopt mini-batches updating with Adam rules (Kingma and Ba 2014). Regularization is achieved by dropout (Srivastava et al. 2014) and L_2 norm penalty.

Experiments

Dataset

We take the ACE 2005 corpus for experimental evaluations. It is a multilingual dataset on which defines 33 types of events. Following the previous studies (Li, Ji, and Huang 2013; Chen et al. 2015; Nguyen and Grishman 2015), we add an extra *NONE* type to designate the non-trigger token, so the Event Detection task is formulated as a 34-class classification problem.

Model	**P**	**R**	**F_1**
CrossEntity (Hong et al.)	72.9	64.3	68.3
CNNED (Nguyen and Grishman)	71.8	66.4	69.0
DLRNN (Duan, He, and Zhao)	77.2	64.9	70.5
ArgATT (Liu et al.)	**78.0**	66.3	71.7
Teacher + emb	71.9	66.0	68.8
Teacher + emb + ety	71.6	69.1	70.3
Teacher + emb + agt	76.3	72.4	74.2
Teacher + emb + ety + agt	76.8	**72.9**	**74.8**

Table 1: Experimental results on the ACE 2005 English set. Bold indicates the best performance with respective to each evaluation metric.

For the English corpus, we use 520 particular documents for training, 30 particular documents for developing and the remaining 40 documents for testing, same as previous studies for comparison. For the Chinese corpus, we adopt the experimental configuration as (Chen and Ji 2009), in which 569/64/64 documents are used as the train/dev/test sets. We use the pretrained 300-dimension Glove vectors as English word embeddings, and we train Chinese embeddings on NYT corpus. Other hyperparameters are tuned on the dev set via a *grid search* strategy. Particularly, the optimized balance parameter λ is set to 0.05. We adopt the Precision (**P**), Recall (**R**) and F1 score (**F_1**) as evaluation metrics.

We have conducted experiments to verify that: 1) Our teacher module can effectively leverage the ground-truth annotations for knowledge learning. 2) By adopting the adversarial imitation strategy, the student module can achieve good performance in the real test scenario where the golden annotations are missing.

Performance on Gold-truth Annotations

To investigate the ability of the teacher module to leverage ground-truth annotations for knowledge representation learning, several state-of-the-art methods are compared:

- *CrossEntity* is the model proposed in (Hong et al. 2011), which studied the entity co-occurrence patterns for ED.
- *CNNED* is the model proposed in (Nguyen and Grishman 2015), which employed Convolution Neural Networks (CNNs) to learn features for ED.
- *DLRNN* is the model proposed in (Duan, He, and Zhao 2017), which incorporated document level clues for ED.
- *ArgAtt* is the model proposed in (Liu et al. 2017), which payed attention to event arguments for ED.

Our teacher module is denoted as *Teacher*, and we use *+emb*, *+ety* and *+agt* to designate the combination with word, entity and event-argument embeddings respectively. Table 1 summarizes the comparison.

From the results, when incorporated with all the ground-truth annotations, the teacher module can outperform other methods by a large margin (4.9% on the average F1 score), which justifies the effectiveness of the teacher module for knowledge representation learning. Additionally, it seems that the annotations of event arguments provide the most

Setting	Model	P	R	$\mathbf{F_1}$
Golden	*CNNED*[‡]	71.8	66.4	69.0
	ArgATT[‡]	78.0	66.3	71.7
	Teacher	76.8	72.9	74.8
Predicted	*CNNED*[‡]	71.9	63.8	67.6
	ArgATT	**76.1**	66.0	70.7
	Teacher	72.4	68.9	70.6
Adv	*Student-Final*	73.4	**69.1**	**71.2**

Table 2: Experimental results on ACE 2005 English corpus. *Golden/Predicted* means resorting to golden/predicted annotations. ‡ indicates taken from the original paper. Bold indicates the best performance.

Setting	Model	P	R	$\mathbf{F_1}$
Golden	*CNNED*	61.3	58.5	59.9
	ArgATT	62.5	58.1	60.2
	Teacher	63.0	59.3	61.9
Predicted	*CNNED*	**60.3**	55.5	57.8
	ArgATT	58.1	57.1	57.6
	Teacher	59.7	55.1	57.3
Adv	*Student-Final*	58.8	**58.2**	**58.5**

Table 3: Experimental results on ACE 2005 Chinese corpus. *Golden/Predicted* means resorting to golden/predicted annotations. Bold indicates the best performance.

contributions, which leads to a improvement of 5.4% on F1 score compared with basic strategy (*Teacher+emb+atg* v.s. *Teacher+emb*). This is sensible, for example, "fire" is usually an ambiguous event trigger, which can evoke both of *Attack* and *End-Position* events. However, if we know an event argument with the role *Instrument* occurs, we can easily assign the *Attack* event to the "fired" without ambiguity. Unfortunately, the event argument information can not be accessible in the testing stage.

Since the student module is designed to imitate the teacher module, theoretically the performance of the "student" cannot exceed the performance of the teacher module — The performance of the teacher module in fact manifest an upper-bound of our final approach.

Performance in the Real Testing Scenario

We next investigate the performance of our model in the real testing scenario, where the golden annotations are missing. Our model is denoted as *Student-Final*. We select *CNNED* and *ArgAtt* as the baseline models for comparison and we also investigate the performance of the "teacher" on predicted tags. To obtain the predicted labels, we train Bi-LSTM-CRF taggers of entity and event-argument on the train set. To the end, the F_1 scores on English entities and event arguments are 83.4% and 65.7% respectively, which matches the state-of-the-arts (Yang and Mitchell 2017; Judea and Strube 2017). The same taggers are applied to Chinese dataset, resulting in 73.5% and 52.3% on F_1 scores.

Table 2 and Table 3 summarize the performance on the ACE corpus. From the results, 1) Compared with using golden annotations for testing (denoted by "Golden" in the Table 2, 3), using the predicted tags leads to a significant drop in performance (on the average -2.3% for the English ED and -3.1% for the Chinese ED), which manifests the error propagation problem faced by pipeline approaches. 2) Our enhanced student modules beats the other approaches and achieves the best performance (71.2% on F_1 for English ED and 58.5% on F_1 for Chinese ED), which justifies the effectiveness of the adversarial imitation strategy.

Model	P	R	$\mathbf{F_1}$
Tea (CNN) + Stu (CNN)	73.0	68.7	70.8
Tea (ATT) + Stu (CNN)	68.5	66.9	67.7
Tea (ATT) + Stu (IFR)	69.5	67.7	68.6
Student-Final	**73.4**	**69.1**	**71.2**

Table 4: Results on the ACE 2005 English corpus to investigate the influence of different module architectures.

Model	P	R	$\mathbf{F_1}$
Discriminator (none)	69.6	68.8	69.2
Discriminator (mse)	69.5	67.7	68.6
Discriminator (fixed)	70.5	68.9	69.7
Student-Final	**73.4**	**69.1**	**71.2**

Table 5: Results on the ACE 2005 English corpus to investigate the effect of different discriminative strategies.

Ablation Study

The Effect of Module Architectures We study the effects of different module implementations in this subsection. Several variation systems are proposed, including: 1) A system that employs CNN structures in both the teacher and student modules (*Tea (CNN) + Stu (CNN)*). 2) A system that only substitutes the student module with a CNN architecture (*Tea (ATT)+Stu (CNN)*). 3) A system that uses a inferior architecture that excludes the attention layer in the "student" (*Tea (ATT) + Stu (IFR)*).

Table 4 summarizes the performance of each system. From the results, 1) When both of the "teacher" and "student" are implemented with CNNs, the performance is also satisfied (70.8% on F1). 2) When the teacher and student modules have a heterogeneous implementations (e.g., the second system), the performance drops significantly (−3.1% on F1). We guess the heterogeneous implementations can cause the output spaces of the "teacher" and "student" rather different, which fails the imitation strategy. 3) The third system with a inferior architecture in the "student" also underperforms our full approach. The reason might be that, to make the adversarial imitation strategy work, it usually calls for a sophisticated structure that can capture complex inter-dependency among context words, such as the attentive architecture adopted in our full approach.

The Effect of the Discriminative Strategy We compare our full approach to several variation systems to study the effects of the discriminative strategies. 1) The first system merely ignores the discriminator. We still concatenate the

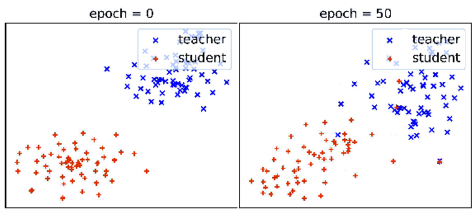

Figure 4: The comparison of the output spaces of the "teacher" and the "student". Visualization is conducted by T-SNE (van der Maaten and Hinton 2008).

student module and the pretrained event classifier for fine-grained tuning, to make it the same as others (denoted as *Discriminator (none)*). 2) The second system adopts the Mean Square Error (MSE) to measure the similarity between the "teacher" and the "student", and the MSE loss also gets backpropagated (denoted as *Discriminator (mse)*). 3) The third system employs the same discriminator as ours, but the discriminator does not continue to update after the pretraining stage (denoted as *Discriminator (fixed)*).

Table 5 summaries the performance of each model. From the results, we observe that: 1) Our full approach outperforms all of the variation systems, which justifies the effectiveness of the adversarial imitation based discriminative strategy. 2) Static strategies (i.e., the second/third system) behave poorly. Particular, compared with *Discriminator (none)*, MSE strategy even yields an adverse effect (-0.6% on F1). For *Discriminator (fixed)* model, after several updates, the accuracy of its discriminator reaches 0.5, i.e., the "student" has learned to "fool" the discriminator in some particular ways. However, since the discriminator is fixed, it cannot get further improved to discriminate the "student" out, making this approach underperform our full model, though it is still better than *Discriminator (none)*. 3) Comparing *Discriminator (none)* with *Teacher+emb* in Table 1, we observe that even the pretrained event classifier could provide some discriminability to bring a slight improvements on the performance ($+0.4\%$ on F1 score).

Learning Visualization

Figure 4 visualizes the output spaces of E_{stu} and E_{tea} at the different learning epochs to understand the learning process. As shown, in the beginning, it exists a distinct boundary between the output spaces of E_{stu} and E_{tea} (left). While, as the learning proceeds, it reveals some overlaps between the output spaces (right). Additionally, we examine the discriminator and find its accuracy fluctuates between 0.55 and 0.70 during the adversarial learning stage. A plausible explanation is that, by adopting the adversarial imitation strategy, E_{stu} is forced to approach to E_{tea} (the overlapping in output spaces), which brings difficulties for the discriminator to discern them (the drop in accuracy).

Related Works

The related words include: 1) For Event Detection, various methods have been proposed. Feature-based approaches include (Ahn 2006) that examined lexical and syntactic features; (Ji and Grishman 2008; Hong et al. 2011; Li, Ji, and Huang 2013) that examined both local and global features and others. Representation-based approaches include (Chen et al. 2015; Nguyen and Grishman 2015) that used Convolution Neural Networks to do automatic feature engineering; (Feng et al. 2016) that leveraged hybrid networks for feature learning; (Nguyen, Cho, and Grishman 2016) that modeled non-continue skip-grams; (Duan, He, and Zhao 2017) that investigated document embedding for ED; (Liu et al. 2017) that used supervised attention mechanism to exploit event arguments information for ED; (Sha et al. 2018; Nguyen and Grishman 2018) that incorporated vectorized syntactic knowledge and others. Nevertheless, as illustrated, to acquire chunk knowledge from raw-sentences, most of these approaches relied on external NLP toolkits and suffered from error propagation problem. 2) For Adversarial Learning, researchers have studied the idea in various aspects such as generative model (Goodfellow et al. 2014), domain adaptation (Ganin et al.) and others. In the field of NLP, it has been applied to many tasks including text classification (Liu, Qiu, and Huang 2017), Chinese word segmentation (Chen et al. 2017), neural response generation (Li et al. 2017), implicit discourse relation extraction (Qin et al. 2017). Knowledge distillation method include (Hinton, Vinyals, and Dean 2015), which learns from the predicted labels. This paper provides an easy GAN-style knowledge distillation approach for the event detection task.

Conclusions and Future Work

In conclusion, we propose a new adversarial imitation based knowledge distillation approach, to acquire knowledge from raw-sentences for event detection task. Our model learns knowledge representation from ground-truth annotations, and incorporates the knowledge distillation process into feature encoding stage in a implicit manner. We conduct extensive experiments on the ACE 2005 datasets, and the experimental results justify the effectiveness of our approach. In the future, we plan to adapt our model to other NLP tasks where the knowledge acquisition from raw-sentences is also critical, such as statistical parsing, relation extraction and others.

Acknowledgments

This research work was supported by the Natural Science Foundation of China (No.61533018 and No.61806201) and the independent research project of National Laboratory of Pattern Recognition. This work was also supported by a grant from Ant Financial Services Group. We would like to thank the anonymous reviewers for their valuable feedback.

References

Aggarwal, C. C., and Zhai, C. 2012. *Mining text data*. Springer Science & Business Media.

Ahn, D. 2006. The stages of event extraction. In *Workshop on Annotating and Reasoning About Time and Events*.

Allan, J. 2002. *Topic Detection and Tracking: Event-Based Information Organization*.

Bahdanau, D.; Cho, K.; and Bengio, Y. 2014. Neural machine translation by jointly learning to align and translate. *CoRR*.

Campos, R.; Dias, G.; Jorge, A. M.; and Jatowt, A. 2014. Survey of temporal information retrieval and related applications. *ACM Comput. Surv.*

Chen, Z., and Ji, H. 2009. Language specific issue and feature exploration in chinese event extraction. In *NAACL*.

Chen, Y.; Xu, L.; Liu, K.; Zeng, D.; and Zhao, J. 2015. Event extraction via dynamic multi-pooling convolutional neural networks. In *Proceedings of the 53rd ACL and the 7th IJCNLP*.

Chen, X.; Shi, Z.; Qiu, X.; and Huang, X. 2017. Adversarial multi-criteria learning for chinese word segmentation. In *ACL*.

Chung, J.; Gulcehre, C.; Cho, K.; and Bengio, Y. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. In *NIPS 2014 Workshop on Deep Learning*.

Duan, S.; He, R.; and Zhao, W. 2017. Exploiting document level information to improve event detection via recurrent neural networks. In *IJCNLP*.

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

Ganin, Y.; Ustinova, E.; Ajakan, H.; Germain, P.; Larochelle, H.; Laviolette, F.; Marchand, M.; and Lempitsky, V. Domain-adversarial training of neural networks. *J. Mach. Learn. Res.*

Ge, T.; Cui, L.; Chang, B.; Li, S.; Zhou, M.; and Sui, Z. 2016. News stream summarization using burst information networks. In *EMNLP*.

Goodfellow, I.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A.; and Bengio, Y. 2014. Generative adversarial nets. In *NIPS*.

Hinton, G.; Vinyals, O.; and Dean, J. 2015. Distilling the knowledge in a neural network. In *NIPS Workshop*.

Hirschberg, J., and Manning, C. D. 2015. Advances in natural language processing. *Science*.

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

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

Judea, A., and Strube, M. 2017. Event argument identification on dependency graphs with bidirectional lstms. In *IJCNLP*.

Kingma, D. P., and Ba, J. 2014. Adam: A method for stochastic optimization. *CoRR*.

Kuchmann-Beauger, N.; Aufaure, M.-A.; and Thollot, R. 2015. Context-aware question answering system. US Patent 8,935,277.

Li, J.; Monroe, W.; Shi, T.; Jean, S.; Ritter, A.; and Jurafsky, D. 2017. Adversarial learning for neural dialogue generation. In *EMNLP*.

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

Liu, S.; Chen, Y.; Liu, K.; and Zhao, J. 2017. Exploiting argument information to improve event detection via supervised attention mechanisms. In *ACL*.

Liu, J.; Chen, Y.; Liu, K.; and Zhao, J. 2018. Event detection via gated multilingual attention mechanism.

Liu, P.; Qiu, X.; and Huang, X. 2017. Adversarial multi-task learning for text classification. In *ACL*.

Marujo, L.; Ribeiro, R.; Gershman, A.; de Matos, D. M.; Neto, J. P.; and Carbonell, J. 2017. Event-based summarization using a centrality-as-relevance model. *Knowledge and Information Systems*.

Mikolov, T.; Sutskever, I.; Chen, K.; Corrado, G.; and Dean, J. 2013. Distributed representations of words and phrases and their compositionality. In *NIPS*.

Nguyen, T. H., and Grishman, R. 2015. Event detection and domain adaptation with convolutional neural networks. In *Proceedings of the 53rd ACL and the 7th IJCNLP*.

Nguyen, T., and Grishman, R. 2018. Graph convolutional networks with argument-aware pooling for event detection.

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

Pennington, J.; Socher, R.; and Manning, C. 2014. Glove: Global vectors for word representation. In *EMNLP*.

Poon, H., and Vanderwende, L. 2010. Joint inference for knowledge extraction from biomedical literature. In *NAACL*.

Qin, L.; Zhang, Z.; Zhao, H.; Hu, Z.; and Xing, E. 2017. Adversarial connective-exploiting networks for implicit discourse relation classification. In *ACL*.

Radford, A.; Metz, L.; and Chintala, S. 2015. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*.

Ritter, A.; Mausam; Etzioni, O.; and Clark, S. 2012. Open domain event extraction from twitter. In *SIGKDD*.

Saurí, R.; Knippen, R.; Verhagen, M.; and Pustejovsky, J. 2005. Evita: a robust event recognizer for qa systems. In *EMNLP*.

Sha, L.; Qian, F.; Chang, B.; and Sui, Z. 2018. Jointly extracting event triggers and arguments by dependency-bridge rnn and tensor-based argument interaction.

Srivastava, N.; Hinton, G.; Krizhevsky, A.; Sutskever, I.; and Salakhutdinov, R. 2014. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*.

van der Maaten, L., and Hinton, G. 2008. Visualizing data using t-SNE. *Journal of Machine Learning Research*.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, Ł.; and Polosukhin, I. 2017. Attention is All you Need. In *NIPS*.

Yang, B., and Mitchell, T. 2017. Leveraging knowledge bases in lstms for improving machine reading. In *ACL*.