Self-regulation: Employing a Generative Adversarial Network to Improve Event Detection

Yu Hong Wenxuan Zhou Jingli Zhang Qiaoming Zhu Guodong Zhou[*]
Institute of Artificial Intelligence, Soochow University
School of Computer Science and Technology, Soochow University
No.1, Shizi ST, Suzhou, China, 215006
{tianxianer, wxchow024, jlzhang05}@gmail.com
{qmzhu, gdzhou}@suda.edu.cn

Abstract

Due to the ability of encoding and mapping semantic information into a high-dimensional latent feature space, neural networks have been successfully used for detecting events to a certain extent. However, such a feature space can be easily contaminated by spurious features inherent in event detection. In this paper, we propose a self-regulated learning approach by utilizing a generative adversarial network to generate spurious features. On the basis, we employ a recurrent network to eliminate the fakes. Detailed experiments on the ACE 2005 and TAC-KBP 2015 corpora show that our proposed method is highly effective and adaptable.

1 Introduction

Event detection aims to locate the event triggers of specified types in text. Normally, triggers are words or nuggets that evoke the events of interest.

Detecting events in an automatic way is challenging, not only because an event can be expressed in different words, but also because a word may express a variety of events in different contexts. In particular, the frequent utilization of common words, ambiguous words and pronouns in event mentions makes them harder to detect:

1) **Generality** – *<u>taken</u> home* <`Transport`>
   Ambiguity 1 – *<u>campaign</u> in Iraq* <`Attack`>
   Ambiguity 2 – *political <u>campaign</u>* <`Elect`>
   Coreference – *Either <u>its</u> bad or good* <`Marry`>

A promising solution to this challenge is through semantic understanding. Recently, neural networks have been widely used in this direction (Nguyen and Grishman, 2016; Ghaeini et al., 2016; Feng et al., 2016; Liu et al., 2017b; Chen et al., 2017), which allows semantics of event mentions (trigger plus context) to be encoded in a high-dimensional latent feature space. This facilitates the learning of deep-level semantics. Besides, the use of neural networks not only strengthens current supervised classification of events but alleviates the complexity of feature engineering.

However, compared to the earlier study (Liao and Grishman, 2010; Hong et al., 2011; Li et al., 2013), in which the features are carefully designed by experts, the neural network based methods suffer more from spurious features. Here, spurious feature is specified as the latent information which looks like the semantically related information to an event, but actually not (Liu et al., 2017a). For example, in the following sample, the semantic information of the word "*prison*" most probably enables spurious features to come into being, because the word often co-occurs with the trigger "*taken*" to evoke an `Arrest-jail` event instead of the ground-truth event `Transport`:

2) *<u>Prison</u> authorities have given the nod for Anwar to be <u>taken</u> home later in the afternoon.*
   Trigger: *taken*.   **Event Type**: `Transport`

It is certain that spurious features often result from the semantically pseudo-related context, and during training, a neural network may mistakenly and unconsciously preserve the memory to produce the fakes. However, it is difficult to determine which words are pseudo-related in a specific case, and when they will "jump out" to mislead the generation of latent features during testing.

To address the challenge, we suggest to regulate the learning process with a two-channel self-regulated learning strategy. In the self-regulation process, on one hand, a generative adversarial network is trained to produce the most spurious features, while on the other hand, a neural network

[*] Corresponding author

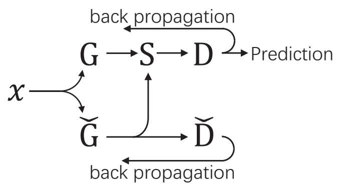

Figure 1: Self-regulated learning scheme

is equipped with a memory suppressor to eliminate the fakes. Detailed experiments on event detection show that our proposed method achieves a substantial performance gain, and is capable of robust domain adaptation.

2 Task Definition

The task of event detection is to determine whether there is one or more event triggers in a sentence. Trigger is defined as a token or nugget that best signals the occurrence of an event. If successfully identified, a trigger is required to be assigned a tag to indicate the event type:

Input: *Either its bad or good*

Output: *its* <trigger>; Marry <type>

We formalize the event detection problem as a multi-class classification problem. Given a sentence, we classify every token of the sentence into one of the predefined event classes (Doddington et al., 2004) or non-trigger class.

3 Self-Regulated Learning (SELF)

SELF is a double-channel model (Figure 1), consisted of a cooperative network (Islam et al., 2003) and a generative adversarial net (GAN) (Goodfellow et al., 2014). A memory suppressor *S* is used to regulate communication between the channels.

3.1 Cooperative Network

In channel 1, the generator G is specified as a multilayer perceptron. It plays a role of a "diligent student". By a differentiable function $G(x, \theta_g)$ with parameters θ_g, the generator learns to produce a vector of latent features o_g that may best characterize the token x, i.e., $o_g = G(x, \theta_g)$.

The discriminator D (called "a lucky professor") is a single-layer perceptron, implemented as a differentiable function $D(o_g, \theta_d)$ with parameters θ_d. Relying on the feature vector o_g, it attempts to accurately predict the probability of the token x triggering an event for all event classes, i.e., $\hat{y} = D(o_g, \theta_d)$, and assigns x to the most probable class *c* (*iff* $\hat{y}_c > \forall \hat{y}_{\bar{c}}, \bar{c} \neq c$).

Therefore, G and D cooperate with each other during training, developing the parameters θ_g and θ_d with the same goal – to minimize the performance loss $\mathcal{L}(\hat{y}, y)$ in the detection task:

$$\begin{bmatrix} \theta_g \\ \theta_d \end{bmatrix} = argmin\ \mathcal{L}(\hat{y}, y) \tag{1}$$

where, y denotes the ground-truth probability distribution over event classes, and $\mathcal{L}$ indicates the deviation of the prediction from the ground truth.

3.2 Generative Adversarial Network

In channel 2, the generator $\breve{G}$ and discriminator $\breve{D}$ have the same perceptual structures as G and D. They also perform learning by differentiable functions, respectively $\breve{G}(x, \theta_{\breve{g}})$ and $\breve{D}(o_{\breve{g}}, \theta_{\breve{d}})$. A major difference, however, is that they are caught into a cycle of highly adversarial competition.

The generator $\breve{G}$ is a "trouble maker". It learns to produce spurious features, and utilizes them to contaminate the feature vector $o_{\breve{g}}$ of the token x. Thus $\breve{G}$ changes a real sample x into a fake z – sometimes successfully, sometimes less so. Using the fakes, $\breve{G}$ repeatedly instigates the discriminator $\breve{D}$ to make mistakes. On the other side, $\breve{D}$ ("a hapless professor") has to avoid being deceived, and struggles to correctly detect events no matter whether it encounters x or z.

In order to outsmart the adversary, $\breve{G}$ develops the parameters $\theta_{\breve{g}}$ during training to maximize the performance loss, but on the contrary, $\breve{D}$ develops the parameters $\theta_{\breve{d}}$ to minimize the loss:

$$\theta_{\breve{g}} = argmax\ \mathcal{L}(\hat{y}, y) \tag{2}$$

$$\theta_{\breve{d}} = argmin\ \mathcal{L}(\hat{y}, y) \tag{3}$$

Numerous studies have confirmed that the two-player minmax game enables both $\breve{G}$ and $\breve{D}$ to improve their methods (Goodfellow et al., 2014; Liu and Tuzel, 2016; Huang et al., 2017).

3.3 Regulation with Memory Suppressor

Using a memory suppressor, we try to optimize the diligent student G. The goal is to enable G to be as dissimilar as possible to the troublemaker $\breve{G}$.

The suppressor uses the output $o_{\breve{g}}$ of $\breve{G}$ as a reference resource which should be full of spurious features. On the basis, it looks over the output o_g of G, so as to verify whether the features in o_g are different to those in $o_{\breve{g}}$. If very different, the suppressor allows G to preserve the memory (viz., θ_g in $G(x, \theta_g)$), otherwise update. In other word, for G, the suppressor forcibly erases the memory which may result in the generation of spurious features. We call this the self-regulation.

Self-regulation is performed for the whole sentence which is fed into G and $\check{G}$. Assume that O_g is a matrix, constituted with a series of feature vectors, i.e., the vectors generated by G for all the tokens in an input sentence ($o_g \in O_g$), while $O_{\check{g}}$ is another feature matrix, generated by $\check{G}$ for the tokens ($o_{\check{g}} \in O_{\check{g}}$). Thus, we utilize the matrix approximation between O_g and $O_{\check{g}}$ for measuring the loss of self-regulation learning $\mathcal{L}_{diff}$. The higher the similarity, the greater the loss. During training, the generator G is required to develop the parameters θ_g to minimize the loss:

$$\theta_g = argmin\ \mathcal{L}_{diff}(o_g, o_{\check{g}}) \tag{4}$$

We present in detail the matrix approximate calculation in section 4.4, where the squared Frobenius norm (Bousmalis et al., 2016) is used.

3.4 Learning to Predict

We incorporate the cooperative network with the GAN, and enhance their learning by joint training.

In the 4-member incorporation, i.e., $\{G, \check{G}, D$ and $\check{D}\}$, the primary beneficiary is the lucky professor D. It can benefit from both the cooperation in channel 1 and the competition in channel 2. The latent features it uses are well-produced by G, and decontaminated by eliminating possible fakes like those made by $\check{G}$. Therefore, in experiments, we choose to output the prediction results of D.

In this paper, we use two recurrent neural networks (RNN) (Sutskever et al., 2014; Chung et al., 2014) of the same structure as the generators. And both the discriminators are implemented as a fully-connected layer followed by a softmax layer.

4 Recurrent Models for SELF

RNN with long short-term memory (abbr., LSTM) is adopted due to the superior performance in a variety of NLP tasks (Liu et al., 2016a; Lin et al., 2017; Liu et al., 2017a). Furthermore, the bidirectional LSTM (Bi-LSTM) architecture (Schuster and Paliwal, 1997; Ghaeini et al., 2016; Feng et al., 2016) is strictly followed. This architecture enables modeling of the semantics of a token with both the preceding and following contexts.

4.1 LSTM based Generator

Given a sentence, we follow Chen et al (2015) to take all the tokens of the whole sentence as the input. Before feeding the tokens into the network, we transform each of them into a real-valued vector $x \in \mathbb{R}^e$. The vector is formed by concatenating a word embedding with an entity type embedding.

- **Word Embedding**: It is a fixed-dimensional real-valued vector which represents the hidden semantic properties of a token (Collobert and Weston, 2008; Turian et al., 2010).

- **Entity Type Embedding**: It is specially used to characterize the entity type associated with a token. The BIO2 tagging scheme (Wang and Manning, 2013; Huang et al., 2015) is employed for assigning a type label to each token in the sentence.

For the input token x_t at the current time step t, the LSTM generates the latent feature vector $o_t \in \mathbb{R}^d$ by the previous memory. Meanwhile, the token is used to update the current memory.

The LSTM possesses a long-term memory unit $c_t \in \mathbb{R}^d$ and short-term $\widetilde{c}_t \in \mathbb{R}^d$. In addition, it is equipped with the input gate i_t, forgetting gate f_t and a hidden state h_t, which are assembled together to promote the use of memory, as well as dynamic memory updating. Similarly, they are defined as a d-dimensional vector in $\mathbb{R}^d$. Thus LSTM works in the following way:

$$\begin{bmatrix} o_t \\ \widetilde{c}_t \\ i_t \\ f_t \end{bmatrix} = \begin{bmatrix} \sigma \\ tanh \\ \sigma \\ \sigma \end{bmatrix} \left(\mathrm{W} \begin{bmatrix} x_t \\ h_{t-1} \end{bmatrix} + \mathrm{b} \right) \tag{5}$$

$$h_t = o_t \odot tanh(c_t) \tag{6}$$

$$c_t = \widetilde{c}_t \odot i_t + c_{t-1} \odot \mathrm{f}_t \tag{7}$$

where $\mathrm{W} \in \mathbb{R}^{4d \times (d+e)}$ and $\mathrm{b} \in \mathbb{R}^{4d}$ are parameters of affine transformation; σ refers to the logistic sigmoid function and $\odot$ denotes element-wise multiplication.

The output functions of both the generators in SELF, i.e., G and $\check{G}$, can be boiled down to the output gate $o_t \in \mathbb{R}^d$ of the LSTM cell:

$$o_t = LSTM(x_t; \theta) \tag{8}$$

where, the function LSTM (·;·) is a shorthand for Eq. (5-7) and θ represents all the parameters of LSTM. For both G and $\check{G}$, θ are initialized with the same values in experiments. But due to the distinct training goals of G and $\check{G}$ (diligence or making-trouble), the values of the parameters in the two cases will change to be very different after training. Therefore, we have $o_{g,t} = LSTM(x_t, \theta_{g,t})$ and $o_{\breve{g},t} = LSTM(x_t, \theta_{\breve{g},t})$.

4.2 Fully-connected Layer for Discrimination

Depending on the feature vectors $o_{g,t}$ and $o_{\breve{g},t}$, the two discriminators D and $\breve{D}$ predict the probability of the token x_t triggering an event for all event classes. As usual, they compute the probability distribution over classes using a fully connected layer followed by a softmax layer:

$$\hat{y} = softmax(\hat{W} \cdot o_t + \hat{b}) \tag{9}$$

where $\breve{y}$ is a C-dimensional vector, in which each dimension indicates the prediction for a class; C is the class number; $\hat{W} \in \mathbb{R}^d$ is the weight which needs to be learned; $\hat{b}$ is a bias term.

It is noteworthy that the discriminator D and $\breve{D}$ don't share the weight and the bias. It means that, for the same token x_t, they may make markedly different predictions: $\hat{y}_{g,t} = softmax(\hat{W}_g \cdot o_{g,t} + \hat{b}_g)$ and $\hat{y}_{\breve{g},t} = softmax(\hat{W}_{\breve{g}} \cdot o_{\breve{g},t} + \hat{b}_{\breve{g}})$.

4.3 Classification Loss

We specify the loss as the cross-entropy between the predicted and ground-truth probability distributions over classes. Given a batch of training data that includes N samples (x_i, y_i), we calculate the losses the discriminators cause as below:

$$\mathcal{L}(\hat{y}_g, y) = -\sum_{i=1}^{N}\sum_{j=1}^{C} y_i^j log(\hat{y}_{g,i}^j) \tag{10}$$

$$\mathcal{L}(\hat{y}_{\breve{g}}, y) = -\sum_{i=1}^{N}\sum_{j=1}^{C} y_i^j log(\hat{y}_{\breve{g},i}^j) \tag{11}$$

where y_i is a C-dimensional one-hot vector. The value of its j-th dimension is set to be 1 only if the token x_i triggers an event of the j-th class, otherwise 0. Both $\hat{y}_{g,i}$ and $\hat{y}_{\breve{g},i}$ are the predicted probability distributions over the C classes for x_i.

4.4 Loss of Self-regulated Learning

Assume that O_g is a matrix, consisted of the feature vectors output by G for all the tokens in a sentence, i.e., $o_{g,t} \in O_g$, and $O_{\breve{g}}$ is that provided by $\breve{G}$, i.e., $o_{\breve{g},t} \in O_{\breve{g}}$, thus we compute the similarity between O_g and $O_{\breve{g}}$ and use it as the measure of self-regulation loss $\mathcal{L}_{diff}(O_g, O_{\breve{g}})$:

$$\mathcal{L}_{diff}(O_g, O_{\breve{g}}) = \|O_g O_{\breve{g}}^\top\|_F^2 \tag{12}$$

where, $\|\cdot\|_F^2$ denotes the squared Frobenius norm (Bousmalis et al., 2016), which is used to calculate the similarity between matrices.

It is noteworthy that the feature vectors a generator outputs are required to serve as the rows in the matrix, deployed in a top-down manner and arranged in the order in which they are generated. For example, the feature vector $o_{g,t}$ the generator G outputs at the time t needs to be placed in the t-th row of the matrix O_g.

At the very beginning of the measurement, the similarity between every feature vector in O_g and that in $O_{\breve{G}}$ is first calculated by the matrix-matrix multiplication $O_g O_{\breve{g}}^\top$:

$$\begin{pmatrix} o_{g,1}o_{\breve{g},1}^\top & \cdots & o_{g,1}o_{\breve{g},t}^\top & \cdots & o_{g,1}o_{\breve{g},l}^\top \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ o_{g,1}o_{\breve{g},t}^\top & \cdots & o_{g,t}o_{\breve{g},t}^\top & \cdots & o_{g,t}o_{\breve{g},l}^\top \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ o_{g,1}o_{\breve{g},l}^\top & \cdots & o_{g,l}o_{\breve{g},t}^\top & \cdots & o_{g,l}o_{\breve{g},l}^\top \end{pmatrix}$$

where, the symbol $\top$ denotes the transpose operation; l is the sentence length which is defined to be uniform for all sentences (l=80), and if it is larger than the real ones, padding is used; $o_{g,i}o_{\breve{g},j}$ denotes the scalar product between the feature vectors $o_{g,i}$ and $o_{\breve{g},j}$.

Let $A_{m\times n}$ be a matrix, the squared Frobenius norm of $A_{m\times n}$ (i.e., $\|A_{m\times n}\|_F^2$) is defined as:

$$\|A_{m\times n}\|_F^2 = \left(\sum_{i=1}^{m}\sum_{j=1}^{n} |a_{ij}|^2\right)^{\frac{1}{2}} \tag{13}$$

where, a_{ij} denotes the j-th element in the i-th row of $A_{m\times n}$. Thus, if we let $A_{m\times n}$ be the matrix produced by the matrix-matrix multiplication $O_g O_{\breve{g}}^\top$, the self-regulation loss $\mathcal{L}_{diff}(O_g, O_{\breve{g}})$ can be eventually obtained by:

$$\mathcal{L}_{diff}(O_g, O_{\breve{g}}) = \left(\sum_{i=1}^{l}\sum_{j=1}^{l} |o_{g,i}o_{\breve{g},j}|^2\right)^{\frac{1}{2}} \tag{14}$$

For a batch of training data that includes N' sentences, the global self-regulation loss is specified as the sum of the losses for all the sentences: $\mathcal{L}_{SELF} = \sum_{i=1}^{N'} \mathcal{L}_{diff}(O_g, O_{\breve{g}})$.

4.5 Training

We train the cooperative network in SELF to minimize the classification loss $\mathcal{L}(\hat{y}_g, y)$ and the loss of self-regulated learning $\mathcal{L}_{SELF}$:

$$\theta_g = argmin\ (\mathcal{L}\hat{y}_g, y) \qquad (15)$$

$$\theta_d = argmin\ (\mathcal{L}(\hat{y}_g, y) + \lambda \cdot \mathcal{L}_{SELF}) \qquad (16)$$

where λ is a hyper-parameter, which is used to harmonize the two losses.

The min-max game is utilized for training the adversarial net in SELF: $\theta_{\breve{g}} = argmax\ \mathcal{L}(\hat{y}_{\breve{g}}, y)$; $\theta_{\breve{d}} = argmin\ \mathcal{L}(\hat{y}_{\breve{g}}, y)$.

All the networks in SELF are trained jointly using the same batches of samples. They are trained via stochastic gradient descent (Nguyen and Grishman, 2015) with shuffled mini-batches and the AdaDelta update rule (Zeiler, 2012). The gradients are computed using back propagation. And regularization is implemented by a dropout (Hinton et al., 2012).

5 Experimentation

5.1 Resource and Experimental Datasets

We test the presented model on the ACE 2005 corpus. The corpus is annotated with single-token event triggers and has 33 predefined event types (Doddington et al., 2004; Ahn, 2006), along with one class "*None*" for the non-trigger tokens, constitutes a 34-class classification problem.

For comparison purpose, we use the corpus in the traditional way, randomly selecting 30 articles in English from different genres as the development set, and utilizing a separate set of 40 English newswire articles as the test set. The remaining 529 English articles are used as the training set.

5.2 Hyperparameter Settings

The word embeddings are initialized with the 300-dimensional real-valued vectors. We follow Chen et al (2015) and Feng et al (2016) to pre-train the embeddings over NYT corpus using Mikolov et al (2013)'s skip-gram tool. The entity type embeddings, as usual (Nguyen et al., 2016; Feng et al., 2016; Liu et al., 2017b), are specified as the 50-dimensional real-valued vectors. They are initialized with the 32-bit floating-point values, which are all randomly sampled from the uniformly distributed values in [-1, 1][1]. We initialize other adjustable parameters of the back-propagation algorithm by randomly sampling in [-0.1, 0.1].

We follow Feng et al (2016) to set the dropout rate as 0.2 and the mini-batch size as 10. We tune the initialized parameters mentioned above, harmonic coefficient λ, learning rate and the L2 norm on the development set. Grid search (Liu et al., 2017a) is used to seek for the optimal parameters. Eventually, we take the coefficient λ of 0.1^{+3}, learning rate of 0.3 and L2 norm of 0.

The source code of SELF[2] to reproduce the experiments has been made publicly available.

5.3 Compared Systems

The state-of-the-art models proposed in the past decade are compared with ours. By taking learning framework as the criterion, we divide the models into three classes:

Minimally supervised approach: is Peng et al (2016)'s **MSEP-EMD**.

Feature based approaches: primarily including Liao and Grishman (2010)'s **Cross-Event** inference model, which is based on the max-entropy classification and embeds the document-level confident information in the feature space; Hong et al (2011)'s **Cross-Entity** inference model, in which existential backgrounds of name entities are employed as the additional discriminant features; and Li et al (2013)'s **Joint** model, a sophisticated predictor frequently ranked among the top 3 in recent TAC-KBP evaluations for nugget and coreference detection (Hong et al., 2014, 2015; Yu et al., 2016). It is based on structured perceptron and combines the local and global features.

Neural network based approaches: including the convolutional neural network (**CNN**) (Nguyen and Grishman, 2015), the non-consecutive *N*-grams based CNN (**NC-CNN**) (Nguyen and Grishman, 2016) and the CNN that is assembled with a dynamic multi-pooling layer (**DM-CNN**) (Chen et al., 2015). Others include Ghaeini et al (2016)'s forward-backward recurrent neural network (**FB-RNN**) which is developed using gated recurrent units (GRU), Nguyen et al (2016)'s bidirectional RNN (**Bi-RNN**) and Feng et al (2016)'s **Hybrid** networks that consist of a Bi-LSTM and a CNN.

Besides, we compare our model with Liu et al (2016b)'s artificial neural networks (**ANNs**), Liu et al (2017b)'s attention-based ANN (**ANN-S2**) and Chen et al (2017)'s **DM-CNN***. The models recently have become popular because, although simple in structure, they are very analytic by learning from richer event examples, such as those in

[1]https://www.tensorflow.org/api_docs/python/tf/random_uniform

[2]https://github.com/JoeZhouWenxuan/Self-regulation-Employing-a-Generative-Adversarial-Network-to-Improve-Event-Detection/tree/master

Method	P (%)	R (%)	F (%)
Joint (Local+Global)	76.9	65.0	70.4
MSEP-EMD	75.6	69.8	72.6
DM-CNN	80.4	67.7	73.5
DM-CNN*	79.7	69.6	74.3
Bi-RNN	68.5	75.7	71.9
Hybrid: Bi-LSTM+CNN	80.8	71.5	75.9
SELF: Bi-LSTM+GAN	**75.3**	**78.8**	**77.0**

Table 1: Trigger identification performance

FrameNet (**FN**) and Wikipeida (**Wiki**).

5.4 Experimental Results

We evaluate our model using Precision (P), Recall (R) and F-score (F). To facilitate the comparison, we review the best performance of the competitors, which has been evaluated using the same metrics, and publicly reported earlier.

Trigger identification

Table 1 shows the trigger identification performance. It can be observed that SELF outperforms other models, with a performance gain of no less than 1.1% F-score.

Frankly, the performance mainly benefits from the higher recall (78.8%). But in fact the relatively comparable precision (75.3%) to the recall reinforces the advantages. By contrast, although most of the compared models achieve much higher precision over SELF, they suffer greatly from the substantial gaps between precision and recall. The advantage is offset by the greater loss of recall.

GAN plays an important role in optimizing Bi-RNN. This is proven by the fact that SELF (Bi-LSTM+GAN) outperforms Nguyen et al (2016)'s Bi-RNN. To be honest, the models use two different kinds of recurrent units. Bi-RNN uses GRUs, but SELF uses the units that possess LSTM. Nevertheless, GRU has been experimentally proven to be comparable in performance to LSTM (Chung et al., 2014; Jozefowicz et al., 2015). This allows a fair comparison between Bi-RNN and SELF.

Event classification

Table 2 shows the performance of multi-class classification. SELF achieves nearly the same F-score as Feng et al (2016)'s Hybrid, and outperforms the others. More importantly, SELF is the only one which obtains a performance higher than 70% for both precision and recall.

Besides, by analyzing the experimental results, we have identified the following regularities:

Methods	P (%)	R (%)	F (%)
MSEP-EMD	70.4	65.0	67.6
Cross-Event	68.8	68.9	68.8
Cross-Entity	72.9	64.3	68.3
Joint (Local+Global)	73.7	62.3	67.5
CNN	71.8	66.4	69.0
DM-CNN	75.6	63.6	69.1
NC-CNN	-	-	71.3
FB-RNN (GRU)	66.8	68.0	67.4
Bi-RNN (GRU)	66.0	73.0	69.3
ANNs (ACE+FN)	77.6	65.2	70.7
DM-CNN*(ACE+Wiki)	75.7	66.0	70.5
ANN-S2 (ACE+FN)	76.8	67.5	71.9
Hybrid: Bi-LSTM+**CNN**	**84.6**	64.9	**73.4**
SELF: Bi-LSTM+**GAN**	71.3	**74.7**	**73.0**

Table 2: Detection performance (trigger identification plus multi-class classification)

- Similar to the pattern classifiers that are based on hand-designed features, the CNN models enable higher precision to be obtained. However the recall is lower.
- The RNN models contribute to achieving a higher recall. However the precision is lower.
- Expansion of the training data set helps to increase the precision.

Let us turn to the structurally more complicated models, SELF and Hybrid.

SELF inherits the merits of the RNN models, classifying the events with higher recall. Besides, by the utilization of GAN, SELF has evolved from the traditional learning strategies, being capable of learning from GAN and getting rid of the mistakenly generated spurious features. So that it outperforms other RNNs, with improvements of no less than 4.5% precision and 1.7% recall.

Hybrid is elaborately established by assembling a RNN with a CNN. It models an event from two perspectives: language generation and pragmatics. The former is deeply learned by using the continuous states hidden in the recurrent units, while the later the convolutional features. Multi-angled cognition enables Hybrid to be more precise. However it is built using a single-channel architecture, concatenating the RNN and the CNN. This results in twofold accumulation of feature information, causing a serious overfitting problem. Therefore, Hybrid is localized to much higher precision but substantially lower recall.

Overfitting results in enlargement of the gap between precision and recall when the task changes to be more difficult. For Hybrid, as illustrated in

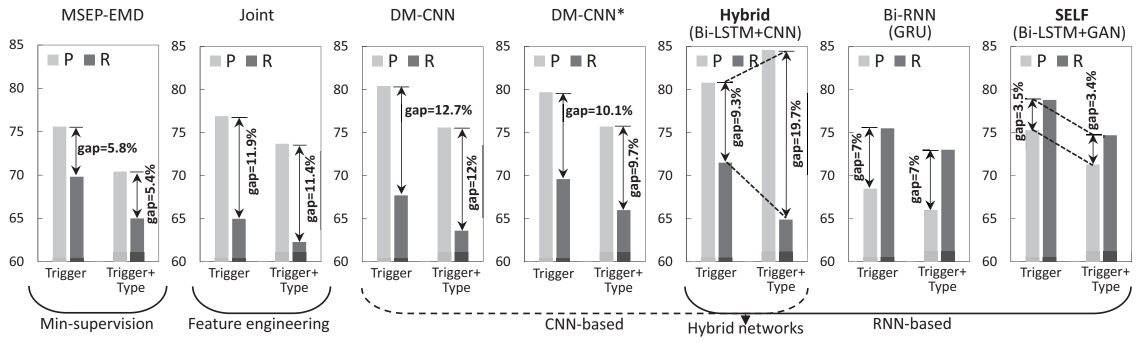

Figure 2: Gaps between precision and recall in the tasks of trigger identification and event classification

Methods	Embedding Types	Training Data
ANNs	word	ACE+FN
ANN-S2	word, NE-type	ACE+FN
DM-CNN*	word, PSN	ACE+Wiki
CNN	word, NE-type, PSN	ACE
NC-CNN	word, NE-type, PSN	ACE
Bi-RNN	word, NE-type, DEP	ACE
Hybrid	word, NE-type, PSN	ACE
DM-CNN	word, PSN	ACE
FB-RNN	word, branch	ACE
SELF	**word, NE-type**	**ACE**

Table 3: Embedding types and training data (DEP: Dependency grammar; PSN: Position)

Figure 2, the gap becomes much wider (from 9% to 19.7%) when the binary classification task (trigger identification) is shifted to multi-class classification (event detection). By contrast, other work shows a nearly constant gap. In particular, SELF yields a minimum gap in each task, which changes negligibly from 3.5% to 3.4%.

It may be added that, similar to DM-CNN and FB-RNN, SELF is cost-effective. Compared to other models (Table 3), it either uses less training data, or is only required to learn two kinds of embeddings, such as that of words and entity types.

5.5 Discussion: Adaptation, Robustness and Effectiveness

Domain adaptation is a key criteria for evaluating the utility of a model in practical application. A model can be thought of being adaptable only if it works well for the unlabeled data in the target domain when trained on the source domain (Blitzer et al., 2006; Plank and Moschitti, 2013).

We perform two groups of domain adaptation experiments, respectively, using the ACE 2005 corpus and the corpus for TAC-KBP 2015 event nugget track (Ellis et al., 2015).

The ACE corpus consists of 6 domains: broadcast conversation (`bc`), broadcast news (`bn`), telephone conversation (`cts`), newswire (`nw`), usenet (`un`) and web blogs (`wl`). Following the common practice of adaptation research on this data (Nguyen and Grishman, 2014, 2015; Plank and Moschitti, 2013), we take the union of `bn` and `nw` as the source domain and `bc`, `cts` and `wl` as three different target domains. We randomly select half of the instances from `bc` to constitute the development set. The TAC-KBP corpus consists of 2 domains: newswire (`NW`) and discussion forum (`DF`). We follow Peng et al (2016) to use one of `NW` and `DF` in alternation as the source domain, while the other the target domain. We randomly select a proportion (20%) of the instances from the target domain to constitute the development set.

We compare with **Joint**, **CNN**, **MSEP-EMD**, **SSED** (Sammons et al., 2015) and **Hybrid**. All the models except Hybrid have been reported for the performance assessment of domain adaptation. In this section, we only cite the best performance they obtained. We reproduce Hybrid by using the source code given by authors. To ensure a fair comparison, we perform 3 runs, in each of which, both Hybrid and SELF were redeveloped on a new development set. What we report herein is the average performance they obtained over the 3 runs.

Adaptation Performance

We show the adaptation performance on the ACE corpus in Tables 4 and that on TAC-KBP in Table 5. It can be observed that SELF outperforms other models in the out-of-domain scenarios.

Besides, when testing is performed on the out-of-domain ACE corpus, the performance degradation of SELF is not much larger than that of CNN and Hybrid. When the out-of-domain TAC-KBP corpus is used, the performance of SELF is impaired much less severely than SSED and Hybrid.

Methods	In-domain (`bn+nw`)			Out-of-domain (`bc`)				Out-of-domain (`cts`)				Out-of-domain (`wl`)			
	P(%)	R(%)	F(%)	P(%)	R(%)	F(%)	Loss	P(%)	R(%)	F(%)	Loss	P(%)	R(%)	F(%)	Loss
Joint	72.9	63.2	67.7	68.8	57.5	62.6	↓5.1	64.5	52.3	57.7	↓10.0	56.4	38.5	45.7	↓22.0
CNN	69.2	67.0	68.0	70.2	65.2	67.6	↓0.4	68.3	58.2	62.8	↓5.2	54.8	42.0	47.5	↓20.5
Hybrid	68.8	54.8	61.0	64.7	58.8	61.6	↑0.6	59.9	50.6	54.9	↓6.1	54.0	37.9	44.5	↓16.5
SELF	**73.8**	**65.7**	**69.5**	**70.0**	**67.2**	**68.9**	**↓0.6**	**68.3**	**60.2**	**63.3**	**↓6.2**	**58.0**	**44.0**	**50.0**	**↓19.5**

Table 4: Experimental results of domain adaptation on the ACE 2005 corpus

Methods	In-domain (`NW`)			Out-of-domain (`DF`)				In-domain (`DF`)			Out-of-domain (`NW`)			
	P(%)	R(%)	F(%)	P(%)	R(%)	F(%)	Loss	P(%)	R(%)	F(%)	P(%)	R(%)	F(%)	Loss
MSEP-EMD	NA	NA	58.5	NA	NA	52.8	↓5.7	NA	NA	57.9	NA	NA	55.1	↓2.8
SSED	NA	NA	63.7	NA	NA	52.3	↓11.4	NA	NA	62.6	NA	NA	54.8	↓7.8
Hybrid	72.6	55.4	62.9	62.3	39.2	48.1	↓14.8	66.0	42.6	51.8	59.1	48.4	53.3	↑1.5
SELF	**67.6**	**60.6**	**63.9**	**69.0**	**58.7**	**56.7**	**↓7.2**	**70.5**	**48.3**	**57.3**	**69.3**	**51.7**	**59.2**	**↑1.9**

Table 5: Experimental results of domain adaptation on the TAC-KBP 2015 corpus (NA: not released)

More importantly, the adaptability of SELF is relatively close to that of MSEP-EMD. Considering that MSEP-EMD is stable due to using minimal supervision (Peng et al., 2016), we suggest the fully trained networks in SELF may not appear to be extremely inflexible, but on the contrary, they should be transferable for use (Ge et al., 2016).

Robustness in Resource-Poor Settings

There are two resource-poor conditions discussed in this section, including lack of in-domain training data and that of out-domain. Hybrid and SELF are brought into the discussion.

For the former (**in-domain**) case, we went over the numbers of samples used for training in the adaptation experiments, which are shown in Table 6. It can be observed that there is a minimum number of training samples (triggers plus tokens) contained in the domain of `NW`. By contrast, the domain of `bn+nw` contains the smallest number of positive samples (triggers) though an overwhelming number of negative samples (general tokens).

Under such conditions, Hybrid performs better in the domain of `NW` compared to `bn+nw` and `DF` in the three in-domain adaptation experiments (see the column labelled as "In-domain `bn+nw`" in Table 4 as well as "In-domain `NW`" and "In-domain `DF`" in Table 5). It illustrates that Hybrid unnecessarily relies on a tremendous number of training samples to ensure the robustness. But SELF does. It needs far more negative samples than Hybrid because of the following reasons:

- It relies on the use of spurious features to implement self-regulation during training.

Domain	**Training**		**Testing**	
	trigger	token	trigger	token
`bn+nw`	1,721	74,179	343	16,336
`NW`	2,098	31,014	2,813	55,459
`DF`	4,106	10,9275	1,773	43,877

Table 6: Data distribution in the source domains

- For a positive sample, the concerned spurious features (if have) most probably hide in some negative samples.
- It's impossible to be aware of such negative samples. Therefore, taking into consideration as many negative samples as possible may help to increase the probability that the spurious features will be discovered.

This is demonstrated by the fact that SELF obtains better performance in the domain of `bn+nw` but not `NW` (see the column labeled as "Training" in Table 6 and "In-domain" in Table 4 and 5). It may be added that SELF performs worse in `DF` although there are more negative samples used for training (see Table 6). Taking a glance at the number of positive samples, one may find that it is approximately 2.4 times more than that in `bn+nw`. But the number of negative samples in `DF` is only 1.5 times more than that in `bn+nw`. It implies that, if there are more positive samples used for training, SELF needs to consume proportionally more negative samples for self-regulation. Otherwise, the performance will degrade.

For the **out-domain** case, ideally, both Hybrid and SELF encounter the problem that there is lack of target domain data available for training. In this case, SELF displays less performance degradation

Event mentions	**Type**
*And **it** still does*	Die
*We had no part in **it***	Arrest-Jail
*Nobody questions if **this** is right or ...*	Attack
*And **that** is what ha- what is happening*	End-Position
*Oh, yeah, **it** wasn't perfect*	Marry

Table 7: Examples of pronouns that act as a trigger

(7.2%) than Hybrid (14.8%) when `NW` is used for training. Considering that `NW` contains the minimum number of samples, we would like to believe that SELF is more robust than Hybrid for cross-domain event detection in a resource-poor setting.

Recall and Missing

SELF is able to accurately recall the events whose occurrence is triggered by ambiguous words, such as "*fine*", "*charge*", "*campaign*", etc. These ambiguous words easily causes confusion. For example, "*campaign*" may trigger an `Elect` event or `Attack` in the ACE corpus.

More importantly, SELF fishes out the common words which serve as a trigger, although they are not closely related to any kind of events, such as "*take*", "*try*", "*acquire*", "*become*", "*create*", etc. In general, it is very difficult to accurately recall such triggers because their meanings are not concrete enough, and the contexts may be full of kinds of noises (see example 2 in pg. 1). We observe that Bi-RNN and Hybrid seldom pick them up.

However, SELF fails to recall the pronouns that act as a trigger. This is because they occur in spoken language much more frequently than they occur in written language. The lack of narrative content makes it difficult to learn the relationship between the pronouns and the events. Some real examples collected from ACE are shown in Table 7.

6 Related Work

Event detection is an important subtask of event extraction (Doddington et al., 2004; Ahn, 2006).

The research can be traced back to the pattern based approach (Grishman et al., 2005). Encouraged by the high accuracy and the benefit of easy-to-use, researchers have made great efforts to extract discriminative patterns. Cao et al (2015a; 2015b) use dependency regularization and active leaning to generalize and expand the patterns.

In the earlier study, another trend is to explore the features that best characterize each event class, so as to facilitate supervised classification. A variety of strategies have emerged for converting classification clues into feature vectors (Ahn, 2006; Patwardhan and Riloff, 2009; Liao and Grishman, 2010; Hong et al., 2011; Li et al., 2013, 2014; Wei et al., 2017). Benefiting from the general modeling framework, the methods enable the fusion of multiple features, and more importantly, they are flexible to use by feature selection. But considerable expertise is required for feature engineering.

Recently, the use of neural networks for event detection has become a promising line of research. The closely related work has been presented in section 5.3. The primary advantages of neural networks have been demonstrated in the work, such as performance enhancement, self-learning capability and robustness.

The generative adversarial network (Goodfellow et al., 2014) has emerged as an increasingly popular approach for text processing (Zhang et al., 2016; Lamb et al., 2016; Yu et al., 2017). Liu et al (2017a) use the adversarial multi-task learning for text classification. We follow the work to create spurious features, but use them to regulate the self-learning process in a single-task situation.

7 Conclusion

We use a self-regulated learning approach to improve event detection. In the learning process, the adversarial and cooperative models are utilized in decontaminating the latent feature space.

In this study, the performance of the discriminator in the adversarial network is left to be evaluated. Most probably, the discriminator also performs well because it is gradually enhanced by fierce competition. Considering this possibility, we suggest to drive the two discriminators in our self-regulation framework to cooperate with each other. Besides, the global features extracted in Li et al (2013)'s work are potentially useful for detecting the event instances referred by pronouns, although involve noises. Therefore, in the future, we will encode the global information by neural networks and use the self-regulation strategy to reduce the negative influence of noises.

Acknowledgments

We thank Xiaocheng Feng and his colleagues who shared the source code of Hybrid with us.

This work was supported by the national Natural Science Foundation of China (NSFC) via Grant Nos. 61525205, 61751206, 61672368.

References

David Ahn. 2006. The stages of event extraction. In *Proceedings of the Workshop on Annotating and Reasoning about Time and Events, Association for Computational Linguistics (ACL'06)*. Association for Computational Linguistics, pages 1–8. http://www.aclweb.org/anthology/W06-0901.

John Blitzer, Ryan McDonald, and Fernando Pereira. 2006. Domain adaptation with structural correspondence learning. In *Proceedings of the 2006 conference on Empirical Methods in Natural Language Processing (EMNLP'06)*. Association for Computational Linguistics, pages 120–128. http://www.aclweb.org/anthology/W06-1615.

Konstantinos Bousmalis, George Trigeorgis, Nathan Silberman, Dilip Krishnan, and Dumitru Erhan. 2016. Domain separation networks. In *Advances in Neural Information Processing Systems*. pages 343–351.

Kai Cao, Xiang Li, Miao Fan, and Ralph Grishman. 2015a. Improving event detection with active learning. In *Proceedings of the International Conference Recent Advances in Natural Language Processing (RANLP'15)*. pages 72–77. http://www.aclweb.org/anthology/R15-1010.

Kai Cao, Xiang Li, and Ralph Grishman. 2015b. Improving event detection with dependency regularization. In *Proceedings of the International Conference Recent Advances in Natural Language Processing (RANLP'15)*. pages 78–83. http://www.aclweb.org/anthology/R15-1011.

Yubo Chen, Shulin Liu, Xiang Zhang, Kang Liu, and Jun Zhao. 2017. Automatically labeled data generation for large scale event extraction. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL'17)*. volume 1, pages 409–419. https://doi.org/10.18653/v1/P17-1038.

Yubo Chen, Liheng Xu, Kang Liu, Daojian Zeng, Jun Zhao, et al. 2015. Event extraction via dynamic multi-pooling convolutional neural networks. In *Proceedings of the 53th Annual Meeting of the Association for Computational Linguistics (ACL'15)*. pages 167–176. https://doi.org/10.3115/v1/P15-1017.

Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555* .

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning (ICML'08)*. ACM, pages 160–167.

George R Doddington, Alexis Mitchell, Mark A Przybocki, Lance A Ramshaw, Stephanie Strassel, and Ralph M Weischedel. 2004. The automatic content extraction (ACE) program-tasks, data, and evaluation. In *LREC*. volume 2, pages 1–4. http://www.aclweb.org/anthology/L04-1011.

Joe Ellis, Jeremy Getman, Dana Fore, Neil Kuster, Zhiyi Song, Ann Bies, and Stephanie Strassel. 2015. Overview of linguistic resources for the tac kbp 2015 evaluations: Methodologies and results. In *Proceedings of TAC KBP 2015 Workshop, National Institute of Standards and Technology (TAC'15)*. pages 16–17.

Xiaocheng Feng, Lifu Huang, Duyu Tang, Heng Ji, Bing Qin, and Ting Liu. 2016. A language-independent neural network for event detection. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL'16)*. volume 2, pages 66–71. https://doi.org/10.18653/v1/P16-2011.

Tao Ge, Lei Cui, Baobao Chang, Zhifang Sui, and Ming Zhou. 2016. Event detection with burst information networks. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*. pages 3276–3286.

Reza Ghaeini, Xiaoli Fern, Liang Huang, and Prasad Tadepalli. 2016. Event nugget detection with forward-backward recurrent neural networks. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL'16)*. volume 2, pages 369–373. https://doi.org/10.18653/v1/P16-2060.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *Advances in neural information processing systems*. pages 2672–2680.

Ralph Grishman, David Westbrook, and Adam Meyers. 2005. Nyu's English ACE 2005 system description. *ACE'05* .

Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R Salakhutdinov. 2012. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580* .

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 Text Analysis Conference (TAC'15)*.

Yu Hong, Xiaobin Wang, Yadong Chen, Jian Wang, Tongtao Zhang, Jin Zheng, Dian Yu, Qi Li, Boliang Zhang, Han Wang, et al. 2014. RPI BLENDER TAC-KBP2014 knowledge base population system. In *Proceedings of Text Analysis Conference (TAC'14)*.

Yu Hong, Jianfeng Zhang, Bin Ma, Jianmin Yao, Guodong Zhou, and Qiaoming Zhu. 2011. Using cross-entity inference to improve event extraction. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies (ACL-HLT'11)*. Association for Computational Linguistics, pages 1127–1136. http://www.aclweb.org/anthology/P11-1113.

Xun Huang, Yixuan Li, Omid Poursaeed, John Hopcroft, and Serge Belongie. 2017. Stacked generative adversarial networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. volume 2, page 4.

Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional LSTM-CRF models for sequence tagging. *arXiv preprint arXiv:1508.01991* .

Md M Islam, Xin Yao, and Kazuyuki Murase. 2003. A constructive algorithm for training cooperative neural network ensembles. *IEEE Transactions on neural networks* 14(4):820–834.

Rafal Jozefowicz, Wojciech Zaremba, and Ilya Sutskever. 2015. An empirical exploration of recurrent network architectures. In *Proceedings of the 32nd International Conference on Machine Learning (ICML'15)*. pages 2342–2350.

Alex M Lamb, Anirudh Goyal ALIAS PARTH GOYAL, Ying Zhang, Saizheng Zhang, Aaron C Courville, and Yoshua Bengio. 2016. Professor forcing: A new algorithm for training recurrent networks. In *Advances In Neural Information Processing Systems*. pages 4601–4609.

Qi Li, Heng Ji, and Liang Huang. 2013. Joint event extraction via structured prediction with global features. In *Proceedings of the 51th Annual Meeting of the Association for Computational Linguistics (ACL'13)*. pages 73–82. http://www.aclweb.org/anthology/P13-1008.

Qi Li, Heng Ji, HONG Yu, and Sujian Li. 2014. Constructing information networks using one single model. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP'14)*. pages 1846–1851. https://doi.org/10.3115/v1/D14-1198.

Shasha Liao and Ralph Grishman. 2010. Using document level cross-event inference to improve event extraction. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics (ACL'10)*. Association for Computational Linguistics, pages 789–797. http://www.aclweb.org/anthology/P10-1081.

Zhouhan Lin, Minwei Feng, Cicero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. 2017. A structured self-attentive sentence embedding. *arXiv preprint arXiv:1703.03130* .

Ming-Yu Liu and Oncel Tuzel. 2016. Coupled generative adversarial networks. In *Advances in neural information processing systems*. pages 469–477.

Pengfei Liu, Xipeng Qiu, Jifan Chen, and Xuanjing Huang. 2016a. Deep fusion LSTMs for text semantic matching. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL'16)*. volume 1, pages 1034–1043. https://doi.org/10.18653/v1/P16-1098.

Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. 2017a. Adversarial multi-task learning for text classification. *arXiv preprint arXiv:1704.05742* https://doi.org/10.18653/v1/P17-1001.

Shulin Liu, Yubo Chen, Shizhu He, Kang Liu, and Jun Zhao. 2016b. Leveraging framenet to improve automatic event detection. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL'16)*. https://doi.org/10.18653/v1/P16-1201.

Shulin Liu, Yubo Chen, Kang Liu, and Jun Zhao. 2017b. Exploiting argument information to improve event detection via supervised attention mechanisms 1:1789–1797. https://doi.org/10.18653/v1/P17-1164.

Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. 2013. Linguistic regularities in continuous space word representations. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL'13)*. volume 13, pages 746–751. http://www.aclweb.org/anthology/N13-1090.

Thien Huu Nguyen, Kyunghyun Cho, and Ralph Grishman. 2016. Joint event extraction via recurrent neural networks. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL'16)*. pages 300–309. https://doi.org/10.18653/v1/N16-1034.

Thien Huu Nguyen and Ralph Grishman. 2014. Employing word representations and regularization for domain adaptation of relation extraction. In *Proceedings of the 52th Annual Meeting of the Association for Computational Linguistics (ACL'14)*. pages 68–74. https://doi.org/10.3115/v1/P14-2012.

Thien Huu Nguyen and Ralph Grishman. 2015. Event detection and domain adaptation with convolutional neural networks. In *Proceedings of the 53th Annual Meeting of the Association for Computational Linguistics (ACL'15)*. pages 365–371. https://doi.org/10.3115/v1/P15-2060.

Thien Huu Nguyen and Ralph Grishman. 2016. Modeling skip-grams for event detection with convolutional neural networks. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP'16)*. pages 886–891. https://doi.org/10.18653/v1/D16-1085.

Siddharth Patwardhan and Ellen Riloff. 2009. A unified model of phrasal and sentential evidence for information extraction. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing (EMNLP'09)*. Association for Computational Linguistics, pages 151–160. http://www.aclweb.org/anthology/D09-1016.

Haoruo Peng, Yangqiu Song, and Dan Roth. 2016. Event detection and co-reference with minimal supervision. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP'16)*. pages 392–402. https://doi.org/10.18653/v1/D16-1038.

Barbara Plank and Alessandro Moschitti. 2013. Embedding semantic similarity in tree kernels for domain adaptation of relation extraction. In *Proceedings of the 51th Annual Meeting of the Association for Computational Linguistics (ACL'13)*. pages 1498–1507. http://www.aclweb.org/anthology/P13-1147.

Mark Sammons, Haoruo Peng, Yangqiu Song, Shyam Upadhyay, Chen-Tse Tsai, Pavankumar Reddy, Subhro Roy, and Dan Roth. 2015. Illinois CCG TAC 2015 event nugget, entity discovery and linking, and slot filler validation systems. In *Proceedings of Text Analytics Conference (TAC'15)*.

Mike Schuster and Kuldip K Paliwal. 1997. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing* 45(11):2673–2681.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*. pages 3104–3112.

Joseph Turian, Lev Ratinov, and Yoshua Bengio. 2010. Word representations: a simple and general method for semi-supervised learning. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics (ACL'10)*. Association for Computational Linguistics, pages 384–394. https://doi.org/http://www.aclweb.org/anthology/P10-1040.

Mengqiu Wang and Christopher D Manning. 2013. Effect of non-linear deep architecture in sequence labeling. In *Proceedings of the Sixth International Joint Conference on Natural Language Processing (IJCNLP'13)*. pages 1285–1291. https://doi.org/http://www.aclweb.org/anthology/I13-1183.

Sam Wei, Igor Korostil, Joel Nothman, and Ben Hachey. 2017. English event detection with translated language features. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL'17)*. volume 2, pages 293–298. https://doi.org/10.18653/v1/P17-2046.

Dian Yu, Xiaoman Pan, Boliang Zhang, Lifu Huang, Di Lu, Spencer Whitehead, and Heng Ji. 2016. RPI BLENDER TAC-KBP2016 system description. In *Proceedings of Text Analysis Conference (TAC'16)*.

Lantao Yu, Weinan Zhang, Jun Wang, and Yong Yu. 2017. Seqgan: Sequence generative adversarial nets with policy gradient. In *Proceedings of the 32nd AAAI Conference on Artificial Intelligence (AAAI'17)*. pages 2852–2858.

Matthew D Zeiler. 2012. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701* .

Yizhe Zhang, Zhe Gan, and Lawrence Carin. 2016. Generating text via adversarial training. In *NIPS workshop on Adversarial Training*. volume 21.