Exploring Pre-trained Language Models for Event Extraction and Generation

Sen Yang[†], **Dawei Feng**[†], **Linbo Qiao, Zhigang Kan, Dongsheng Li**[‡]
National University of Defense Technology, Changsha, China
{sen_yang,linbo.qiao,kanzhigang13}@nudt.edu.cn
davyfeng.c@gmail.com, lds1201@163.com

Abstract

Traditional approaches to the task of ACE event extraction usually depend on manually annotated data, which is often laborious to create and limited in size. Therefore, in addition to the difficulty of event extraction itself, insufficient training data hinders the learning process as well. To promote event extraction, we first propose an event extraction model to overcome the roles overlap problem by separating the argument prediction in terms of roles. Moreover, to address the problem of insufficient training data, we propose a method to automatically generate labeled data by editing prototypes and screen out generated samples by ranking the quality. Experiments on the ACE2005 dataset demonstrate that our extraction model can surpass most existing extraction methods. Besides, incorporating our generation method exhibits further significant improvement. It obtains new state-of-the-art results on the event extraction task, including pushing the F1 score of trigger classification to 81.1%, and the F1 score of argument classification to 58.9%.

1 Introduction

Event extraction is a key and challenging task for many NLP applications. It targets to detect event trigger and arguments. Figure 1 illustrates a sentence containing an event of type *Meet* triggered by *"meeting"*, with two arguments: "*President Bush*" and "*several Arab leaders*", both of which play the role "*Entity*".

There are two interesting issues in event extraction that require more efforts. On the one hand, roles in an event vary greatly in frequency (Figure 2), and they can overlap on some words,

Event type: *Meet*
Sentence : ***President Bush***[Entity] is going to be ***meeting***[Trigger] with ***several Arab leaders***[Entity]

Figure 1: An event of type *Meet* is highlighted in the sentence, including one trigger and two arguments.

even sharing the same argument (**the roles overlap problem**). For example, in sentence *"The explosion killed the bomber and three shoppers"*, *"killed"* triggers an *Attack* event, while argument *"the bomber"* plays the role *"Attacker"* as well as the role *"Victim"* at the same time. There are about 10% events in the ACE2005 dataset (Doddington et al., 2004) having the roles overlap problem. However, despite the evidence of the roles overlap problem, few attentions have been paid to it. On the contrary, it is often simplified in evaluation settings of many approaches. For example, in most previous works, if an argument plays multiple roles in an event simultaneously, the model classifies correctly as long as the prediction hits any one of them, which is obviously far from accurate to apply to the real world. Therefore, we design an effective mechanism to solve this problem and adopt more rigorous evaluation criteria in experiments.

On the other hand, so far most deep learning based methods for event extraction follow the supervised-learning paradigm, which requires lots of labeled data for training. However, annotating accurately large amounts of data is a very laborious task. To alleviate the suffering of existing methods from the deficiency of predefined event data, event generation approaches are often used to produce additional events for training (Yang et al., 2018; Zeng et al., 2018; Chen et al., 2017). And distant supervision (Mintz et al., 2009) is a commonly used technique to this end for labeling external corpus. But the quality and quantity

[†]These two authors contributed equally.
[‡]Corresponding Author.

Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, pages 5284–5294
Florence, Italy, July 28 - August 2, 2019.

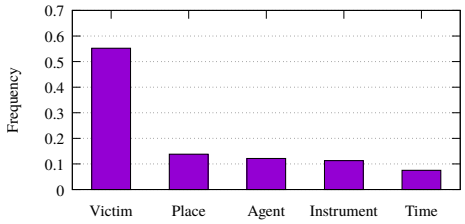

Figure 2: Frequency of roles that appear in events of type *Injure* in the ACE2005 dataset.

of events generated with distant supervision are highly dependent on the source data. In fact, external corpus can also be exploited by pre-trained language models to generate sentences. Therefore, we turn to pre-trained language models, attempting to leverage their knowledge learned from the large-scale corpus for event generation.

Specifically, this paper proposes a framework based on pre-trained language models, which includes an event extraction model as our baseline and a labeled event generation method. Our proposed event extraction model is constituted of a trigger extractor and an argument extractor which refers result of the former for inference. In addition, we improve the performance of the argument extractor by re-weighting the loss function based on the importance of roles.

Pre-trained language models have also been applied to generating labeled data. Inspired by the work of Guu et al. (2018), we take the existing samples as prototypes for event generation, which contains two key steps: argument replacement and adjunct token rewriting. Through scoring the quality of generated samples, we can pick out those of high quality. Incorporating them with existing data can further improve the performance of our event extractor.

2 Related work

Event Extraction In terms of analysis granularity, there are document-level event extraction (Yang et al., 2018) and sentence-level event extraction (Zeng et al., 2018). We focus on the statistical methods of the latter in this paper. These methods can be further divided into two detailed categories: the feature based ones (Liao and Grishman, 2010; Liu et al., 2010; Miwa et al., 2009; Liu et al., 2016; Hong et al., 2011; Li et al., 2013b) which track designed features for extraction, and the neural based ones that take advantage of neural networks to learn features automatically (Chen et al., 2015; Nguyen and Grishman, 2015; Feng et al., 2016).

Event Generation External resources such as Freebase, Frame-Net and WordNet are commonly employed to generate event and enrich the training data. Several previous event generation approaches (Chen et al., 2017; Zeng et al., 2018) base a strong assumption in distant supervision[1] to label events in unsupervised corpus. But in fact, co-occurring entities could have none expected relationship. In addition, Huang et al. (2016) incorporates abstract meaning representation and distribution semantics to extract events. While Liu et al. (2016, 2017) manages to mine additional events from the frames in FrameNet.

Pre-trained Language Model Pre-trained language models are capable of capturing the meaning of words dynamically in consideration of their context. McCann et al. (2017) exploits language model pre-trained on supervised translation corpus in the target task. ELMO (Embeddings from Language Models) (Peters et al., 2018) gets context sensitive embeddings by encoding characters with stacked bidirectional LSTM (Long Short Term Memory) and residual structure (He et al., 2016). Howard and Ruder (2018) obtains comparable result on text classification. GPT (Generative Pre-Training) (Radford et al., 2018) improves the state of the art in 9 of 12 tasks. BERT (Bidirectional Encoder Representations from Transformers) (Devlin et al., 2018) breaks records of 11 NLP task and received a lot of attention.

3 Extraction Model

This section describes our approach to extract events that occur in plain text. We consider event extraction as a two-stage task, which includes trigger extraction and argument extraction, and propose a **P**re-trained **L**anguage **M**odel based **E**vent **E**xtractor (**PLMEE**). Figure 3 illustrates the architecture of PLMEE. It consists of a trigger extractor and an argument extractor, both of which rely on the feature representation of BERT.

3.1 Trigger Extractor

Trigger extractor targets to predict whether a token triggers an event. So we formulate trigger extraction as a token-level classification task with labels

[1] If two entities have a relationship in a knowledge base, then all sentences that mention these two entities will express that relationship.

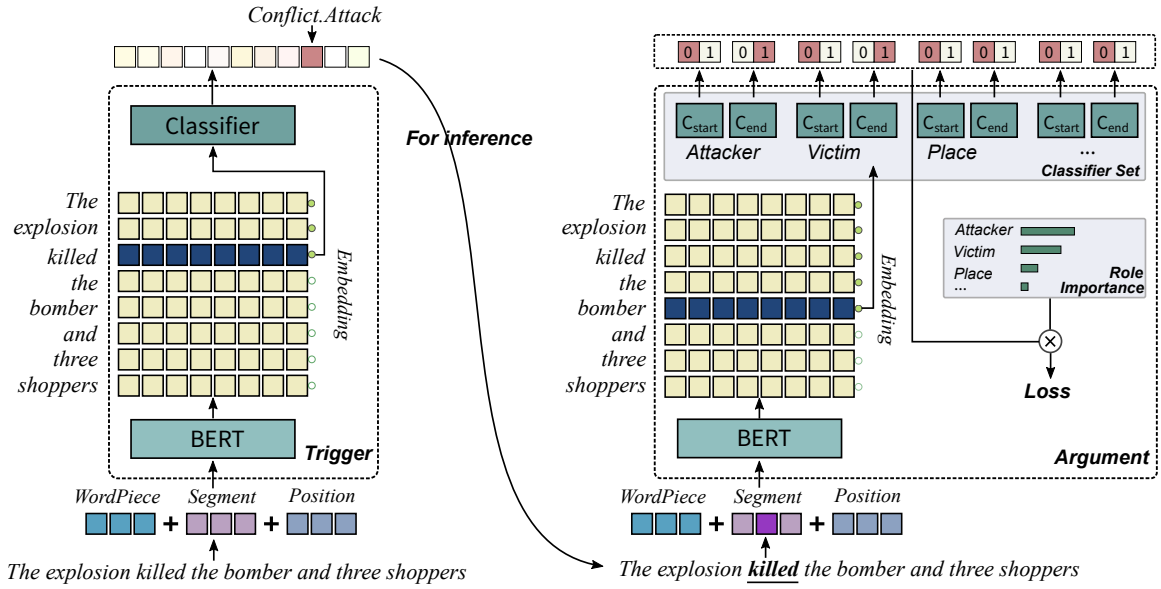

Figure 3: Illustration of the PLMEE architecture, including a trigger extractor and an argument extractor. The processing procedure of an event instance triggered by the word "killed" is also shown.

being event types, and just add a multi-classifier on BERT to build the trigger extractor.

The input of the trigger extractor follows the BERT, i.e. the sum of three types of embeddings, including WordPiece embedding (Wu et al., 2016), position embedding and segment embedding. Since the input contains only one sentence, all its segment ids are set to zero. In addition, token [CLS] and [SEP][2] are placed at the start and end of the sentence.

In many cases, the trigger is a phrase. Therefore, we treat consecutive tokens which share the same predicted label as a whole trigger. As general, we adopt cross entropy as the loss function for fine-tuning.

3.2 Argument Extractor

Given the trigger, argument extractor aims to extract related arguments and all roles they play. Compared with trigger extraction, argument extraction is more complicated because of three issues: the dependency of arguments on the trigger, most arguments being long noun phrases, and the roles overlap problem. We take exactly a series of actions to deal with these obstacles.

In common with trigger extractor, argument extractor requires three kinds of embeddings as well. However, it needs to know which tokens comprise the trigger. Therefore, we feed argument extractor with the segment ids of trigger tokens being one.

[2][CLS], [SEP] and [MASK] are special tokens of BERT.

To overcome the latter two issues in argument extraction, we add multiple sets of binary classifiers on the BERT. Each set of classifiers sever for a role to determine the spans (each span includes a start and an end) of all arguments that play it. This approach is similar to the question answering task on the SQuAD (Rajpurkar et al., 2016) in which there is only one answer, while multiple arguments playing the same role can appear simultaneously in an event. Since the prediction is separated with roles, an argument can play multiple roles, and a token can belong to different arguments. Thus, the roles overlap problem can also be solved.

3.3 Argument Span Determination

In PLMEE, a token t is predicted as the start of an argument that plays role r with probability:

$$P_s^r(t) = Softmax\left(W_s^r \cdot \mathcal{B}(t)\right),$$

while as the end with probability:

$$P_e^r(t) = Softmax\left(W_e^r \cdot \mathcal{B}(t)\right),$$

in which we use subscript "s" to represent "start" and subscript "e" to represent "end". W_s^r is the weight of binary classifier that aims to detect starts of arguments playing role r, while W_e^r is the weight of another binary classifier that aims to detect ends. $\mathcal{B}$ is the BERT embedding.

For each role r, we can get two lists B_s^r and B_e^r of 0 and 1 according to P_s^r and P_e^r. They indicate respectively whether a token in the sentence is the

start or end of an argument that plays role r[3]. Algorithm 1 is used to detect each token sequentially to determine spans of all arguments that play the role r.

Algorithm 1 Argument span determination

In: P_s^r and P_e^r, B_s^r and B_e^r, sentence length l.
Out: Span list L of the arguments that play role r
Initiate: $a_s \leftarrow$-1, $a_e \leftarrow$-1

for $i \leftarrow 0$ to l **do**
  if In State 1 **&** the i^{th} token is a start **then**
    $a_s \leftarrow i$ and change to State 2
  end if
  if In State 2 **then**
    if the i^{th} token is a new start **then**
      $a_s \leftarrow i$ **if** $P_s^r[i] > P_s^r[a_s]$
    end if
    if the i^{th} token is an end **then**
      $a_e \leftarrow i$ and change to State 3
    end if
  end if
  if In State 3 **then**
    if the i^{th} token is a new end **then**
      $a_e \leftarrow i$ **if** $P_e^r[i] > P_e^r[a_e]$
    end if
    if the i^{th} token is a new start **then**
      Append $[a_s, a_e]$ to L
      $a_e \leftarrow$ -1, $a_s \leftarrow i$ and change to State 2
    end if
  end if
end for

Algorithm 1 contains a finite state machine, which changes from one state to another in response to B_s^r and B_e^r. There are three states totally: 1) Neither start nor end has been detected; 2) Only a start has been detected; 3) A start as well as an end have been detected. Specially, the state changes according to the following rules: State 1 changes to State 2 when the current token is a start; State 2 changes to State 3 when the current token is an end; State 3 changes to State 2 when the current token is a new start. Notably, if there has been a start and another start arises, we will choose the one with higher probability, and the same for end.

3.4 Loss Re-weighting

We initially define $\mathcal{L}_s$ as the loss function of all binary classifiers that are responsible for detecting starts of arguments. It is the average of cross entropy between the output probabilities and the golden label $\boldsymbol{y}$:

$$\mathcal{L}_s = \frac{1}{|\mathcal{R}| \times |\mathcal{S}|} \sum_{r \in \mathcal{R}} \text{CE}\left(P_s^r, \boldsymbol{y}_s^r\right),$$

in which CE is cross entropy, $\mathcal{R}$ is the set of roles, $\mathcal{S}$ is the input sentence, and $|\mathcal{S}|$ is the number of tokens in $\mathcal{S}$. Similarly, we define $\mathcal{L}_e$ as the loss function of all binary classifiers that detect ends:

$$\mathcal{L}_e = \frac{1}{|\mathcal{R}| \times |\mathcal{S}|} \sum_{r \in \mathcal{R}} \text{CE}\left(P_e^r, \boldsymbol{y}_e^r\right).$$

We finally average $\mathcal{L}_s$ and $\mathcal{L}_e$ as the loss $\mathcal{L}$ of argument extractor.

As Figure 2 shows, there exists a big gap in frequency between roles. This implies that roles have different levels of "importance" in an event. The "importance" here means the ability of a role to indicate events of a specific type. For example, the role "*Victim*" is more likely to indicate a *Die* event than the role "*Time*". Inspired by this, we re-weight $\mathcal{L}_s$ and $\mathcal{L}_e$ according to the importance of roles, and propose to measure the importance with the following definitions:

Role Frequency (RF) We define RF as the frequency of role r appearing in events of type v:

$$\text{RF}(r, v) = \frac{N_v^r}{\sum_{k \in \mathcal{R}} N_v^k},$$

where N_v^r is the count of the role r that appear in the events of type v.

Inverse Event Frequency (IEF) As the measure of the universal importance of a role, we define IEF as the logarithmically scaled inverse fraction of the event types that contain the role r:

$$\text{IEF}(r) = \log \frac{|\mathcal{V}|}{|\{v \in \mathcal{V} : r \in v\}|},$$

where V is tht set of event types.

Finally we take RF-IEF as the product of RF and IEF: $\text{RF-IEF}(r, v) = \text{RF}(r, v) \times \text{IEF}(r)$. With RF-IEF, we can measure the importance of a role r in events of type v:

$$I(r, v) = \frac{\exp^{\text{RF-IEF}(r,v)}}{\sum_{r' \in R} \exp^{\text{RF-IEF}(r',v)}}.$$

We choose three event types and list the two most important roles of each type in Table 1. It shows that although there could be multiple roles

[3]The i^{th} token is a start if $B_s^r[i]$=1 or an end if $B_e^r[i]$=1.

Event Type	Top 2 Roles	Sum
Transport(15)	Artifact, Origin	0.76
Attack(14)	Attacker, Target	0.85
Die(12)	Victim, Agent	0.90

Table 1: Top two roles and their sum importance for each event type. The number in brackets behind event type is the count of roles that have appeared in it.

in events of someone type, only a few of them is indispensable.

Give the event type v of input, we re-weight $\mathcal{L}_s$ and $\mathcal{L}_e$ based on each role's importance in v:

$$\mathcal{L}_s = \sum_{r \in \mathcal{R}} \frac{I(r,v)}{|\mathcal{S}|} \mathrm{CE}\left(P_s^r, \boldsymbol{y}_s^r\right)$$

$$\mathcal{L}_e = \sum_{r \in \mathcal{R}} \frac{I(r,v)}{|\mathcal{S}|} \mathrm{CE}\left(P_e^r, \boldsymbol{y}_e^r\right).$$

The loss of argument extractor $\mathcal{L}$ is still the average of $\mathcal{L}_s$ and $\mathcal{L}_e$.

4 Training Data Generation

In addition to PLMEE, we also propose a pre-trained language model based method for event generation as illustrated in Figure 4. By editing prototypes, this method can generate a controllable number of labeled samples as the extra training corpus. It consists of three stages: pre-processing, event generation and scoring.

To facilitate the generation method, we define adjunct tokens as the tokens in sentences except triggers and arguments, including not only words and numbers, but also punctuation. Taking sentence in Figure 1 as an example, "*is*" and "*going*" are adjunct tokens. It is evident that adjunct tokens can adjust the smooth and diversity of expression. Therefore, we try to rewrite them to expand the diversity of the generation results, while keeping the trigger and arguments unchanged.

4.1 Pre-processing

With the golden labels, we first collect arguments in the ACE2005 dataset as well as the roles they play. However, those arguments overlap with others are excluded. Because such arguments are often long compound phrases that contain too much unexpected information, and incorporating them in argument replacement could bring more unnecessary errors.

We also adopt BERT as the target model to rewrite adjunct tokens in the following stage, and fine-tune it on the ACE2005 dataset with the masked language model task (Devlin et al., 2018) to bias its prediction towards the dataset distribution. In common with the pre-training procedure of BERT, each time we sample a batch of sentences and mask 15% of tokens. Its goal is still to predict the correct token without supervision.

4.2 Event generation

To generate events, we conduct two steps on a prototype. We first replace the arguments in the prototype with those similar that have played the same role. Next, we rewrite adjunct tokens with the fine-tuned BERT. Through these two steps, we can obtain a new sentence with annotations.

Argument Replacement The first step is to replace arguments in the event. Both the argument to be replaced and the new one should have played ever the same role. While the roles are inherited after replacement, so we can still use origin labels for the generated samples.

In order not to change the meaning drastically, we employ similarity as the criteria for selecting new arguments. It is based on the following two considerations: one is that two arguments that play the same role may diverge significantly in semantics; another is that the role an argument plays is largely dependent on its context. Therefore, we should choose arguments that are semantically similar and coherent with the context.

We use cosine similarity between embeddings to measure the similarity of two arguments. And due to ELMO's ability to handle the OOV problem, we employ it to embed arguments:

$$\mathcal{E}(a) = \frac{1}{|a|} \sum_{t \in a} \mathcal{E}(t),$$

where a is the argument, $\mathcal{E}$ is ELMO embedding. We choose the top 10 percent most similar arguments as candidates, and use softmax operation on their similarity to allocate probability.

An argument is replaced with probability 80% while keeping constant with probability 20% to bias the representation towards the actual event (Devlin et al., 2018). Note that the triggers remain unchanged to avoid undesirable deviation of dependency relation.

Adjunct Token Rewriting The results of argument replacement can already be considered as the generated data, but the constant context may increase the risk of overfitting. Therefore, to smooth

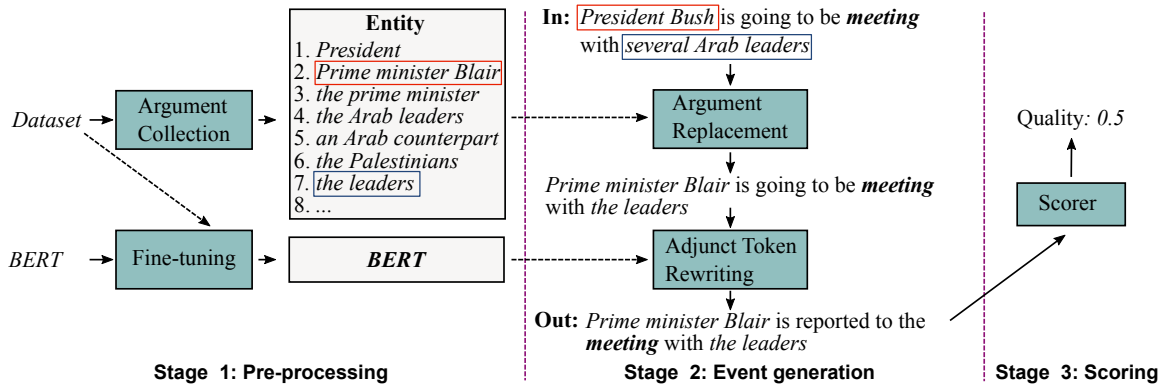

Figure 4: Flow chart of the generation approach.

the generated data and expand their diversity, we manage to rewrite adjunct tokens with the fine-tuned BERT.

The rewriting is to replace some adjunct tokens in the prototype with the new ones that are more matchable with the current context. We take it as a Cloze task (Taylor, 1953), where some adjunct tokens are randomly masked and the BERT fine tuned in the first stage is used to predict vocabulary ids of suitable tokens based on the context. We use a parameter m to denote the proportion of adjunct tokens that need to be rewritten.

Adjunct token rewriting is a step-by-step process. Each time we mask 15% of adjunct tokens (with the token [MASK]). Then the sentence is fed into BERT to produce new adjunct tokens. The adjunct tokens that have not yet been rewritten will temporarily remain in the sentence.

To further illustrate the above two steps, we give an instance in Figure 4. In this instance, we set m to 1.0, which means all the adjunct tokens will be rewritten. The final output is "*Prime minister Blair is reported to the meeting with the leaders*", which shares the labels with the original event in the prototype. It is evident that some adjunct tokens are preserved despite m is 1.0.

4.3 Scoring

Theoretically, infinite number of events can be generated with our generation method. However, not all of them are valuable for the extractor and some may even degrade its performance. Therefore, we add an extra stage to quantify the quality of each generated sample to pick out those valuable. Our key insight for evaluating the quality lies that it is tightly related to two factors, which are the perplexity and the distance to the original dataset. The former reflects the rationality of generation, and the latter reflects the differences between the data.

Perplexity (PPL) Different with the masked perplexity (Devlin et al., 2018) of logarithmic version, we take the average probability of those adjunct tokens that have been rewritten as the perplexity of generated sentence $\mathcal{S}'$:

$$\mathrm{PPL}(\mathcal{S}') = \frac{1}{|A(\mathcal{S}')|} \sum_{t \in A(\mathcal{S}')} P(t),$$

where A is the set of adjunct tokens in $\mathcal{S}'$ that have been rewritten.

Distance (DIS) We measure the distance between $\mathcal{S}'$ and the dataset $\mathcal{D}$ with cosine similarity:

$$\mathrm{DIS}(\mathcal{S}', \mathcal{D}) = 1 - \frac{1}{|\mathcal{D}|} \sum_{\mathcal{S} \in \mathcal{D}} \frac{\mathcal{B}(\mathcal{S}') \cdot \mathcal{B}(\mathcal{S})}{|\mathcal{B}(\mathcal{S}')| \times |\mathcal{B}(\mathcal{S})|}.$$

Different with embedding arguments by ELMO, we utilize BERT to embed sentence and take the embedding of the first token [CLS] as the sentence embedding.

Both the PPL and the DIS are limited in [0,1]. We consider that generated samples of high quality should have both low PPL and DIS. Therefore, we define the quality function as:

$$Q(\mathcal{S}') = 1 - \left(\lambda \mathrm{PPL}\left(\mathcal{S}'\right) + (1-\lambda)\,\mathrm{DIS}\left(\mathcal{S}', \mathcal{D}\right)\right)$$

, where $\lambda \in [0, 1]$ is the balancing parameter. This function is used to select generated samples of high quality in experiments.

5 Experiments

In this section, we first evaluate our event extractor PLMEE on the ACE2005 dataset. Then we give a case study of generated samples and conduct automatic evaluations by adding them into the training set. Finally, we illustrate the limitations of the generation method.

Model \ Phase	Trigger Identification(%)			Trigger Calssfication(%)			Argument Identification(%)			Argument Calssfication(%)		
	P	R	F	P	R	F	P	R	F	P	R	F
Cross Event		N/A		68.7	68.9	68.8	50.9	49.7	50.3	45.1	44.1	44.6
Cross Entity		N/A		72.9	64.3	68.3	53.4	52.9	53.1	51.6	45.5	48.3
Max Entropy	76.9	65.0	70.4	73.7	62.3	67.5	69.8	47.9	56.8	64.7	44.4	52.7
DMCNN	80.4	67.7	73.5	75.6	63.6	69.1	68.8	51.9	59.1	62.2	46.9	53.5
JRNN	68.5	75.7	71.9	66.0	73.0	69.3	61.4	64.2	62.8	54.2	56.7	55.4
DMCNN-DS	79.7	69.6	74.3	75.7	66.0	70.5	71.4	56.9	63.3	62.8	50.1	55.7
ANN-FN		N/A		79.5	60.7	68.8		N/A			N/A	
ANN-AugATT		N/A		78.0	66.3	71.7		N/A			N/A	
PLMEE(-)	84.8	83.7	**84.2**	81.0	80.4	**80.7**	71.5	59.2	64.7	61.7	53.9	57.5
PLMEE							71.4	60.1	**65.3**	62.3	54.2	**58.0**

Table 2: Performance of all methods. Bold denotes the best result.

As previous works (Li et al., 2013b; Chen et al., 2015; Hong et al., 2011), we take the test set with 40 newswire documents, while 30 other documents as the validation set, and the remaining 529 documents to be the training set. However, different with previous works, we take the following criteria to evaluate the correctness of each predicted event mention:

1. A trigger prediction is correct only if its span and type match with the golden labels.

2. An argument prediction is correct only if its span and all roles it plays match with the golden labels.

It is worth noting that all the predicted roles for an argument are required to match with the golden labels, instead of just one of them. We adopt *Precision (P)*, *Recall (R)* and *F measure (F1)* as the evaluation metrics.

5.1 Results of Event Extraction

We take several previous classic works for comparison, and divide them into three categories:

Feature based methods Document-level information is utilized in *Cross event* (Liao and Grishman, 2010) to assist event extraction. While *Cross entity* (Hong et al., 2011) uses cross-entity inference in extraction. *Max Extropy* (Li et al., 2013a) extracts triggers as well as arguments together based on structured prediction.

Neural based methods *DMCNN* (Chen et al., 2015) adopts firstly dynamic multi-pooling CNN to extract sentence-level features automatically. *JRNN* (Nguyen et al., 2016) proposes a joint framework based on bidirectional RNN for event extraction.

External resource based methods *DMCNN-DS* (Chen et al., 2017) uses FreeBase to label potential events in unsupervised corpus by distance supervision. *ANN-FN* (Liu et al., 2016) improves extraction with additionally events automatically detected from FrameNet, while *ANN-AugATT* (Liu et al., 2017) exploits argument information via the supervised attention mechanisms to improve the performance further.

In order to verify the effectiveness of loss re-weighting, two groups of experiments are conducted for comparison. Namely, the group where the loss function is simply averaged on all classifiers' output (indicated as **PLMEE(-)**) and the group where the loss is re-weighted based on role importance (indicated as **PLMEE**).

Table 2 compares the results of the aforementioned models with PLMEE on the test set. As is shown, in both the trigger extraction task and the argument extraction task, PLMEE(-) has achieved the best results among all the compared methods. The improvement on the trigger extraction is quite significant, seeing a sharp increase of near 10% on the F1 score. While the improvement in argument extraction is not so obvious, achieving about 2%. This is probably due to the more rigorous evaluation metric we have taken and the difficulty of argument extraction task as well. Moreover, compared with feature based methods, neural based methods can achieve better performance. And the same observation appears when comparing external resource based methods with neural based methods. It demonstrates that external re-

Prototype	m	Generated Event
President Bush is going to be **meeting** with *several Arab leaders*	0.2	*Russian President Putin* is going to the **meeting** with *the Arab leaders*
	0.4	*The president* is reported to be **meeting** with *an Arab counterpart*
	0.6	*Mr. Bush* is summoned to a **meeting** with *some Shiite Muslim groups*
	0.8	*The president* is attending to the **meeting** with *the Palestinians*
	1.0	*Prime minister Blair* is reported to the **meeting** with *the leaders*

Table 3: Example samples generated with different proportion of rewritten adjunct tokens. Italic indicates argument and bold indicates trigger.

sources are useful to improve event extraction. In addition, the PLMEE model can achieve better results on the argument extraction task - with improvement of 0.6% on F1 score for identification and 0.5% for classification - than the PLMEE(-) model, which means that re-weighting the loss can effectively improve the performance.

5.2 Case Study

Table 3 illustrates a prototype and its generation with parameter m ranging from 0.2 to 1.0. We can observe that the arguments after replacement can match the context in prototype relatively well, which indicates that they are resembling with the original ones in semantic.

On the other hand, rewriting the adjunct tokens can smooth the generated data and expand their diversity. However, since there is no explicit guide, this step can also introduce unpredictable noise, making the generation not fluent as expected.

5.3 Automatic Evaluation of Generation

So far, there are mainly three aspects of the generation method that could have significant impacts on the performance of the extraction model, including the amount of generated samples (represented by n, which indicates times the generation size is the number of dataset size), the proportion of rewritten adjunct tokens m, and the quality of the generated samples. The former two factors are controllable in the generation process. Specially, we can reuse a prototype and get a variety of combinations of arguments via similarity based replacement, which will bring different contexts for rewriting adjunct tokens. Moreover, the proportion of rewritten adjunct tokens can be adjusted, making a further variation. Although the quality of generation cannot be controlled arbitrarily, it can be quantified by the score function Q so that those samples of higher quality can be picked out and added into the training set. With λ in Q changing, different selection strategies can be used to screen out the generated samples.

We first tuned the former two parameters on the development set through grid search. Specially, we set m ranging from 0.2 to 1.0 with an interval of 0.2, and set n to be 0.5, 1.0 and 2.0, while keeping other parameters unchanged in the generation process. We conduct experiments with these parameters. By analyzing the results, we find that the best performance of PLMEE on both trigger extraction and argument extraction can be achieved with $m = 0.4$ and $n = 1.0$. It suggests that neither too few generated samples nor too much is a better choice for extraction. Too few has limited influence, while too much could bring more noise that disturbs the distribution of the dataset. For the better extraction performance, we use such parameter settings in the following experiments.

We also investigate the effectiveness of the sample selection approach, a comparison is conducted between three groups with different selection strategies. We obtain a total of four times the size of the ACE2005 dataset using our generation method with $m = 0.4$, and pick out one quarter of them ($n = 1.0$) with λ being 0, 0.5 and 1.0 respectively. When λ is 0 or 1.0, it is either perplexity or distance that determines the quality exclusively. We find that the selection method with $\lambda = 0.5$ in quality function is able to pick out samples that are more advantageous to promote the extraction performance.

Model	Trigger(%)	Argument(%)
PLMEE	80.7	58.0
PLMEE(+)	**81.1**	**58.9**

Table 4: F1 score of trigger classification and argument classification on the test set.

Finally, we incorporate the above generated data with the ACE2005 dataset and investigate the effectiveness of our generation method on the test

set. In Table 4, we use PLMEE(+) denotes the PLMEE model trained with extra generated samples. The results illustrate that with our event generation method, the PLMEE model can achieve the state of the art result of event extraction.

5.4 Limitation

By comparing the annotations in generated samples and manually labeled samples, we find that one issue of our generation method is that the roles may deviate, because the semantics could change a lot with only a few adjunct tokens been rewritten. Taking Figure 5 as an example. The roles played by argument "*Pittsburgh*" and "*Boston*" should be "*Destination*" and "*Origin*", rather not the opposite as in the prototype. This is because the token "*from*" has been replaced with the token "*for*", while token "*drive to*" been replaced with "*return from*".

Prototype: Leave from Niagara Falls and drive to Toronto, on 85 miles		
Trigger	leave	
Event type	*Movement.Transport*	
Arguments	Niagara Falls	Toronto
Roles	*Origin*	*Destination*
Generation: Leave for Pittsburgh and return from Boston in 200 miles		
Trigger	leave ✓	
Event type	*Movement.Transport*	
Arguments	Pittsburgh	Boston
Roles	*Origin* ✗	*Destination* ✗

Figure 5: One of the generated samples with wrong annotations.

6 Conclusion and Discussion

In this paper, we present a framework to promote event extraction by using a combination of an extraction model and a generation method, both of which are based on pre-trained language models. To solve the roles overlap problem, our extraction approach tries to separate the argument predictions in terms of roles. Then it exploits the importance of roles to re-weight the loss function. To perform event generation, we present a novel method that takes the existing events as prototypes. This event generation method can produce controllably labeled samples through argument replacement and adjunct tokens rewriting. It also benefits from the scoring mechanism which is able to quantify the quality of generated samples. Experimental results show that the quality of generated data is competitive and incorporating them with existing corpus can make our proposed event extractor to be superior to several state of the art approaches.

On the other hand, there are still limitations in our work. Events of the same type often share similarity. And co-occurring roles tend to hold a tight relation. Such features are ignored in our model, but they deserve more investigation for improving the extraction model. In addition, although our generation method can control the number of generated samples and filter with quality, it still suffers the deviation of roles alike with distant supervision. Therefore, for the future work, we will incorporate relation between events and relation between arguments into pre-trained language models, and take effective measures to overcome the deviation problem of roles in the generation.

Acknowledgments

The work was sponsored by the National Key Research and Development Program of China under Grant No.2018YFB0204300, and National Natural Science Foundation of China under Grant No.61872376 and No.61806216.

References

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 (Volume 1: Long Papers)*, volume 1, pages 409–419.

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

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

George R Doddington, Alexis Mitchell, Mark A Przybocki, Lance A Ramshaw, Stephanie M Strassel, and Ralph M Weischedel. 2004. The automatic content extraction (ace) program-tasks, data, and evaluation. In *LREC*, volume 2, page 1.

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 (Volume 2: Short Papers)*, volume 2, pages 66–71.

Kelvin Guu, Tatsunori B Hashimoto, Yonatan Oren, and Percy Liang. 2018. Generating sentences by editing prototypes. *Transactions of the Association of Computational Linguistics*, 6:437–450.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

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-Volume 1*, pages 1127–1136. Association for Computational Linguistics.

Jeremy Howard and Sebastian Ruder. 2018. Universal language model fine-tuning for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 328–339.

Lifu Huang, Taylor Cassidy, Xiaocheng Feng, Heng Ji, Clare R Voss, Jiawei Han, and Avirup Sil. 2016. Liberal event extraction and event schema induction. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 258–268.

Peifeng Li, Qiaoming Zhu, and Guodong Zhou. 2013a. Joint modeling of argument identification and role determination in chinese event extraction with discourse-level information. In *Twenty-Third International Joint Conference on Artificial Intelligence*.

Qi Li, Heng Ji, and Liang Huang. 2013b. Joint event extraction via structured prediction with global features. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 73–82.

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*, pages 789–797. Association for Computational Linguistics.

Bing Liu, Longhua Qian, Hongling Wang, and Guodong Zhou. 2010. Dependency-driven feature-based learning for extracting protein-protein interactions from biomedical text. In *Proceedings of the 23rd International Conference on Computational Linguistics: Posters*, pages 757–765. Association for Computational Linguistics.

Shulin Liu, Yubo Chen, Shizhu He, Kang Liu, and Jun Zhao. 2016. Leveraging framenet to improve automatic event detection. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 2134–2143.

Shulin Liu, Yubo Chen, Kang Liu, and Jun Zhao. 2017. Exploiting argument information to improve event detection via supervised attention mechanisms. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1789–1798.

Bryan McCann, James Bradbury, Caiming Xiong, and Richard Socher. 2017. Learned in translation: Contextualized word vectors. In *Advances in Neural Information Processing Systems*, pages 6294–6305.

Mike Mintz, Steven Bills, Rion Snow, and Dan Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2-Volume 2*, pages 1003–1011. Association for Computational Linguistics.

Makoto Miwa, Rune Sætre, Yusuke Miyao, and Jun'ichi Tsujii. 2009. A rich feature vector for protein-protein interaction extraction from multiple corpora. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 1-Volume 1*, pages 121–130. Association for Computational Linguistics.

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*, pages 300–309.

Thien Huu Nguyen and Ralph Grishman. 2015. Event detection and domain adaptation with convolutional neural networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, volume 2, pages 365–371.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training. *URL https://s3-us-west-2. amazonaws. com/openai-assets/research-covers/languageunsupervised/language understanding paper. pdf*.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392.

Wilson L Taylor. 1953. "cloze procedure": A new tool for measuring readability. *Journalism Bulletin*, 30(4):415–433.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*.

Hang Yang, Yubo Chen, Kang Liu, Yang Xiao, and Jun Zhao. 2018. Dcfee: A document-level chinese financial event extraction system based on automatically labeled training data. *Proceedings of ACL 2018, System Demonstrations*, pages 50–55.

Ying Zeng, Yansong Feng, Rong Ma, Zheng Wang, Rui Yan, Chongde Shi, and Dongyan Zhao. 2018. Scale up event extraction learning via automatic training data generation. In *Thirty-Second AAAI Conference on Artificial Intelligence*.