WIP Event Detection System at TAC KBP 2016 Event Nugget Track

Ying Zeng and **Bingfeng Luo** and **Yansong Feng** and **Dongyan Zhao**
Institute of Computer Science and Technology, Peking University
{ying.zeng, bf_luo, fengyansong, zhaody}@pku.edu.cn

Abstract

Event detection aims to extract events with specific types from unstructured data, which is the crucial and challenging task in event related applications, such as event coreference resolution and event argument extraction. In this paper, we propose an event detection system that combines traditional feature-based methods and novel neural network (NN) models. Experiments show that our ensemble approaches can achieve promising performance in the Event Nugget Detection task.

1 Introduction

Event detection, also called trigger labelling, aims to identify the mentions of some predefined event types. In this paper, we focus on the event extraction task proposed by TAC KBP 2016 competition (Song et al., 2016). An event nugget, as defined by the competition annotation guidelines, includes a word or a phrase of multiple words that instantiates an event, a classification of event type and subtype, and an indication of the REALIS value (ACTUAL, GENERIC, or OTHER) of the event. Below are some examples of event nuggets. The words underlined and in **bold face** are event nuggets that represent a single event.

- Hillary Clinton was not **elected** president in 2008. [Personnel_Elect, OTHER]
- The police investigated the **murder incident**.[1] [Conflict_Attack, ACTUAL]
- Correa was even accused without any evidences of **murder**.[2] [Conflict_Attack, OTHER; Life_Die, OTHER]
- Kennedy was **shot** **dead** by Oswald.[3] [Conflict_Attack, ACTUAL], [Life_Die, ACTUAL]

In the remaining parts of this paper, we first provide an overview of our system in Section 2. The following three sections describe the models we proposed in each subtask in detail. Section 6 discusses the experimental results, and Section 7 concludes the paper.

2 System Overview

Existing event extraction approaches can be divided into feature-based and NN-based methods.

Traditional approaches (Ahn, 2006; Chen and NG, 2012; Li et al., 2013; Li et al., 2014) usually rely on a series of NLP tools to extract lexical features (e.g., part-of-speech tagging, named entity recognition) and sentence-level features (e.g., dependency parsing). Although they achieve high performance, they often suffer from hard feature engineering and error propagation from those external tools. Recently, neural network models have been proved to show competitive performance against traditional models in event extraction. Chen et al. (2015) propose a convolutional neural network (CNN) to capture lexical features, with a dynamic multi-pooling layer to encode sentence-level clues.

[1] This is an example of multi-word event nugget.

[2] This is an example of multi-type event nugget.

[3] There are some cases where multiple event nuggets appear in the same sentence.

While Ghaeini et al. (2016) utilize a recurrent neural network (RNN) to solve the multi-word event nugget issue. Methods based on neural networks keep improving the performance on event extraction, and yield state-of-art.

Inspired by previous work, our system combines the feature-based method and neural-network-based method. To be specific, we first preprocess the raw text using Stanford CoreNLP tools (Manning et al., 2014), including sentence splitting, tokenization, POS tagging, lemmatization and named entity recognition. Then we input these sentences into different types of models, and ensemble their outputs at last.

3 Feature-based Method

Our feature-based method follows the standard pipeline paradigm, which divide event nugget detection into three subtasks:

1. trigger identification: recognize the event trigger, which is the main word or phrase that most clearly expresses the occurrence of an event.
2. trigger classification: assign an event type and subtype for an identified trigger
3. REALIS classification: assign a REALIS value for an identified trigger.

3.1 Trigger Identification

In the first step, we consider event trigger identification as a sequence labelling task. Sentences are tagged in the BIO scheme, where each token is labeled as B if it is the beginning of an event trigger, or I if it is inside a trigger, or O otherwise. We use two traditional classifiers, a Max Entropy model (Berger et al., 1996) and a Conditional Random Field (CRF) model (Lafferty et al., 2001). The feature templates used for trigger identification in different models are listed in Table 1.

Max Entropy Model We only keep those features that appear more than 3 times in the training set. For example, if a bigram feature appears 4 times in the training set, then we will keep it. Otherwise, we will discard this feature if it appeared less than 4 times in the training set. We use the implementation of Le Zhang [4] for all max entropy classifiers in our system.

Feature Templates	Max Entropy	CRF
$w_{i-2}w_{i-1}w_i$	$\surd$	
$w_{i-1}w_i$	$\surd$	
w_i	$\surd$	$\surd$
w_iw_{i+1}	$\surd$	
$w_iw_{i+1}w_{i+2}$	$\surd$	
$p_{i-1}p_i$	$\surd$	
p_i	$\surd$	
p_ip_{i+1}	$\surd$	
$l_{i-2}l_{i-1}l_i$		$\surd$
$l_{i-1}l_i$		$\surd$
l_i	$\surd$	$\surd$
l_il_{i+1}		$\surd$
$l_il_{i+1}l_{i+2}$		$\surd$
s_i	$\surd$	$\surd$
$wordnet\_synset_i$	$\surd$	

Table 1: Feature templates used in each model. w, p, l, s represents word, POS tag, lemma, and stem respectively. $wordnet\_synset_i$ indicates the WordNet synset that word w_i belongs to.

CRF Model The feature templates used in Max Entropy and CRF are designed to be slightly different, in order to obtain complementary contributions from the two classifiers. We use the CRF implementation from the CRF++ toolkit [5].

3.2 Trigger Classification

Although the event type system in Rich ERE Annotation Guidelines is a two-level hierarchy, we only consider the subtype level for classification since no subtype is shared by two or more first-level types. We build a Max Entropy model to perform the type classification task, where the feature templates we used are listed in Table 2.

However, Max Entropy model is not a flawless solution because it only assign one type for each trigger, while one trigger may possibly have multiple subtypes. We find that co-occurrence based heuristic rules can help to classify multi-type triggers.

First, we collect all triggers that may have multiple types, and record their most probable subtype combinations in the training set. Since most of them can be both single-type and multi-type with respect to the context, we need also develop a classifier to determine whether this appearance of the trigger should have multiple subtypes or not in the given

[4] https://github.com/lzhang10/maxent

[5] https://taku910.github.io/crfpp/

Feature	Description
$w_{ifirst\sim ilast}$	words in a trigger
$s_{ifirst\sim ilast}$	stems in a trigger
$synset_{ifirst\sim ilast}$	WordNet synsets in a trigger
$w_{i-2}w_{ifirst}$	w_{i-2} and first word of a trigger
$w_{i-1}w_{ifirst}$	w_{i-1} and first word of a trigger
$w_{i+1}w_{ilast}$	w_{i+1} and last word of a trigger
$w_{i+2}w_{ilast}$	w_{i+2} and last word of a trigger
$nearest\_entity$	the nearest entity to a trigger

Table 2: Feature templates used in our Max Entropy model for trigger classification. Note that one trigger may contain multiple words.

Feature	Description
$w_{ifirst\sim ilast}$	words in a trigger
$w_{i-2}w_{ifirst}$	w_{i-2} and first word of a trigger
d$w_{i-1}w_{ifirst}$	w_{i-1} and first word of a trigger
$w_{i+1}w_{ilast}$	w_{i+1} and last word of a trigger
$w_{i+2}w_{ilast}$	w_{i+2} and last word of a trigger
$p_{ifirst\sim ilast}$	POS tags of words in a trigger
$s_{ifirst\sim ilast}$	suffixes of words in a trigger
$m_{ifirst\sim ilast}$	modal auxiliaries of words in a trigger

Table 3: Feature templates used in our Max Entropy model for REALIS classification.

sentence. Specifically, if the current trigger is in our collected multi-type trigger list, we will use the Max Entropy model described in this section to output prediction scores for each subtype. If the difference of scores between top 2 subtypes is smaller than 0.5, then we will consider this trigger as a multi-type trigger, and assign the most probable subtype combination for this trigger.

3.3 REALIS Classification

Similar to the above subtasks, we build a Max Entropy model to perform the REALIS classification, where the features we use are listed in Table 3.

4 Neural Network Method

Unlike previous feature-based method, we jointly learn trigger identification and type classification by one neural network to reduce the error propagation problem of a pipeline model. Then we assign a REALIS value for each trigger.

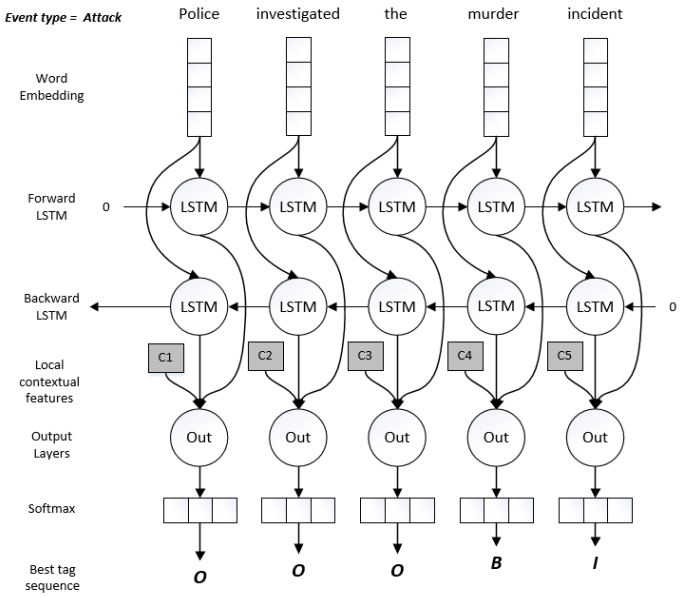

Figure 1: The main architecture of our word-based model. The local contextual feature c_t (grey rectangle) in Figure 1 for each word w_t is computed by the CNN as Figure 2 illustrated.

4.1 Trigger identification and classification

To address the multi-word trigger and multi-type trigger issues, we treat this trigger labeling task (including trigger identification and classification) as a sequence labeling problem. For each event type $type$, we train a neural network model that labels each sentence in the BIO scheme. Specifically, a word is labeled as B if it is the beginning of a trigger of type $type$, or I if it is inside a trigger of type $type$, or O otherwise. As we train the models for each type independently, one word can belong to several types.

Figure 1 illustrates the main architecture of our neural network model, which is a BiLSTM model with a CNN layer as shown in Figure 2.

BiLSTM Network Recurrent neural networks maintain a memory based on historical contextual information, which makes them a natural choice for processing sequential data. Long Short-Term Memory (Hochreiter and Schmidhuber, 1997) is explicitly designed to solve the long-term dependency problem through purpose-built memory cells. For the event detection task, if we access to both past and future contexts for a given time, we can make use of more sentence-level information and make

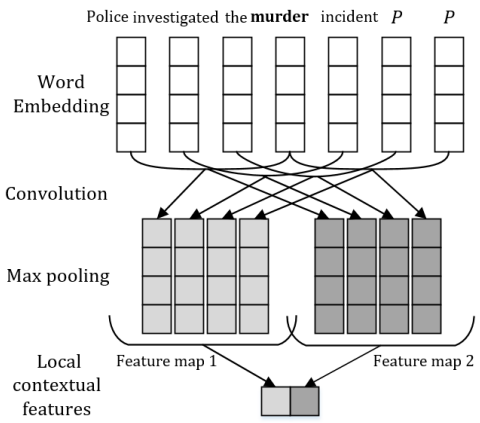

Figure 2: Our convolutional neural network learns a representation of local context information about the center word **murder**. Here the context size is 7 (3 words to the left and to the right of a center word), and we use a kernel of size 4 with two feature maps. The symbol P in sentence represents a padding word.

better prediction. This can be done by bidirectional LSTM networks. A forward LSTM network computes the hidden state $\overrightarrow{h_t}$ of the past (left) context of the sentence at word w_t, while a backward LSTM network reads the same sentence in reverse and outputs $\overleftarrow{h_t}$ given the future (right) context. In our implementation, we concatenate these two vectors to form the hidden state of a BiLSTM network, i.e. $h_t = [\overrightarrow{h_t}; \overleftarrow{h_t}]$.

Convolutional Neural Network We employ a convolutional neural network as illustrated in Figure 2 to extract local contextual information for every word. Given a sentence containing n words $\{w_1, w_2, ..., w_n\}$, and the current center word w_t, a convolution operation involves a kernel, which is applied to words surrounding w_t in a window to generate the feature map. We can utilize multiple kernels with different widths to extract local features of various granularities. Then max pooling is performed over each map so that only the largest number of each feature map is recorded. One property of pooling is that it produces a fixed size output vector, which enables us to apply variable kernel sizes. Finally, we take the fixed length output vector c_{w_t} as a representation of local contextual information about center word w_t.

The Output Layer We concatenate the hidden state h_t of BiLSTM with contextual feature c_{w_t} extracted by CNN at each time step t. Then $[h_t; c_{w_t}]$ is fed into a softmax layer to produce the log-probabilities of each label for w_t.

Implementation We implement this neural network using Tensorflow library (Abadi et al., 2016). The 100-dimension word embeddings are pre-trained on the training set, and fine-tuned during training. In LSTM, the state size is 100. As for CNN, the sliding window size is 7 (3 words to the left and to the right of a center word), and we set the kernel sizes from 2 to 7 to capture context information of various granularities.

4.2 REALIS Classification

In the above section, we present our convolution BiLSTM model for trigger labeling. The idea of this neural network architecture is also suitable for REALIS classification. Next, we will present the main differences between the models used in these two tasks.

The Input Layer As a pipeline system, besides word embeddings, we can use information extracted from upstream trigger labeling task. Therefore, we propose four additional types of feature embeddings to form the input layer of BiLSTM and CNN.

- Trigger position feature: whether a word is in the current trigger
- Trigger type feature: trigger type of a word, and a special type *NONE* for words outside the current trigger
- Name entity type feature: entity type of a word, and a special type *NONE* type for non-entity words
- POS tag feature: part-of-speech tag of a word

We then transform these features into vectors by their lookup tables, and concatenate them with the original word embeddings, as the final input layer of BiLSTM and CNN.

The Output Layer It is worth mentioning that REALIS classification is no longer a sequence labelling task, but a classification task. We only need to assign a REALIS value for each trigger instead of every word in the whole sentence. For instance, there are three trigger (bold words) in the following sentence, which together makes up three words to be classified.

- Six **murders** occurred in France, including the **assassination** of Bob and the **killing** of Joe.

We modify the output layers of both CNN and BiLSTM network to adapt to the the new task. For BiLSTM, we regard the hidden state of the last word h_n as sentence-level information. And for CNN, we take all words of the entire sentence as the context, rather than a shallow window for each center word. Finally, we feed the concatenation of output vectors from two networks into a softmax classifier just like trigger labeling.

5 Ensemble

Since we have more than one predictors in each subtask, we need to combine the outputs of each model to produce more reliable results. Our ensemble strategy follows the same three-step pipeline paradigm as feature-based method.

In the trigger identification step, we train a Max Entropy model, a CRF model and a Convolution BiLSTM (C-BiLSTM) model. The first two models simply predict whether a word is in a trigger, while C-BiLSTM models predict the BIO label with respect to different event types. So we first combine the results of C-BiLSTM models by following rules:

1. If a word is labelled as B by any C-BiLSTM model, label the word as B
2. If a word is labelled as I by any C-BiLSTM model, label the word as I
3. If a word is labelled as O by all C-BiLSTM models, label the word as O

After determining the result of C-BiLSTM models, we predict the final label by majority voting.

In the second step, for each event type e, we calculate a new score of every word in the sentences according to the formulas below:

$$score_e = \frac{1}{feature\_rank_e} + NN_e \quad (1)$$

$$NN_e = \begin{cases} 0.8 & \text{when } label_e = B \text{ or } I \\ 0 & otherwise \end{cases} \quad (2)$$

$feature\_rank_e$ is the confidence score rank of type e among all event types. And $label_e$ is the label output by the C-BiLSTM model trained for event type e. As an event trigger could be annotated with multiple event types, the resulting scores are further enhanced in a multi-type style using the co-occurrence based heuristic rules introduced in Section 3.2.

In the third step, we also calculate a new score for each REALIS value r with the following formulas:

$$score_r = \frac{1}{feature\_rank_r} + NN_r \quad (3)$$

$$NN_r = \begin{cases} 0.8 & \text{when } label = r \\ 0.2 & otherwise \end{cases} \quad (4)$$

$feature\_rank_e$ is the confidence score rank of value r, while $label$ is the output of C-BiLSTM model introduced in Section 4.2. We choose the value that gets maximum $score$ as final prediction.

6 Experiments

6.1 Setup

We use the training and evaluation data in TAC KBP 2015 from LDC2016E36 dataset for training. There are 360 documents in this corpus (158 for training, and 202 for evaluation). We randomly select 60 documents from evaluation data as validation set, and the remaining 300 documents as training set. During training, we keep checking performance on the validation set and pick the parameters that preforms best for final evaluation.

6.2 Results

The result on the TAC KBP 2016 test set are shown in Table 4.

7 Conclusion

In this paper, we propose an event detection system that can detect event triggers and assign both event

Attributes	Micro		
	Precision	Recall	F1
plain	57.09	43.28	49.23
mention_type	51.43	38.99	44.35
realis_status	42.86	32.49	36.96
type+realis	38.38	29.10	33.10

Attributes	Macro		
	Precision	Recall	F1
plain	53.71	41.46	46.80
mention_type	47.42	36.93	41.52
realis_status	39.61	30.87	34.70
type+realis	34.71	27.29	30.56

Table 4: Results on the final test set.

type and event REALIS value. This system incorporates many effective classifiers and obtains promising results in the final evaluations. Currently, we only consider the ensemble of different classifiers locally, and it would be worth incorporating more global constraints to further reduce the error propagation in the pipeline.

Acknowledgement

This work was supported by National High Technology R&D Program of China (Grant No. 2015AA015403), Natural Science Foundation of China (Grant No. 61672057) and the joint project with IBM Research. Any correspondence please refer to Yansong Feng.

References

Martın Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, et al. 2016. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467*.

David Ahn. 2006. The stages of event extraction. In *Proceedings of the Workshop on Annotating and Reasoning about Time and Events*, pages 1–8. Association for Computational Linguistics.

Adam L Berger, Vincent J Della Pietra, and Stephen A Della Pietra. 1996. A maximum entropy approach to natural language processing. *Computational linguistics*, 22(1):39–71.

Chen Chen and V Incent NG. 2012. Joint modeling for chinese event extraction with rich linguistic features. In *In COLING*. Citeseer.

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, pages 167–176.

Reza Ghaeini, Xiaoli Z Fern, Liang Huang, and Prasad Tadepalli. 2016. Event nugget detection with forward-backward recurrent neural networks. In *The 54th Annual Meeting of the Association for Computational Linguistics*, page 369.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

John Lafferty, Andrew McCallum, and Fernando Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the eighteenth international conference on machine learning, ICML*, volume 1, pages 282–289.

Qi Li, Heng Ji, and Liang Huang. 2013. Joint event extraction via structured prediction with global features. In *ACL (1)*, pages 73–82.

Qi Li, Heng Ji, Yu Hong, and Sujian Li. 2014. Constructing information networks using one single model. In *EMNLP*, pages 1846–1851.

Christopher D Manning, Mihai Surdeanu, John Bauer, Jenny Rose Finkel, Steven Bethard, and David McClosky. 2014. The stanford corenlp natural language processing toolkit. In *ACL (System Demonstrations)*, pages 55–60.

Zhiyi Song, Ann Bies, Stephanie Strassel, Joe Ellis, Teruko Mitamura, Hoa Dong, Yukari Yamakawa, and Sue Holm. 2016. Event nugget and event coreference annotation. In *The 2016 Conference of the North American Chapter of the Association for Computational Linguistics-Human Language Technologies (NAACL HLT 2016). 4th Workshop on EVENTS: Definition, Detection, Coreference, and Representation*.