WIP Event Detection System at TAC KBP 2017 Event Nugget Track

Ying Zeng and **Yansong Feng** and **Dongyan Zhao**
Institute of Computer Science and Technology, Peking University
{ying.zeng, 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 argument extraction and event coreference resolution. In this paper, we propose an event detection system that combines traditional feature-based methods and 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 2017 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, 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.

S1: Hillary Clinton was not **elected** president in 2008. [Elect, OTHER]

S2: The police are investigating the **murder incident**.[1] [Attack, ACTUAL]

S3: Correa was even accused without any evidences of **murder**.[2] [Attack, OTHER; Die, OTHER]

S4: Kennedy was **shot** **dead** by Oswald.[3] [Attack, ACTUAL], [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

Most existing approaches to event extraction are supervised and 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

[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.

clues. While Ghaeini et al. (2016) utilize a recurrent neural network (RNN) to solve the multi-word event nugget issue. Feng et al. (2016) combine Bi-directional LSTM (BiLSTM) and convolutional neural networks to learn a continuous representation for each word and predict whether it is an event trigger or not. 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. Specifically, 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 a conditional random field (CRF) model, a maximum entropy (MaxEnt) model, and a bidirectional recurrent neural network (RNN) model, separately. Finally, we ensemble outputs from different models 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.

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

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.

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.

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

[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.

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 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.

S4	Kennedy was **shot** **dead** by Oswald.
NN$_{attack}$	`O O B-ACTUAL O O O`
NN$_{die}$	`O O O B-ACTUAL O O`
NN$_{elect}$	`O O O O O O`

Table 4: Examples of output sequences by three NN models trained for different event types, $attack$, die and $elect$.

4 Neural Network Method

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

4.1 Tagging Scheme

To address the multi-word trigger and multi-type trigger issues as mentioned in Section 1, we treat event nugget detection task 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-REALIS` if it is the beginning of a trigger with regard to a `type` event whose REALIS value is `REALIS`, or `I-REALIS` if it is inside a trigger, or `O` otherwise. For better understanding, resulting sequences of **S4** labeled by models of different event type are listed in Table 4. As we train the models for each type independently, one word can belong to several types.

Next, we introduce the layers in our BiGRU-CRF network one-by-one from bottom to top.

4.2 BiGRU 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 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 apply a variation of LSTM units, Gated Recurrent Unit (GRU) (Cho et al., 2014), which is found to be superior to LSTM on a suit of tasks by Chung et al. (2014). We concatenate these two vectors to form the hidden state of a BiGRU network, i.e. $h_t = [\overrightarrow{h_t}; \overleftarrow{h_t}]$.

4.3 CRF layer

We propose a BiGRU-CRF model that considers the correlations between labels in neighborhoods and jointly decodes the best sequence of labels via a CRF layer. Given an input sentence of length n, we consider $\boldsymbol{P}$ to be a matrix of confidence scores output by BiGRU network. $\boldsymbol{P}$ is of size $n \times e$, where e is the number of distinct tags, and $P_{i,j}$ correspond to the confidence of the j-th tag for the i-th word in a sentence. We add a state transition matrix $\boldsymbol{A}$ in CRF layer such that $A_{i,j}$ represents the score of a transition from label i to label j. We take into account neural network outputs and transition scores, and score a sentence $\boldsymbol{X}$ along with a path of labels $\boldsymbol{y} = \{y_1, y_2, \ldots, y_n\}$ to be

$$score(\boldsymbol{X}, \boldsymbol{y}) = \sum_{i=0}^{n} A_{y_i, y_{i+1}} + \sum_{i=1}^{n} P_{i, y_i}, \tag{1}$$

where y_0 and y_{n+1} are the special labels, `start` and `end`, that we add to the set of possible labels. $\boldsymbol{A}$ is therefore a square matrix of size $(e+2) \times (e+2)$.

We normalize this score over all possible label sequences $\tilde{\boldsymbol{y}}$ using a softmax, and we interpret the resulting ratio as a conditional label sequence probability over all possible label sequences:

$$p(\boldsymbol{y}|\boldsymbol{X}) = \frac{\exp(score(\boldsymbol{X}, \boldsymbol{y}))}{\sum_{\tilde{\boldsymbol{y}} \in \boldsymbol{\mathcal{Y}}} \exp(score(\boldsymbol{X}, \tilde{\boldsymbol{y}}))}. \tag{2}$$

We implement this neural network using Tensorflow library (Abadi et al., 2016). The 100-dimension word embeddings are pre-trained on the Wikipedia dump, and fine-tuned during training. And the size of BiGRU unit is set to be 64. All models share a generic stochastic gradient descent forward and backward training procedure. Parameter optimization is performed using Adam (Kingma and Ba, 2014) with gradient clipping (Pascanu et al., 2013). We apply the dropout method (Srivastava et al., 2014) on both the input and output vectors of all models to alleviate overfitting.

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 BiGRU-CRF model. The first two models simply predict whether a word is in a trigger, while BiGRU-CRF models predict the `BIO` label with respect to different event types. So we first combine the results of BiGRU-CRF models by following rules:

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

After determining the result of BiGRU-CRF 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:

$$score_e = \frac{1}{feature\_rank_e} + NN_e \tag{3}$$

$$NN_e = \begin{cases} 0.8 & \text{when } label_e = B \text{ } or \text{ } I \\ 0 & otherwise \end{cases} \tag{4}$$

$feature\_rank_e$ is the confidence score rank of type e among all event types. And $label_e$ is the label output by the BiGRU-CRF 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:

$$score_r = \frac{1}{feature\_rank_r} + NN_r \tag{5}$$

Attributes	Micro		
	Precision	Recall	F1
plain	65.91	43.86	52.67
mention_type	57.84	38.49	46.22
realis_status	48.12	32.02	38.45
type+realis	42.21	28.08	33.73

Attributes	Macro		
	Precision	Recall	F1
plain	66.46	44.39	53.23
mention_type	58.70	39.44	47.18
realis_status	48.56	33.27	39.49
type+realis	42.56	29.33	34.73

Table 5: Results on the final test set.

$$NN_r = \begin{cases} 0.8 & \text{when } label = r \\ 0.2 & otherwise \end{cases} \tag{6}$$

$feature\_rank_e$ is the confidence score rank of value r, while $label$ is the output of BiGRU-CRF model introduced in Section 4. 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 and 2016 contest for training. There are altogether 529 documents in these corpora. We randomly select 104 documents as validation set, and the remaining 425 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 2017 test set are shown in Table 5.

7 Conclusion

In this contest, we propose an event nugget detection system that can detect event triggers and assign an event type and a REALIS value to each trigger. This system incorporates many effective classifiers and obtains promising results in the final evaluations.

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.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.

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*.

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.

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.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

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.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. 2013. On the difficulty of training recurrent neural networks. *ICML (3)*, 28:1310–1318.

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*.

Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958.