WIP Event Detection System at TAC KBP 2015 Event Nugget Track

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

Abstract

Event detection is the first step in various event related applications, such as slot filling and event coreference resolution, which are usually investigated in a pipeline style. Therefore, it is crucial that we can achieve reasonably good performance in the upstreaming step of the pipeline, i.e., event detection and categorisation. In this paper, we describe an event detection system that combines traditional Max Entropy and CRF event detectors, recent Neural Network models and a seed-based empirical method. Experimental results show that such an ensemble strategy can obtain promising performances in the Event Nugget Detection task.

1 Introduction

Event detection is the task of identifying the main word tokens that indicate an event of a pre-defined type from given text. For example, given a sentence *An American tank fired on the hotel*, *fired* is a trigger of an attack event. Traditional event detectors first detect word tokens as event triggers, and then classify them into appropriate event types. In TAC KBP 2015, the Event Nugget Track also requires the participants to provide a realis type for all detected events, where event mentions will refer to ACTUAL when the events actually occurred, while GENERIC events are those without a specific known or unknown time and/or place, and OTHER refers to failed events, future events, conditional statements and all other non-generic variations.

In the remaining parts of this paper, we will first provide an overview of our system, and describe the models we used in each subtask in detail. Finally, we show how our system works in the evaluation dataset and conclude our system in the conclusion section.

2 System Overview

Our system follows the standard pipeline paradigm, and our main framework is shown in Figure 1. First, we preprocesses the raw text using Stanford CoreNLP tools (Manning et al., 2014), including tokenization, sentence splitting, POS tagging, lemmatization and named entity recognition. After that, the preprocessed data is passed to the event trigger identifiers. We use four different trigger identification models to make their predictions independently, and ensemble these predictions before delivering them to the event type classifiers. Similarly, we build three type classifiers and combine their predictions as input to the last realis classification step. There are two realis classifiers and the final output is produced by combining all previous results.

3 Trigger Identification

In the first step, we consider event trigger identification as a sequence labelling task. To be specific, we adopt the BIO tagging scheme. For each trigger in the training set, we tag the first word as **B**, the remaining words of this trigger as **I**, and for all other words that are not belonging to any triggers, we tag them as **O**. In our experiments, we uses three classifiers, a Max Entropy model, a Conditional Random Field model, and a Neural Networks model.

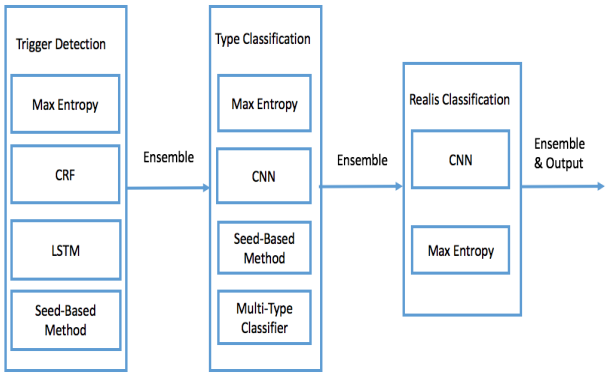

Figure 1: Overview of our proposed event detection system.

3.1 The Max Entropy Model

The first classifier we used is the Max Entropy model(Berger et al., 1996). We use the implementation of Le Zhang[1] for all max entropy classifiers in our system. The feature templates used for trigger identification are listed in Table 1. Note that 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.

3.2 The CRF Model

Our second classifier is the Conditional Random Field model (CRF) (Lafferty et al., 2001), a sequence labelling model. We use the CRF implementation from the CRF++ toolkit[2]. The feature templates used in the trigger identification are listed in Table 1. Note that, 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.

3.3 Long Short Term Memory Network

We also implement a two-layer Long Short Term Memory (LSTM) neural network model (Hochreiter and Schmidhuber, 1997) using Torch[3] to perform the sequence labelling task. Specifically, we use 200-dimension GloVe (Pennington et al., 2014) as our initial word vectors. For each word, we append additionally another 50-dimension randomly initialized POS tag vector to the GloVe word vector to explore shallow syntactic information. Each layer is of size 250, and for training, we use 0.5 dropout rate combined with momentum stochastic gradient descent (SGD). The learning rate is 0.01 and momentum is 0.9.

[1]https://github.com/lzhang10/maxent
[2]https://taku910.github.io/crfpp
[3]https://github.com/torch/torch7

Feature Template	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 refers to word, p refers to POS tag, l refers to lemma, s refers to stem. $wordnet\_synset_i$ refers to the **WordNet synset** that the current word belongs to.

3.4 The Seed-Based Method

We will go through this module in the event type classification section (Section 4.3), since trigger detection and type classification are performed jointly in this method.

4 Event Type 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.

4.1 The Max Entropy Model

First, we build a Max Entropy model to perform the type classification task, where the feature templates we used are listed in Table 2.

Feature	Description
$w_{ifirst-ilast}$	words in this trigger
$s_{ifirst-ilast}$	stems in this trigger
$synset_{ifirst-ilast}$	WordNet synsets in this trigger
$w_{i-2}w_{ifirst}$	i-2 word, first word of this trigger
$w_{i-1}w_{ifirst}$	i-1 word, first word of this trigger
$w_{i+1}w_{ilast}$	i+1 word, last word of this trigger
$w_{i+2}w_{ilast}$	i+2 word, last word of this trigger
nearest_entity	the nearest entity to this trigger

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

Feature	Description
Lemma	Do the candidate token and a seed share the same lemma?
Synonym	Is a seed a WN synonym of the candidate token?
Hypernym	Is a seed a WN hypernym or instance-hypernym of the candidate token?
Similarity Relations	Does one of these WN relations hold between a seed and a candidate token? Synonym, Hypernym, Instance Hypernym, Part Holonym, Member Holonym, Substance Meronym, Entailment

Table 3: Features used in the seed-based method. WN refers to WordNet.

4.2 Convolutional Neural Network

Secondly, we implement a convolutional neural network (CNN) model(LeCun and Bengio, 1995) with Torch to predict event types. We also use 200-dimension GloVe word vectors, and append additionally two 50-dimension randomly initialized vectors for POS tag and NER tag, respectively, and a 30-dimension vector for the distance from the current word to the first word of the current trigger. We use 400 convolution kernels and max-pooling over the outputs of each convolution kernel to produce the convolution output. Combined with the 200-dimension GloVe word vector and the 50-dimension POS tag vector of the first word of the trigger, the convolution output is passed to a 2-hidden-layer perceptron followed by a softmax layer to predict the event type label for the current trigger.

4.3 Seed-Based Method

This seed-based method is originally proposed in (Bronstein et al., 2015), where the basic idea is that most triggers for certain event types are related to the same base triggers, to some extent. Since both ACE guidelines and Rich ERE Annotation Guidelines provide example triggers for each event type, we can simply utilize these triggers as seeds, and judge the relatedness of each word in a sentence to these seed triggers. If the current word does share some similarity with one of the seed triggers, then we can consider this word as a trigger of the event type that is same with the specified seed trigger.

To be specific, we collect 218 seed triggers from ACE guidelines and TAC KBP Event Track Documents. We train a structured perceptron with beam search to calculate the relatedness of the current word to seed triggers for each subtype. The features we used are listed in Table 3. For each subtype, we train 5 predictors with different training sets. For each predictor, we randomly choose 6 other subtypes and 50 event mentions of these 6 subtypes as training set. During testing, we judge the relatedness of the current word to each subtype. For each subtype, if half of the predictors give positive predictions, then we consider the current word belonging to this subtype. Note that one word may belong to several subtypes.

4.4 Multi-Type Classification Method

Note that one event trigger may possibly have multiple subtypes, but our Max Entropy model and CNN model provide only one subtype for each trigger. One way to deal with this issue is to use the seed-based method, which handles multiple labels in its nature. We also find that co-occurrence based heuristic rules can help to output multiple labels.

First, in the training set, we collect all triggers that may have multiple types, and record their most probable subtype combinations. 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

Feature	Description
$w_{ifirst-ilast}$	words in current trigger
$w_{i-2}w_{ifirst}$	i-2 word, the first word of current trigger
$w_{i-1}w_{ifirst}$	i-1 word, the first word of current trigger
$w_{i+1}w_{ilast}$	i+1 word, the last word of current trigger
$w_{i+2}w_{ilast}$	i+2 word, the last word of current trigger
$p_{ifirst-ilast}$	POS tags in current trigger
$s_{ifirst-ilast}$	suffixes of words in current trigger
$m_{ifirst-ilast}$	modal auxiliaries of words in current trigger

Table 4: Features used in the Max Entropy model for realis classification.

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.

5 Event Realis Classification

5.1 The Max Entropy Model

Again, we first build a Max Entropy model to perform the event realis classification, where the features we use are listed in Table 4.

5.2 Convolutional Neural Network

Here, we use the same convolutional neural network framework as the one we introduced in the event type classification to perform event realis classification.

6 Ensemble

Since we have more than one predictors in each subtask, we need to combine the results of each model to produce more reliable results. Generally, not applicable for the seed-based method and multi-type classifier, we use the sum of inverse rank as the new score for each candidate predictions. The formula is given below:

$$Score_i = \sum_j \frac{1}{rank_{ij}} \quad (1)$$

For label i, we sum its inverse rank of all the predictors to get its final score (j refers to the jth predictor).

7 The Seed-Based Method

In practice, we find that the seed-based method can achieve a higher precision for certain subtypes. We thus consider the seed-based method as a default classifier for those subtypes. That is, for these subtypes, we simply trust the predictions from the seed-based method, even though other classifiers do not agree with the seed-based models.

8 Dealing with Multiple Event Types

An event trigger could be annotated with multiple event types. Therefore, we first ensemble the Max Entropy event type classifier and the CNN event type detector using formula 1, the output of which are subsequently filtered and by our seed-based method. The resulting predictions are further enhanced in a multi-type style using the co-occurrence based heuristic rules introduced in Section 4.4.

9 Experiments

9.1 Setup

We use the LDC2015E73 dataset for training. Specifically, we randomly choose 90% as training data and the other 10% as development data. Since we have several neural networks models in our system, which are expensive to train and tune in a full cross-validation form, we thus tune our system in a two-step style. We first run one set of empirical parameters on 4 training/validation splits, choose the training/validation split that performs best, and fine tune our model in this training/validation split.

9.2 Results

The results on the final test set are shown in Table 5.

Attributes	Micro		
	precision	recall	F1
plain	79.40	48.61	60.30
mention type	71.06	43.50	53.97
realis status	57.79	35.38	43.89
type + realis	52.12	31.90	39.58

Attributes	Macro		
	precision	recall	F1
plain	77.79	48.57	59.80
mention type	70.14	43.67	53.83
realis status	57.94	35.96	44.38
type + realis	52.56	32.47	40.14

Table 5: Results on the final test set.

10 Conclusion

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

Acknowledgments

This work was supported by National High Technology R&D Program of China (Grant No. 2015AA015403, 2014AA015102), Natural Science Foundation of China (Grant No. 61202233, 61272344, 61370055) and the joint project with IBM Research. Any correspondence please refer to Yansong Feng.

References

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.

Ofer Bronstein, Ido Dagan, Qi Li, Heng Ji, and Anette Frank. 2015. Seed-based event trigger labeling: How far can event descriptions get us? *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL 2015, July 26-31, 2015, Beijing, China, Volume 2: Short Papers*, page 372.

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

John Lafferty, Andrew McCallum, and Fernando CN Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data.

Yann LeCun and Yoshua Bengio. 1995. Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, 3361(10).

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*, pages 55–60.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. *Proceedings of the Empiricial Methods in Natural Language Processing (EMNLP 2014)*, 12:1532–1543.