IECAS Event Detection System at TAC KBP 2017 Event Nugget Track

Yunyan Zhang [1,2],Tinglei Huang[1],Yang Wang[1]
[1]Institute of Electronics, Chinese Academy of Sciences, Beijing 100190, China
[2]University of Chinese Academy of Sciences, Beijing 100190, China
zhangyunyan15@mails.ucas.ac.cn
{tlhuang, ywang1}@mail.ie.ac.cn

Abstract— **This paper describes our Event Nugget Detection systems that we submitted to the TAC KBP 2017 Event Track. We implement pipelined event detection systems based on convolutional neural network (CNN) and bidirectional Long Short-Term Memery (BiLSTM) for both event type and realis classification. Our systems use only word embeddings as external resources for training so they can be easily adapted to multiple languages. The experimental results show that our approaches can achieve promising performances in the Event Nugget Detection task.**

I. INTRODUCTION

Event extraction, which aims to extract event mentions of specific types and their corresponding event arguments from unstructured text, is a traditional task in Information Extraction (IE). Event detection is an essential and challenging problem in event extraction and attracts much attention in recent years.

Text Analysis Conference Knowledge Base Population (TAC KBP) 2017 is an evaluation workshop in Neural Language Processing, which provides a large test collection and common evaluation procedures.

We participate in the Event Nugget Track for Event Nugget Detection, which is one of the tacks in TAC KBP 2017. This task is required to detect the event mentions in the input documents and classify them into the types and subtypes predefined in the Rich ERE guidelines. Also the task is to identify REALIS attribute (actual, generic, other) for every event mention. Here are some examples of event nugget with specific type and realis status in annotated data. The words in **bold face** are event nuggets.

- Fortunately, despite **firing** [Conflict.Attack, ACTUAL] at close range, the shooter didn't **hit** [Life.Injure, OTHER] anyone.
- She was **killed** [Life.Die, ACTUAL] in an automobile acciden.
- Fred **visited** [Movement.Transportperson, ACTUAL] New York on Friday.
- Fred **visited** [Contact.Meet, ACTUAL] Harry in New York on Friday.
- They used tear gas to **break up** [Conflict.Attack, ACTUAL] a **gathering** [Conflict.Demonstrate, ACTUAL] of some masked protesters.

We follow a pipelined manner to build our Event Nugget Detection systems, which involve two components in order: (1) event detection and classification (called TYPE), (2) event realis classification (called REALIS). The input for TYPE is raw text and the output of TYPE becomes the input of REALIS. The output of REALIS is submitted to the Event Nugget Detection task evaluation. We use neural networks as the main technique in both components. We train and evaluate our systems only on English documents. Since our systems use only word embeddings as resources for training, they can be adapted for multiple languages easily.

II. RELATED WORK

Existing methods for event detection can be devided into two groups: feature-based methods and neural network based methods.

Traditional approaches [1], [2], [3], [4], [5], [6] usually rely on lexical features (such as Part-of-

Speech, lemma, named entity tag) and context features (such as dependency parsing). However, the performance of these approaches often depends highly on the quality of the feature engineering and these approaches inevitably suffer from error propogation of external NLP tools.

Recently, deep learning techniques have been widely used in many NLP tasks and also outperform the traditional methods for event detection. [7] implement a dynamic multi-pooling convolutional neural network that automaticly learn effective feature representations for event trigger detection. [8] use a bidirectional Long Short-Term Memory model to handle multi-word events. [9] develop a non-consecutive convolution neural network to model the non-consecutive n-grams that are crucial in some situations.

III. SYSTEM DESCRIPTION

In this work, we consider the event detection and classification task as a classification problem for every token in the input documents. We overlook the multi-word event mentions and assume event mentions to be only single tokens in text to introduce the position information into neural network models.

The Rich ERE annotation guidelines defines 7 event types and 18 event subtypes belonging to the corresponding event types. We classify the event mentions into their specific subtypes directly since none of subtypes are shared by two or more event types. That's to say, we predict the tokens for 19 classes (18 subtypes plus one type for "NONE" if it's not an event mention).

Our preprocessing steps include sentence detection and tokenization using the Stanford CoreNLP toolkit [1] .

A. Encoding

For every token in an input sentence, we want to predict its event subtype. The current word x_0 along with its context in the same sentence constitute an event mention candidate $x = [x_{-m}, x_{-m+1}, \ldots, x_0, \ldots, x_{n-1}, x_n]$. In order to prepare the input of neural networks, we limit the length of event mention candidate by trimming longer sentences and padding shorter sentences.

[1]http://stanfordnlp.github.io/CoreNLP

Before entering the neural networks, each token x_i in the event mention candidates is transferred into a real-valued vector using the concatenation of two vectors:

- Word embedding of x_i: Following the previous work, we use word embeddings to capture the hidden sementic and syntactic properties of tokens. Specifically, we use 300d pre-trained word embeddings trained on a 6B token corpus with GloVe [10] to initialize the word embeddings.
- Position embedding of x_i: It's necessary to specify which token is the predicted event mention in the sentence. So we utilize the position embedding to embed the relative distance between x_i and the current token x_0. The dimension of position embedding is set to 10. We initialize this table randomly.

As a result the original event mention candidate is transformed into a matrix $\mathbf{x} = [\mathbf{x}_{-m}, \mathbf{x}_{-m+1}, \ldots, \mathbf{x}_0, \ldots, \mathbf{x}_{n-1}, \mathbf{x}_n]$ of size $d \times l$ (d is the dimensionality of the concatenated vectors of the tokens. l is the length of x, in this case, $l = m + n + 1$). The matrix is then fed into the neural network models.

B. Models

We mainly use three deep learning models in our submission: CNN, RNN, RCNN and finally a soft voting method for model ensemble:

1) CNN: Since the conventional neural network (CNN) is capable of capturing consecutive n-gram features from text, we design a CNN model following the previous works [7], [11]. Specifically, we use multiple convolutional filters with different widths to capture n-grams of various granularities. Each convolutional layer is followed by a Batch-Norm layer, ReLU activation and a MaxPooling layer. The outputs of CNN with different filters are concatenated to be the input of a two-layer fully connected classifier with ReLU hidden units and softmax outputs to compute the probability distribution over the possible event subtypes for the event mention candidate. An illustration of CNN is given in Figure 1.

2) RNN: Besides CNN, we use a bidirectional Long Short-Term Memory (BiLSTM) for event detection. BiLSTM is a two-way recurrent neural

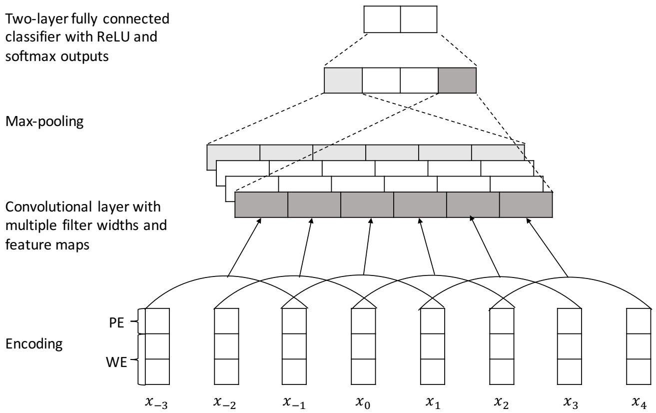

Fig. 1. An illustration of CNN model

network which can model both preceding and following contexts.

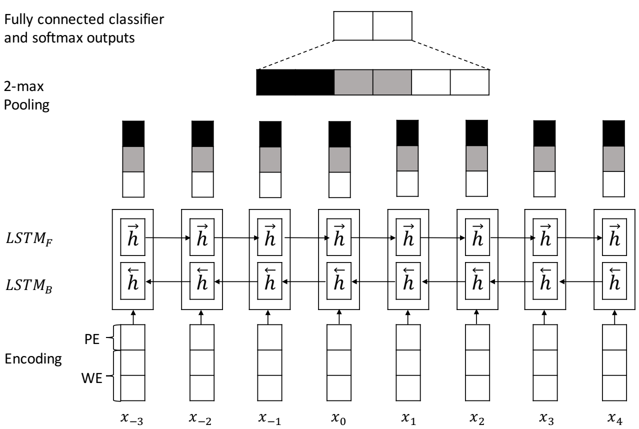

Fig. 2. An illustration of BiLSTM model

As presented in Figure 2, BiLSTM is composed of two LSTM neural networks, a forward $LSTM_F$ to capture the preceding context information and a backward $LSTM_B$ to capture the following context information. In particular, the input sequence of BiLSTM is the encoding matrix of the event mention candidate, denoted by $\mathbf{x} = [\mathbf{x}_{-m}, \mathbf{x}_{-m+1}, ..., \mathbf{x}_0, ..., \mathbf{x}_{n-1}, \mathbf{x}_n]$. For $LSTM_F$, we compute the hidden state h_i based on the current input $\mathbf{x}_i$ and the previous hidden state h_{i-1} at each time step i. This process is conducted over $\mathbf{x}$ to get the hidden state sequence $\overrightarrow{h}(\mathbf{x}_{-m}, \mathbf{x}_{-m+1}, ..., \mathbf{x}_0, ..., \mathbf{x}_{n-1}, \mathbf{x}_n) = (h_{-m}, h_{-m+1}, ..., h_0, ..., h_{n-1}, h_n)$. Then for $LSTM_B$ we run the LSTM in the reverse direction of the input sequence from $\mathbf{x}_n$ to $\mathbf{x}_{-m}$ to generate the second hidden state sequence $\overleftarrow{h}(\mathbf{x}_n, \mathbf{x}_{n-1}, ..., \mathbf{x}_0, ..., \mathbf{x}_{-m+1}, \mathbf{x}_{-m}) = (h'_n, h'_{n-1}, ..., h'_0, ..., h'_{-m+1}, h'_{-m})$. Afterwards, we concatenate the output $\overrightarrow{h}$ of $LSTM_F$ with the output $\overleftarrow{h}$ of $LSTM_B$ at each time step i: $f_i = [h_i, h'_i]$.

Instead of using the hidden state of the last time step $f_n = [h_n, h'_n]$, we feed the output features of each hidden state to a dynamic k-max pooling layer. Dynamic k-max pooling [12] is a generalisation of the max pooling operator. Unlike max pooling operator, which returns the single maximum value, k-max pooling returns the subsequence of k maximum values to reserve more information in the sequence. Afterwards, the resulting vector is then used as input for a two-layer fully connected layer with ReLU hidden units, followed by a softmax layer.

3) RCNN: We build another neural network model which is a bit different from the RNN model presented in the last section.

Similarly, we use a BiLSTM to capture both preceding and following information. The output features of each hidden state are concatenated as f_{all}. In order to enrich the f_{all}, we concatenate it with f_{em}, which is the embeddings of tokens in the event mention candidate. The resulting concatenated vector $[f_{all}, f_{em}]$ is then used as the input of a CNN block whose architecture is similar to our CNN model. In particular, the CNN block has a convolutional layer followed by a BatchNorm layer, ReLU activation and a MaxPooling layer. Since we use a CNN block here to capture the chunk feature of BiLSTM, we believe one conventional filter (with width of 3) is sufficient. The outputs of the CNN block pass through a two-layer fully connected classifier with ReLU hidden units and softmax outputs.

4) Ensemble: In order to improve generalization and robustness over single predictors, we emsemble multiple independently trained models. Due to the limited time, we only apply a simple soft voting strategy to ensemble CNN, RNN and RCNN models presented in the previous sections. Soft voting returns the class label as argmax of the weighted sum of predicted probabilities. We assign different weights to CNN, RNN, RCNN respectively to maximum the performance.

5) REALIS classification : In the above sections, we presented our systems for event type classification. The similar encoding and network can be adapted to event realis classification. Instead of predicting the tokens for 19 classes, we only need to classify the event tokens into 3 REALIS types (ACTUAL, GENERIC, or OTHER).

6) Implementation: We train the neural network models using stochastic gradient descent with shuffled mini-batches (batch size = 128) and AdaDelta update rule [13]. The fixed length of the event mention candidate is set to 61. In CNN, we use the window size of [2,3,4,5] to generate conventional filters. We utilize 100 filter size for each convolution window size. As for RNN, we use 256 hidden units and 2-max pooling. The parameters in RCNN is similar to CNN and RNN other than the single conventional filter (with width of 3). In ensemble method, we set the weights of [1,2,1] to CNN, RNN, RCNN respectively for event detection and classification and [2,2,1] for event realis classification.

IV. EXPERIMENT

A. Dataset

We use the following corpora as our training data:

- The training and evaluation data for TAC KBP 2015 event nugget task
- The DEFT Rich ERE English Training Annotation (LDC2015E29, LDC2015E68, LDC2015E31)

We use the evaluation data for TAC KBP 2016 as our development data. Since the TAC KBP 2017 event nugget task defines a smaller set of event types than TAC KBP 2015 and the DEFT Rich ERE English Training data, we only use event nuggets in these two corpora whose event types are included in the evaluation this year. Considering the imbalance of training data, 10 negative samples are choosed randomly for training.

B. Evaluation

We submitted three runs to the Event Nugget Detection evaluation this year. The runs are different in the type of neural networks used in the two components: event detection and classification (called TYPE) and event realis classification (called REALIS). The configuration of the runs are presented in Table 1.

TABLE I

MODELS FOR DIFFERENT RUNS

Runs	TYPE	REALIS
zy1	RNN	CNN
zy2	Ensemble model	Ensemble model
zy3	CNN	CNN

Due to the poorer performance of system zy3, we only present the performance of the first two systems. The performance on the development data and official evaluation data for English is presented in Table 2 and Table 3 respectively. These tables include the precision, recall and F1 scores for event detection (plain), event classification (type), realis classification (realis), type and realis classification (all). All the scores are computed by the official scorer.

TABLE II

PERFORMANCE ON THE DEVELOPMENT DATA

Run	Attri	Micro Average			Macro Average		
		Prec	Rec	F1	Prec	Rec	F1
zy1	plain	56.96	59.08	57.86	53.50	55.67	54.56
	type	46.13	48.07	47.08	43.30	45.36	44.30
	realis	43.79	45.63	44.69	40.84	42.44	41.62
	all	35.73	37.23	36.46	33.08	34.68	33.86
zy2	plain	61.74	54.99	58.17	58.55	51.79	54.96
	type	52.36	46.64	49.33	49.59	44.10	46.68
	realis	48.43	43.14	45.63	45.23	40.20	42.57
	all	41.24	36.74	38.86	38.55	34.39	36.35

TABLE III

PERFORMANCE ON THE OFFICIAL EVALUATION DATA

Run	Attri	Micro Average			Macro Average		
		Prec	Rec	F1	Prec	Rec	F1
zy1	plain	58.23	46.05	51.43	59.91	47.24	52.83
	type	48.11	38.05	42.49	49.75	39.45	44.01
	realis	43.51	34.41	38.42	45.48	35.75	40.03
	all	35.61	28.16	31.45	37.51	29.60	33.09
zy2	plain	64.29	43.14	51.64	65.33	44.21	52.73
	type	55.22	37.06	44.35	56.12	38.10	45.38
	realis	49.28	33.07	39.58	50.45	33.96	40.59
	all	41.87	28.10	33.63	42.85	28.87	34.50

C. Discussion

As we can see from the tables, there're significant performance drops over micro-average scores between development data and test data. The macro-average weights equally all the classes, regardless of how many documents belong to it. The micro-average weights equally all the documents, thus favouring the performance on common classes. This demonstrates there exists over-fitting especially on common classes.

Furthermore, the recall scores drop much over all the subtasks and systems. First, these drops could be attributed to the differences between development data and test data to a certain extent. Second, from the annotated test data, we find an event mention can belong to more than one event subtypes. However, we only classify each event candidate as an event of specific subtype or a non-type. Third, we assume event mentions to be single tokens and don't consider the event mentions containing multiple tokens.

Another limitation is the encodings of event mention candidates. We simply use pre-trained word embeddings from GloVe in the model initialization. In stead it might be more suitable to train our own word embeddings on a large corpus similar to the TAC KBP corpus such as NYT corpus. This is a commonly used technique in previous works [14], [15]. And we could introduce more features such as named entity tags, event arguments and dependency features inspired by previous works[11], [9].

Last but not least, we could use more data resources such as ACE 2005 considering the smallness of training data and cope with the imbalance of training data.

V. CONCLUSIONS

We develop three systems based on neural networks to participate in Event Nugget Detection task this year. Compared with feature-based approaches, which require complicated feature engineering, our systems use only word embeddings as external resources and can be directly applied to different languages. Compared with previous neural models, our systems combine CNN and RNN to capture both chunk features and long-dependencies in a sequence. Our systems obtain promising performances in the final evaluation but have room to improve.

REFERENCES

[1] D. Ahn, The stages of event extraction, in: Proceedings of the Workshop on Annotating and Reasoning about Time and Events, Association for Computational Linguistics, 2006, pp. 1–8.

[2] H. Ji, R. Grishman, et al., Refining event extraction through cross-document inference., in: ACL, 2008, pp. 254–262.

[3] S. Liao, R. Grishman, Using document level cross-event inference to improve event extraction, in: Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, Association for Computational Linguistics, 2010, pp. 789–797.

[4] Y. Hong, J. Zhang, B. Ma, J. Yao, G. Zhou, Q. Zhu, 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, Association for Computational Linguistics, 2011, pp. 1127–1136.

[5] Q. Li, H. Ji, L. Huang, Joint event extraction via structured prediction with global features., in: ACL (1), 2013, pp. 73–82.

[6] Q. Li, H. Ji, Y. Hong, S. Li, Constructing information networks using one single model., in: EMNLP, 2014, pp. 1846–1851.

[7] Y. Chen, L. Xu, K. Liu, D. Zeng, J. Zhao, et al., Event extraction via dynamic multi-pooling convolutional neural networks., in: ACL (1), 2015, pp. 167–176.

[8] R. Ghaeini, X. Fern, L. Huang, P. Tadepalli, Event nugget detection with forward-backward recurrent neural networks, in: Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers), Vol. 2, 2016, pp. 369–373.

[9] T. H. Nguyen, R. Grishman, Modeling skip-grams for event detection with convolutional neural networks., in: EMNLP, 2016, pp. 886–891.

[10] J. Pennington, R. Socher, C. Manning, Glove: Global vectors for word representation, in: Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP), 2014, pp. 1532–1543.

[11] T. H. Nguyen, R. Grishman, Event detection and domain adaptation with convolutional neural networks., in: ACL (2), 2015, pp. 365–371.

[12] N. Kalchbrenner, E. Grefenstette, P. Blunsom, A convolutional neural network for modelling sentences, arXiv preprint arXiv:1404.2188.

[13] M. D. Zeiler, Adadelta: an adaptive learning rate method, arXiv preprint arXiv:1212.5701.

[14] X. Feng, L. Huang, D. Tang, H. Ji, B. Qin, T. Liu, 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), Vol. 2, 2016, pp. 66–71.

[15] S. Liu, Y. Chen, S. He, K. Liu, J. Zhao, Leveraging framenet to improve automatic event detection., in: ACL (1), 2016.