BUPT-PRIS System for TAC 2017 Event Nugget Detection, Event Argument Linking and ADR Tracks

Jiandong Sun, Xiusen Gu, Chen Ding, Chenliang Li, Yan Li, Si Li, Weiran Xu
Pattern Recognition and Intelligent System Laboratory
Beijing University of Posts and Telecommunications
{sunjd, guxiusen}@bupt.edu.cn

Abstract

In this paper, we present an overview of the BUPT-PRIS System for two KBP tasks and two ADR tasks at TAC 2017. The KBP tasks consist of event nugget detection and coreference task (EN), and event argument extraction and linking task (EAL), while the ADR tasks consist of mentions annotation task and relations annotation task. For KBP tasks, we propose an attention-based joint model with two constraints and two downstream models to tackle EN and EAL tasks. For ADR tasks, we combine birectional LSTMs (Bi-LSTMs) and conditional random fields (CRFs) which are based on character embeddings and word embeddings to tackle the mentions annotation task. Then we employ advanced adversarial training method with piece-wise convolutional neural networks (CNNs) for relations annotation task.

1 Introduction

Text Analysis Conference (TAC) is an evaluation workshop in Natural Language Processing which includes two large tracks in 2017 – one is Knowledge Base Population (KBP) track and the other is Adverse Drug Reactions (ADRs) track. For KBP track, several tasks have been launched such as entity discovery, event extraction, sentiment analysis. Similarly, ADR track includes four tasks, such as mentions annotation (task1) and relations annotation (task2). In this year, we participate in four tasks in both tracks: (1) event nugget task and (2) event argument task in KBP track, and (3) mentions annotation task and (4) relations annotation task in ADR track.

In our system, we adopt a joint model proposed by (Nguyen et al., 2016) to tackle the event nugget detection and event argument extraction tasks in KBP track. And then we use two downstream models to address the event REALIS detection and event nugget coreference problems. Figure 1 shows the system architecture of our models. As shown in Figure 1, we use the Stanford CoreNLP to preprocess the document firstly. Secondly we use an attention-based joint model to extract the event nugget and event argument. Then a support vector machine based classifier is used to detect the REALIS type of event. Finally, we use a rule-

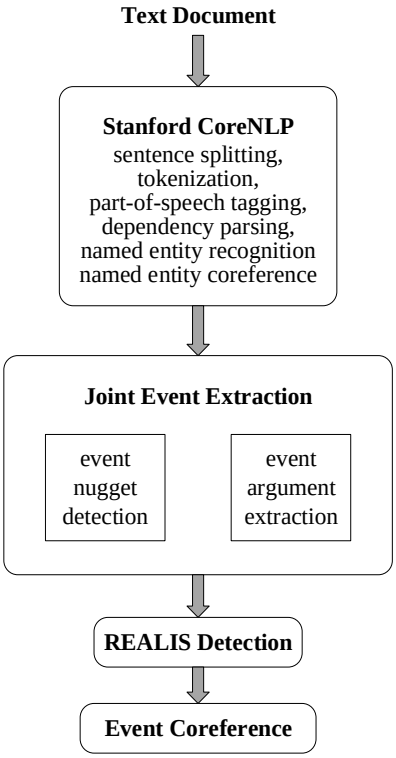

Figure 1: Overview of our TAC2017 system for the TAC KBP 2016 tasks

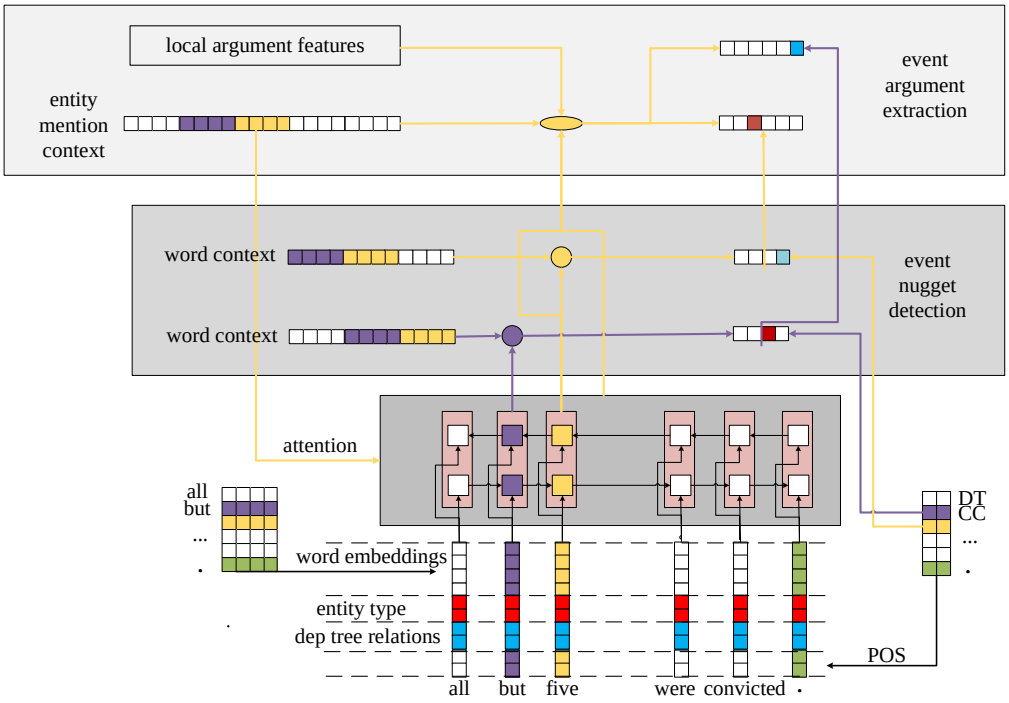

Figure 2: The architecture of our attention based joint model

based model to tackle the event coreference problem.

For ADR track, we employ a pipeline approach for mentions annotation task and relations annotation task. First, we extract mention-style tokens including six types: AdverseReaction, Severity, DrugClass, Negation, Animal and Factor. And then we propose a relations annotation model to identify the relations between AdverseReactions and related mentions (i.e., Negated, Hypothetical and Effect). Both mention and relations annotation tasks are challenging because scarce training data limits the usage of supervised learning methods like CRF, BiLSTM, BiLSTM-CRF, etc. Poor results of mentions annotation also deteriorate relations annotation models performance.

Mentions annotation task in ADR track can be seen as a sequential tagging problem. Conventional sequential labeling algorithms, such as Hidden Markov Models (HMMs) (Rabiner and Juang, 1986), Maximum entropy Markov models (MEMMs) (McCallum et al., 2000), and Conditional Random Fields (CRFs) (Lafferty et al., 2001), rely on handcrafted features to infer the label sequences. By modeling the unary potential functions of a CRF as NN model, recent work (Huang et al., 2015) has combined the benefits of neural networks with CRF. Based on their model, we use Bi-LSTMs to extract features from the input sequences, and a CRF layer to infer the output sequences. Similarly, we treat relations annotation in ADR track as a text classification problem. Previous works have tried to incorporate rich linguistic structures and semantic information into sentence representation, such as position feature (Zeng et al., 2014), bidirectional local information (Cai et al., 2016) and tree structure (Miwa and Bansal, 2016). Our system employs an adversarial training (AT) method adapted from Wu et al. (2017). AT generates continuous perturbations which are added up with word embeddings. As a way of regularizing the classifier, adversarial training can improve model robustness and fit in small training dataset.

The remainder of this paper is organized as follows. Section 2 and Section 3 describe our system for KBP track. Specifically, we describe our joint model for event nugget detection and event argument extraction tasks in Section 2. In Section 3, we present our REALIS detection model and event coreference model. Then, we introduce our models for mentions annotation and relations annotation tasks in ADR track in Section 4 and Section 5 respectively. Finally, we show some experimental

results of both tracks in Section 6, and futhermore offer conclusions in Section 7.

2 Joint Event Extraction

In this section, we introduce our model for event nugget extraction and event argument extraction. Our works are based on the JointEE model mentioned in (Nguyen et al., 2016). Different from JointEE, we introduce an attention machenism and two constraints to improve the accurancy of event nugget detection and event argument extraction tasks. Figure 2 show the architecture of our attention based joint model. As shown in figure 2, we divide the event nugget detection and event argument extraction processing into three phases: sentence encoding, event nugget identification and subtyping, and event argument extraction.

2.1 Sentence Encoding

In the encoding phase, we use a bidirectional Gated Recurrent Unit (GRU) to encode the sentence. Specifically, we first transform each token w_i into a real-valued vector $\boldsymbol{x}_i$ using the concatenation of the following four vectors:

1. The word embedding vector of w_i. We use the google pretrained word vectors which trained on word2vec (Mikolov et al., 2013).

2. The binary vector for the entity type of w_i. The dimension of this vector is equal to the number of possible entity types.

3. The binary vector for the dependency feature. The dimension of this vector corresponds to the possible relations between words in the dependency trees. The value at each dimension of this vector is set to 1 only if there exists one edge of the corresponding relation connected to w_i in the dependency tree of sentence.

4. The binary vector for the part-of-speech (POS) of w_i. The dimension of this vector is equal to the number of possible POS tags.

The transformation from the token w_i to the vector $\boldsymbol{x}_i$ essentially converts the input sentence W into a sequence of real-valued vectors $X = (\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_n)$, to be used by recurrent neural networks to learn a more effective representation. The input sequence X is then fed into a bidirectional GRU to get the representations of the sentence. At each time step i, we compute the hidden vector $\boldsymbol{h}_i$ based on the current input $\boldsymbol{x}_i$ and the pervious hidden vector $\boldsymbol{h}_{i-1}$. This recurrent computation is done over X to generate the hidden vector sequence $(\boldsymbol{h}_1, \boldsymbol{h}_2, ..., \boldsymbol{h}_n)$, denoted by $BiGRU(\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_n) = (\boldsymbol{h}_1, \boldsymbol{h}_2, ..., \boldsymbol{h}_n)$.

2.2 Event Nugget Extraction

Following the sentence encoding stage, we first compute the feature representation vector R^i_{trg} for w_i using the concatenation of the local context vector for w_i and $\boldsymbol{h}_i$. Specifically, the local context vector is generated by concatenating the vectors of the words in a context window d of w_i. Then we feed R^i_{trg} into a feed-forward neural network FFN_{trg} with a softmax layer in the end to compute the probability distribution P^i_{trg} over the subtypes of trigger. Finally, we use $t_i = \arg\max P^i_{trg}$ to get the trigger type t_i for token w_i.

Different from the model mentioned in (Nguyen et al., 2016), we use a POS constaint to imporve the accurancy for event nugget detection. Specificially, we find not all the words are possible to be event trigger. For example, conjunctions and numerals are impossible to be event trigger but verbs are on the contrary. We use the constraint when extract triggers: if a token is conjunction or numeral, the corresponding trigger type should be "Other".

2.3 Event Argument Extraction

In the event argument extraction stage, we first check if the predicted trigger subtype t_i in the previous stage is "Other" or not. If yes, we simply set r_{ij} (the role of entity mention ent_j w.r.t. token w_i) to "Other" for all $j = 1$ to k and go to the next stage immediately. Otherwise, we predict the argument role r_{ij} for each entity mention ent_j with the head index j_k.

First, we generate the representation vector $\boldsymbol{e}_j$ for ent_j by concatenating the features in Table 1. Secondly we feed $\boldsymbol{e}_j$ into a feed-forward neural network to get a distributed representation $\boldsymbol{e}^{dis}_j$ for ent_j. Then we use two ways to predict the event role of ent_j. For the first model, we just concatenate $\boldsymbol{e}^{dis}_j$ and $\boldsymbol{h}_i$, and finally feed the result into a feed-forward neural network FFN_{arg} with a softmax layer to predict the event argument. For the second model, we use an attention machanism to get sentence information with respect to ent_j:

$$\boldsymbol{c}_j = \sum_i \alpha_{ji} \boldsymbol{h}_i \quad (1)$$

Feature	Description
Local context vector of ent_j	The concatenation of the vectors of the words in a context window of w_{j_k}
Entity type	The entity type of ent_j
Dependency feature	The dependency feature of w_{j_k}
POS feature	The POS of w_{j_k}

Table 1: Features for event argument extraction.

$$\alpha_{ji} = \frac{\boldsymbol{e}_j^{dis^T} W \boldsymbol{h}_i}{\sum_{i'} \boldsymbol{e}_j^{dis^T} W \boldsymbol{h}_{i'}} \quad (2)$$

We finally concatenate the $\boldsymbol{c}_j$, $\boldsymbol{e}_j^{dis}$ and $\boldsymbol{h}_i$ to a vector. Like event nugget detection, we use a feed-forward neural network with softmax to predict the event argument.

As the POS constraint we use in event nugget detection, we also use a trigger constaint to imporve the accurancy of the event argument extraction. The constraint of event argument with regards to trigger subtype can be seen in *Rich ERE Annotation Guidelines* .

3 REALIS Detection & Event Coreference

For REALIS detection, we use a one-to-rest support vector machine to identify the REALIS type. And we identify event coreference with three rules:

- *Rule 1.* A pair of events are coreferential only if two events have the same subtypes.
- *Rule 2.* A pair of events is coreferential only if two event triggers have the same lemmatized forms.
- *Rule 3.* A pair of events is coreferential only if two events have the same REALIS types.

Following the three rules, we identify a pair of events which conforming to all the three rules as coreferential events.

4 Mentions Annotation

In this section and the following section, we introduce our models of mentions annotation task and relations annotation task in ADR track respectively. Figure 3 shows the network structure which has an input layer x, hidden layer h and output layer y. Fig 3 illustrates a mentions annotation system in which each word is tagged with one label from the following: other(O), AdverseReaction (Adver), Severity (Sev), DrugClass (Drug), Negation (Neg), Animal (Ani), and Factor(Fac). The sentence "*Exclusive of an uncommon, mild injection site reaction, no other adverse reactions have been reported.* " is tagged as "*O O O O O S-Sev B-Sever M-Sever E-Sever O O O O O O O O O* ", where *B-,M-,E-* tags indicate beginning, middle and ending position of mentitions respectively, and *S-* tag indicates the single word mention.

4.1 Input Embeddings

Input embeddings are concatenated by word vectors and character-based embedddings of words. Word vectors are looked up from an embedding matrix which is pretrained using word2vec. And character-based embeddings are hidden states from the bidirectional LSTMs layer.

4.2 Bi-LSTM Encoder

The LSTM memory cell is implemented as the following:

$$i_t = \sigma(W_{xi}x_t + W_{hi}h_{t-1} + W_{ci}c_{t-1} + b_i) \quad (3)$$

$$c_t = (1 - i_t) \odot c_{t-1} + i_t tanh(W_{xc}x_t + + W_{hc}h_{t-1} + b_c) \quad (4)$$

$$o_t = \sigma(W_{xo}x_t + W_{ho}h_{t-1} + W_{co}c_t + b_o) \quad (5)$$

$$h_t = o_t \odot tanh(c_t) \quad (6)$$

where σ is the element-wise sigmoid function, and $\odot$ is the element-wise product.

For a given sentence $(x_1, x_2, ..., x_n)$ containing n words, the LSTM layer computes representations of $\overrightarrow{l_t}$ and $\overleftarrow{r_t}$ containing left and right context

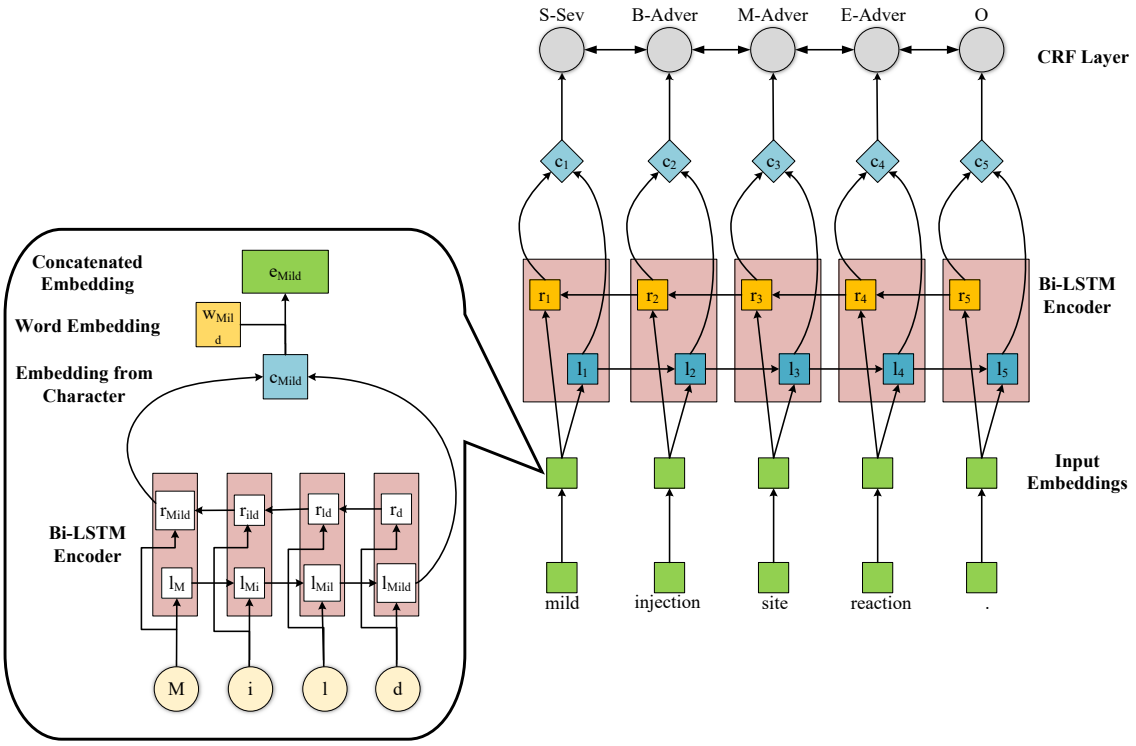

Figure 3: Main architecture of mentions annotation network. Input embeddings of the d-dimensional vectors are concatenated by word vectors and character-based embedddings of words. Concatenated embeddings are given to a bidirectional LSTM to get a representation of the word i in its context, c_i.

information of each word, respectively. Then two kinds of representations are concatenated to form the input of Bi-LSTM $c_t = [\overrightarrow{l_t}, \overleftarrow{r_t}]$.

4.3 CRF Tagging Models

We model tagging decisions jointly using a CRF layer (Lafferty et al., 2001). For an input sentence $X = (x_1, x_2, ..., x_n)$ and a sequence of predictions $y = (y_1, y_2, ..., y_n)$, its score is defined as:

$$s(X, y) = \sum_{i=0}^{n} A_{y_i, y_i+1} + \sum_{i=1}^{n} P_{i,y_i} \qquad (7)$$

where $A_{i,j}$ represents the score of a transition from the tag i to j. $P_{i,j}$ represents the score of the j^{th} tag of the i^{th} word in a sentence.

5 Relations Annotation

5.1 Word Embedding

We define relations annotation task as predicting the relation that exists in a particular entity pair. We use the model mentioned in Wu et al. (2017). For each sentence x_i with an entity pair (m_1, m_2), we first use pretrained word embeddings to project each token into d_w-dimensional space. The out-of-vocabulary words are represented as unknown symbols "UNK".

5.2 Position Embedding

Note that most CNN-based models (Kim, 2014; Santos et al., 2015; Collobert et al., 2011) depend on position embeddings p_i that encode position information and calculate in continuous space. Position embeddings are initialized using normal distribution and fixed on training and test stages. We set the max distance from an entity as 90.

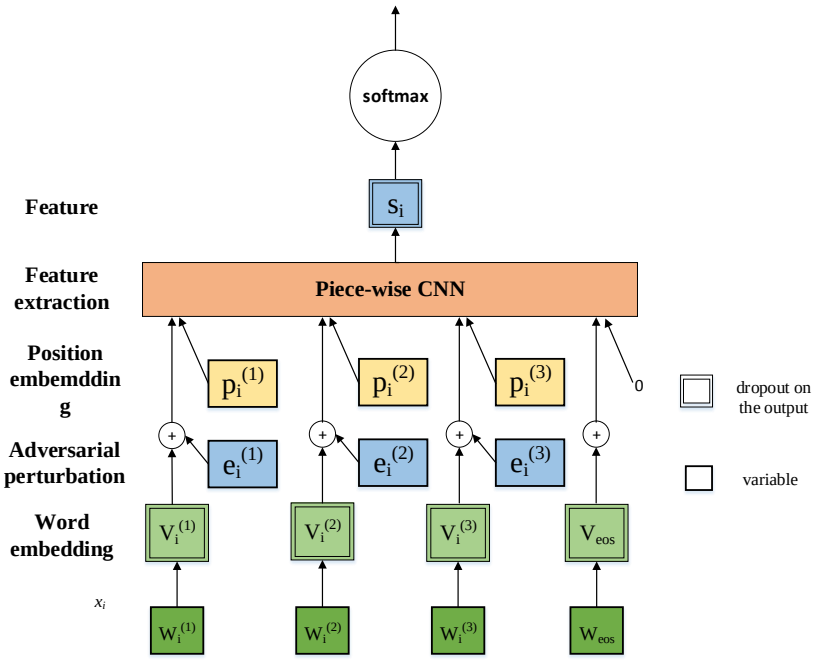

Figure 4: The model architecture of ADR relations annotation task. e_i denotes the adversarial perturbation. We apply dropout to both word embedding layer of x_i and sentence feature vector s_i.

5.3 Feature Extraction

We adopt piece-wise CNN (Zeng et al., 2014) to encode the sentence with annotated entity pair. Entity pair (m_1, m_2) splits the sentence into three parts and convolution operates on each part of the sentence. After Max-pooling, we concatenate three pooling results and get the sentence feature s_i. We add dropout operation to word embedding layer and sentence feature representation of 0.5. A fully-connected network and softmax functiona are performed to get prediction results. The probability of relation r is:

$$P(r|X;\theta) = \text{softmax}(\boldsymbol{A}s_r + \boldsymbol{b}), \quad (8)$$

where **A** is the projection matrix and **b** is the bias term. Small adversarial perturbations e_{adv} are added to word embedding V and optimizes as Eq.(7) .

$$L(X;\theta) = -logP(r|X;\theta). \quad (9)$$

$$L_{adv}(X;\theta) = L(X + e_{adv};\theta) \quad (10)$$

$$e_{adv} = \arg\max_{||e\leq\epsilon||} L(X + e;\hat{\theta}). \quad (11)$$

With respect to Eq.(8), it is computationally intractable for neural networks, so an approximate way proposed by Goodfellow et al. (2014) is applied:

$$e_{adv} = \epsilon g/||g||, \; where \quad (12)$$

$$g = \nabla_V L(X;\hat{\theta}) \quad (13)$$

Here V denotes the word embeddings of all the words in X. $||g||$ denotes the norm of gradients over all the words from all the sentences in X. ϵ represents the coefficient of perturbations.

6 Experiments

6.1 Data

The data provided in TAC 2017 can be devided into two parts: one for KBP track and another for ADR track. For the EN and EAL tasks in KBP track, we use LDC2017E02 and ACE2005 as training datasets. In our systems, 80% of the documents are used for model training, and the remaining 20% are used for development, specifically for tuning hyper-parameters in our model. Note that we only evaluate on the 18 event types and subtypes selected by the KBP 2017 organizers. For ADR track, the original data has 101 annotated labels and 2208 unannotated labels. We preprocess the original data with several steps, such as

Metric	Run1			Run2			Run3		
	Prec	Rec	F1	Prec	Rec	F1	Prec	Rec	F1
plain	67.95	32.74	44.19	72.91	26.78	39.17	65.38	29.69	40.84
mention type	58.92	28.39	38.31	65.44	24.04	35.16	54.24	24.64	33.88
REALIS	46.36	22.34	30.15	49.76	18.28	26.73	46.15	20.96	28.83
mention type + REALIS	39.92	19.24	25.96	44.48	16.34	23.90	38.66	17.56	24.15

Table 2: Event nugget detection performance on the KBP 2017 official evaluation.

Metric	Run1	Run2	Run3
B^3	28.66	26.19	25.53
$CEAF_e$	28.64	26.19	24.90
$CEAF_m$	30.17	27.71	26.40
MUC	19.30	18.00	18.54
$BLANC$	13.56	11.97	12.47
Average	**22.54**	20.59	20.34

Table 3: Event coreference resolution performance on the KBP 2017 official evaluation.

extracting sentences with tagged mentions and relations, eliminating the mentions that span in multiple incontinuous tokes and filtering the relations existing in two sentences. These operations result in 6449 samples containing at least one mention and 2353 relation sentences for training and developing finally.

6.2 Results and Analysis

6.2.1 Event Nugget Extraction and Coreference

Table 2 shows the results of event nugget detection in KBP track, which includes three runs for our systems. For run1 and run2, we use attention mechanism to get sentence information when identify the event argument roles. Run3 is the model without using attention mechanism. When examining the result of each type, we find that events of type Contact, Manufacture, Movement and Transaction have lower performance. One source of precision error can be attributed to the difference of trigger label distribution in the training set and evaluation set. For example, in training set the proportion of trigger with type Movement is about 7%, but in evaluation the proportion is about 18%. One source of recall error can be attributed to the difficulty of correctly extracting features in discussion forum documents owing to their informal writing style. A source of both precision and recall error can be attributed to the error accumulation from entity mention identification and coreference.

Table 3 shows the official results of event coreference resolution in KBP track. For event coreference, a source of precision error is that we do not use argument information. A source of recall error is from *Rule 2* in section 3. According to *Rule 2* , some coreferential events which have triggers with different lemmatized form can not be detected.

6.2.2 Event Argument Extraction and Linking

The official results of event argument extraction and linking in KBP track are shown in Table 4. As event nugget detection, a source of event argument extraction error is the difference of argument role distribution between training set and evaluation set. Another source of event argument extraction error is from the model. Because we use a joint model to extract event nuggets and event arguments, the performance of both EA tasks is influced by event nugget detection. Moreover, the bad performance of event coreference resolution is another important source of the event argument linking error.

6.2.3 Mentions Annotation and Relations Annotation

Table 5 and table 6 are our evaluation results of mentions annotation and relations annotation in ADR track. Note that **P** represents the precision, **R** represents the recall, $\mathbf{F_1}$ is calculated by $\frac{2PR}{P+R}$.

Run	ArgP	ArgR	ArgF1	ArgScore	LinkScore
Run1	17.50	2.55	4.45	0.75	0.32
Run2	23.73	2.59	4.66	1.18	0.45
Run3	15.45	2.27	3.95	0.58	0.35

Table 4: Event argument exrtraction and linking performance on the KBP 2017 official evaluation.

Results	P	R	F_1
Exact (+type)	0.4047	0.1181	**0.1829**
Exact (-type)	0.4047	0.1181	0.1829

Table 5: Mentions annotation task results. Bold result means PRIMARY.

Results	P	R	F_1
Full (+type)	0.0097	0.0038	**0.0055**
Full (-type)	0.0135	0.0054	0.0077
Binary (+type)	0.0163	0.0068	0.0096
Binary (-type)	0.0196	0.0081	0.0115

Table 6: Relations annotation task full results. Bold result means PRIMARY.

7 Conclusion

This paper presents BUPT-PRIS System in the 2017 TAC evaluation. In this system, we propose four models to tackle event nugget detection and coreference task, event argument extraction and linking task, mentions annotation task and relations annotation task respectively. For the first two tasks which are part of KBP track, we employs a joint model which integrates trigger information and arguments information for the tasks. In addition, we incorperate attention mechanism and two constraints to improve the performance of our system. For the remaining two tasks which belong to ADR track, adapt entity extraction to mentions annotation and apply adversarial training model to relations annotation. However, There are still some deficiencies in our system. For example, we only use an end-to-end model to tackle the event nugget detection and event argument extraction, which would be worth incorporating more classifiers to furthre improve the system performance. Another challenge is that our models are under great influence of lack and imbalance of data, and we will integrate semi-supervised methods to the existed architecture to tackle these problems.

Acknowledgements

This work was supported by Beijing Natural Science Foundation (4174098), National Natural Science Foundation of China (61702047) and the Fundamental Research Funds for the Central Universities (2017RC02).

References

Amit Bagga and Breck Baldwin. 1998. Algorithms for scoring coreference chains 5:563–566.

Rui Cai, Xiaodong Zhang, and Houfeng Wang. 2016. Bidirectional recurrent convolutional neural network for relation classification. In *ACL (1)*.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research* 12(Aug):2493–2537.

Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. 2014. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572* .

Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional lstm-crf models for sequence tagging. *arXiv preprint arXiv:1508.01991* .

Yoon Kim. 2014. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882* .

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

Xiaoqiang Luo. 2005. On coreference resolution performance metrics. In *HLT/EMNLP 2005, Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing, Proceedings of the Conference, 6-8 October 2005, Vancouver, British Columbia, Canada*. pages 25–32.

Andrew McCallum, Dayne Freitag, and Fernando CN Pereira. 2000. Maximum entropy markov models for information extraction and segmentation. In *Icml*. volume 17, pages 591–598.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, Curran Associates, Inc., pages 3111–3119. http://papers.nips.cc/paper/5021-distributed-representations-of-words-and-phrases-and-their-compositionality.pdf.

Makoto Miwa and Mohit Bansal. 2016. End-to-end relation extraction using lstms on sequences and tree structures. *arXiv preprint arXiv:1601.00770* .

Thien Huu Nguyen, Kyunghyun Cho, and Ralph Grishman. 2016. Joint event extraction via recurrent neural networks. In *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. pages 300–309.

Lawrence Rabiner and B Juang. 1986. An introduction to hidden markov models. *ieee assp magazine* 3(1):4–16.

M. Recasens and E. Hovy. 2011. Blanc: Implementing the rand index for coreference evaluation. *Natural Language Engineering* 17(4):485–510.

Cicero Nogueira dos Santos, Bing Xiang, and Bowen Zhou. 2015. Classifying relations by ranking with convolutional neural networks. *arXiv preprint arXiv:1504.06580* .

Marc Vilain, John Burger, John Aberdeen, Dennis Connolly, and Lynette Hirschman. 1995. A model-theoretic coreferenc e scoring scheme. In *Conference on Message Understanding, Muc 1995, Columbia, Maryland, Usa, November*. pages 45–52.

Yi Wu, David Bamman, and Stuart Russell. 2017. Adversarial training for relation extraction. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. pages 1779–1784.

Daojian Zeng, Kang Liu, Siwei Lai, Guangyou Zhou, Jun Zhao, et al. 2014. Relation classification via convolutional deep neural network. In *COLING*. pages 2335–2344.