TEA_ICT System for Event Nugget Detection at TAC KBP 2015

Siying Li, Zhiyuan Ji, Pan Du
CAS Key Laboratory of Network Data Science and Technology
Institute of Computing Technology
Chinese Academy of Science
lisiying@software.ict.ac.cn

Abstract

In this paper, we will describe the TEA_ICT system that we established for event nugget detection task in TAC KBP 2015. To complete this task, we developed a model including pre-processing, post-processing and two supervised steps. We tried to make our system find every instance of a mention and identify its types/subtypes. Pre-processing provides data for the following training steps. In the supervised steps, we use CRF (Conditional Random Field) model to extract the event trigger word and SVM (Support Vector Machine) to identify the event mention into types/subtypes and three Realis values. Finally, we aggregate the two individual results as the final result.

1 Introduction

The Event Detection task at TAC KBP 2015 aims to identify the explicit mentioning of Events in text[1]. This year, we only participated in the Event Nugget Detection task (EN Task1).

The Event Nugget Detection task requires that participates must identify all relevant Event Mention instances within each sentence and the types/subtypes of the mention taken from the Rich ERE Annotation Guidelines[2]. In total, there are 8 types and 38 subtypes. In addition, systems must identify three REALIS values (ACTUAL, GENERIC, OTHER), which are also described in the Rich ERE Annotation Guidelines.

In order to meet all the requirement, we propose a system which includes four parts to finish this task: pre-processing for preparing event mentions for training, trigger extraction using CRF-based and rule-based methods, types/subtypes identification and REALIS values identification using SVM model, and post-processing.

The paper is organized as follows. Section 2 describes our approach to do this task including the preparation process of training data, our idea and our model. Section 3 shows the results of our system. In Section 4, we conclude the paper and our work in TAC KBP 2015. Related references are the lasts part.

2 Our Approach

In this part, we will describe the whole process of our approach in detail.

The architecture of our system can be described as Figure 1.

2.1 Data Pre-Processing

The data pre-processing mentioned here is mainly about preparing for training data.

We used the ACE 2005 Multilingual Training Corpus (LDC2006T06) to train our model at first. The annotation of this corpus includes event mentions and entities. As our system caring more about event mentions and triggers, we only extract trigger words, mention types/subtypes and REALIS values as our original training data.

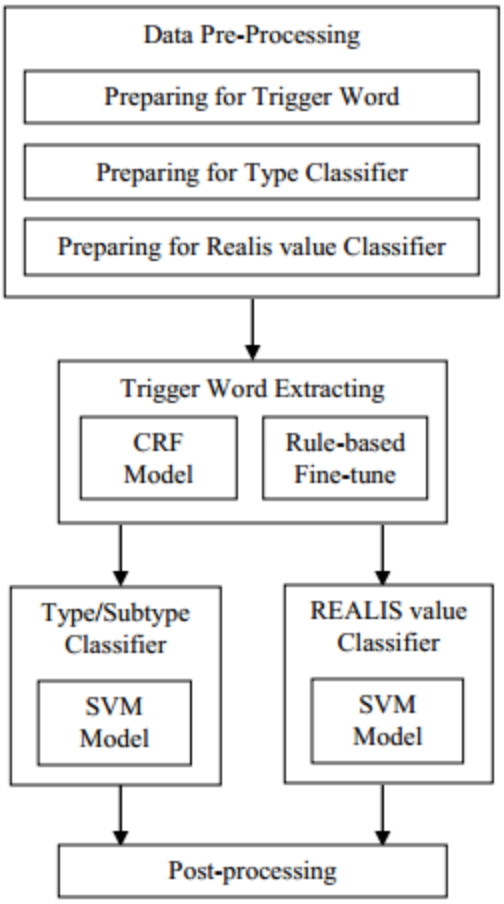

Figure 1: Architecture of TEA_ICT System

When using the data extracted from LDC2006T06, we found a few tagging mistakes. Besides, we observed that the TAC KBP 2015 Event Nugget Training Data Annotation V2 (LDC2015E73) updated in August only focus on event mentions and discarded all the entity information. So we finally chose LDC2015E73 as our training data. Every annotated document has its corresponding source file making it easy to extract the relevant mention information for the training process.

After extracting trigger words from training data, we use Stanford Log-linear Part-Of-Speech Tagger (POS Tagger) to segment word in source file and mark the POS Tagger of every segmented word. Now, the tagged word sequences have been adapted to train the CRF model.

As we have already got every trigger in training data, we can easily extract the whole sentence which contains a trigger word. The entire sentences are supposed to construct feature spaces that are used to train SVM model. In addition, we should identify types/subtypes and REALIS values of every instance of event mention. Thus two classifiers are trained separately by the sentence features.

For validation data, we can get two kinds of data, they are source data and token-format data. We use Stanford Log-linear Part-Of-Speech Tagger (POS Tagger) to simply tag the token sequences making it more adaptive to the following process.

2.2 Trigger Word Extracting

Extracting trigger words is not simply matching some special words. Primarily, it is a binary classification problem. In training data, we labelled every segmented word with "I" or "O". Label "I" represents that the word is a trigger word of a part of trigger words, while "O" represents that the word is not a trigger word. Then Trigger Word Extracting process turns into a binary classification problem. If we can predict every word in validation data whether it belongs to "I" or "O", we can easily find the trigger word.

But it is difficult for us to correctly put them into two classes only considering the words themselves. Thus we must consider the structure of the whole sentence. Finally, we chose CRF model to extract trigger words.

CRF is a supervised model usually used for structured prediction in pattern recognition and machine learning. An ordinary classifier predicts a label for a single sample without regard to "neighboring" samples, while a CRF can take context into account. In our case, we use CRF model to predict sequences of labels for input validation sequences.

The first step is to train CRF model. We directly used the open source toolkit CRF++-0.53 to implement the trigger extracting process. When training CRF, the most important thing is to construct the training data. The structure of CRF training data are formally described in Table 1.

Every segmentation of a sentence is required to be put on the left column of a text file. The middle column is the tag generated by Stanford POS-Tagger system. For example, NN represents for noun, VBP for past tense of a verb. The right column is the label we defined. As we have mention in the beginning of this part, we put "O"

for the word that is not a trigger in training data and "I" for a trigger.

Word	Tag	Label
He	PRP	O
shot	VBP	I
the	DT	O
soldier	NN	O
dead	RB	I

Table 1: Data format for training CRF model

We set a window size of 5, so that CRF can learn a word sequence with the length of 5 in one time. If we set a larger window size, the accuracy of CRF may get higher, but the feature size will explosively increase. In order to balance accuracy and time cost, we finally decided 5 as our window size.

During training CRF model, we cannot ignore the overfitting problem. For instance, some words (i.e. formal) only appears a little time, but more can half are defined as trigger words by the annotators. When testing, CRF labels almost all words of this kind as trigger words. So in the toolkit, we set a large regularization term to avoid the overfitting problem.

After the above configure settings, we can train our CRF then apply the model to the validation data. The structure of validation data is almost like training data. The structure of validation has the same left and middle columns with that of training data. It only has two columns for the reason that the last column is for prediction.

However, the number of "I" labelled by CRF is relatively low. Only using the original CRF leads us facing the situation that we can get a high precision but an extremely low recall. In order to improve the recall, we use the CRF with probability and add some rule-based methods. After adding probability to CRF, we get a result like Table 2.

Word	Tag	Label	Probability
He	PRP	O	0.999714
shot	VBP	I	0.908648
the	DT	O	0.999998
soldier	NN	O	0.987736
dead	RB	O	0.911098

Table 2: Data format for training CRF model

We can see that the word "dead" labels "O" by CRF, but the probability is low. In addition, we sum up a list which contains the obvious trigger word. If a word has been labelled "O" with a low probability and can be sought out in the list (just use the rule-based method), it will be labelled with "I" immediately.

2.3 Classifier

Fundamentally speaking, the task of identifying types/subtypes and REALIS values is a classification problem. As for classification, the most popular machine learning model is SVM. SVM is a supervised learning classifier. Its classification results often depend on the features we choose for SVM. The whole process of classification including feature selection and classification will be described in the following two subsections.

2.3.1 Types Identification

According to our training data, a sentence containing trigger words is not equal to an event mention. A sentence which contains 2 or more triggers may mention different types/subtypes. For example, in the sentence "He shot the soldier dead.", both [conflict. ATTACK] and [life. DIE] are events. Thus when identifying types, a sentence may belong to different classes. So we cannot simply treat the original sentence as all the features applied for SVM. To solve this problem, we select the following dimensions as type classification:

- Every segmented words
- Higher weight of triggers than other words

In addition, we only increase one weight of triggers in one sentence at one time. Thus the trigger become more important than the others. If triggers are extracted correctly, the accuracy of SVM is more than 90%.

2.3.2 Types Identification

Another usage of classification is identifying REALIS values. Event mentions will refer to ACTUAL (events that actually occurred); GENERIC (events that are not specific events with a (known or unknown) time and/or place); or OTHER (which includes failed events, future events, and conditional statements, and all other non-generic variations).

Besides the semantic meaning of a sentence, the REALIS also cares about the tense of a sentence. The tense of a sentence can help identify whether an event has occurred. To identify REALIS values, the selection of features is listed below:

- Every segmented words
- POS-Taggers of all words

Similar to types/subtypes identification, the accuracy of only identifying REALIS values also reaches 90%.

2.4 Post-Processing

After the above efforts, we can almost get a result of several parts. And the result we got may have some obvious mistakes. Post-processing is designed for correcting some mistakes and integrate the several result parts.

3 Result

We submitted a total of four runs. TEA_ICT2 is the most basic result of only applying CRF and SVM. It contains some mistakes and the classification model for types/subtypes appears to be wrong that we get an extremely low precision and recall. TEA_ICT1 improve our models and feature selection methods. After re-training all the models, we get a relatively high improvement. The evaluation indicator of the 2 runs are listed in Table 3.

Run ID		Prec	Rec	F1
TEA_ICT1	Plain	59.08	52.11	55.38
	Type	45.59	40.21	42.73
	Realis	40.86	36.05	38.30
	All	31.65	27.92	29.67
TEA_ICT2	Plain	59.89	38.14	46.60
	Type	2.54	1.62	1.97
	Realis	40.07	25.52	31.18
	All	1.73	1.10	1.35

Table 3: Comparison of our runs

4 Conclusion

This paper describes our efforts and the complete process when doing this Event Nugget Detection task in TAC KBP 2015. We applied two supervised model to accomplish the requirements of the task of this year. Four steps were designed by us and we finally got a relatively reasonable result.

Until now, the official evaluation results have been provided. We still have a lot lifting space and following research to do in the future.

References

TAC KBP Event Detection and Coreference Tasks for English. Available: http://cairo.lti.cs.cmu.edu/kbp/2015/event/Event_Mention_Detection_and_Coreference-2015-v1.1.pdf

Rich ERE Annotation Guidelines Overview V4.1. Available: http://cairo.lti.cs.cmu.edu/kbp/2015/event/summary_rich_ere_v4.1.pdf

Mitamura T, Yamakawa Y, Holm S, et al. Event Nugget Annotation: Processes and Issues[C]//Proceedings of the 3rd Workshop on EVENTS at the NAACL-HLT. 2015: 66-76.

Roth B, Strubell E, Silverstein K, et al. Minimally Supervised Event Argument Extraction using Universal Schema[J].

Sammons M, Song Y, Wang R, et al. Overview of UI-CCG Systems for Event Argument Extraction, Entity Discovery and Linking, and Slot Filler Validation[J]. Urbana, 2014, 51: 61801.

X. Cheng, B. Chen, R. Samdani, K. Chang, Z. Fei, M. Sammons, J. Wieting, S. Roy, C. Wang, and D. Roth. Illinois cognitive computation group ui-ccg tac 2013 entity linking and slot filler validation systems. In TAC, 2013