Report: Three Perceptrons Accomplish Event Extraction Task

Xingyuming Liu
Peking University
liuxym@pku.edu.cn

Abstract

This is an assignment report about event extraction tasks. We devide this task into four stage: trigger detection, trigger classification, argument candidates extraction and argument classification. Trigger detection, trigger classification and argument classification are accomplished by three learnable models based on perceptron. And the stage of argument candidates extraction is completed by named entity recognition (NER) and constituent parsing. In this report, we described the implementation of our models and the design of the pipeline. And the results are presented and analyzed in the report.

1 Pipeline Design

We devide the event extraction task into four stages (see Figure 1). The arrows in the figure indicate Input/Output. Different from the setting in (Ahn, 2006), we don't use a learnable model to achieve argument identification, but directly use NER and consituent parsing to extract argument candidates from the sentence. And the other three modules are implemented by the averaged perceptron.

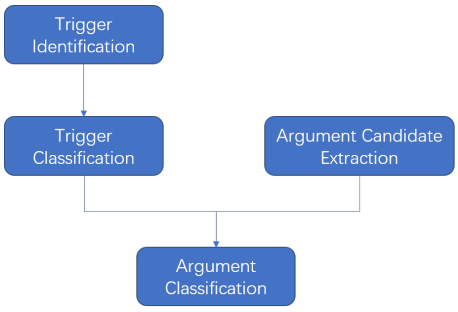

Figure 1: Pipeline of event extraction.

2 Part 1: Trigger Identification and Classification

2.1 Identifying event trigger

We formalize this task as a sequence labeling task, labeling triggers with 1 and other words with 0. This is as same as the setting in Homework 2. We use the average structured perceptron to accomplish this task (the same as Homework). Not much to describe here.

2.2 Classifying event trigger

Trigger classification refers to assigning the correct event types to the identified triggers. We formalize this problem as a simple word classification task. This task is achieved by an average perceptron.

2.3 Features for triggers

We use the following set of features for all configuration of our trigger identification and classification experiments.

- Lexical features: lowercase word, POS tag
- Morphological features: prefix and suffix, containing number, containing hyphen, containing uppercase character
- Left context (2 words): lowercase word
- Right context (2 words): lowercase word

2.4 Experiments

Trigger Identification: The averaged structured perceptron is trained on the training set for 10 iterations and evaluated on the validation and testing set.

Trigger Classification: The averaged perceptron is trained on the training set for 10 iterations and evaluated on the validation and testing set.

Methods	Dataset	Trigger Identification (%)			Trigger Classification (%)			Trigger Identification + Classification (%)		
		P	R	F_1	P	R	F_1	P	R	F_1
Our models	Valid.	80.00	64.97	71.71	73.53	72.28	71.55	72.53	61.18	66.37
	Test.	76.49	61.36	68.10	80.34	76.31	76.66	69.97	58.53	63.74

Table 1: The performance of our models in trigger identification and classification.

Trigger Identification and Classification: The previous two models are combined to complete the task of trigger identification and classification. The second perceptron uses the output of the first perceptron as input.

Evaluation: All evaluations follow the previous work (Li et al., 2013).

2.5 Results

As shown in Table 1, our results are comparable to previous works (Li et al., 2013) (Ahn, 2006). Note that, although dependency features are not used, we still get a good performance on trigger classification, since the event types of triggers depend largely on the feature of the triggers.

However, we believe that there is still room for improvement in trigger identification, and it should perform better if dependency features are considered in the model.

3 Part 2: Argument Extraction and Classification

In this part, we implement the extraction of argument candidates and the classification of arguments.

3.1 Argument Candidates

In the dataset, most of the arguments are noun phrases, named entity and adverb phrases. We can use constituent parse and NER to extract argument candidates. In the experiment, we use *CoreNLPParser* and *ne_chunk* provided by *NLTK* to achieve this.

3.2 Argument Classification

Given triggers and argument candidates, argument classification needs to match the argument with the trigger and specify the role of the argument.

(Ahn, 2006) formalize this as a pair classification task to assign a role type for the trigger-argument pair. However, in my mind, it is not a good way to do this task, since it is possible to assign multiple identical roles to an event under this setting. To avoid this, we formalize it as a pair classification task to assign an argument for the trigger-role pair.

And since an event type usually has fixed types of roles, to reduce computation we can just search for patterns seen in the training set. For example, Event type *Movement:Transport* tends to have argument type like *Origin* and *Destination* instead of *Crime*.

Still, we achieve this by an averaged perceptron.

3.3 Features for arguments

We use the following set of features for all configuration of our trigger identification and classification experiments.

- Trigger word: event type of trigger, lemmatized word
- Constituent head word of argument candidate
- Dependency feature: length of dependency path between anchor word and constituent head word of argument candidate

3.4 Experiment

Argument Candidate Extraction: We use *CoreNLPParser* and *ne_chunk* provided by *NLTK* to achieve argument candidate extraction. Noun phrases and adverb phrase are extracted by *CoreNLPParser*, named entities are extracted by *ne_chunk*. We tested different extraction strategies in experiment.

Argument Classification: The averaged perceptron is trained for only 2 iterations. (Due to the variety of types of triggers and arguments, the training cost is high (2 hours/iters). Due to time constraints, we did not complete the expected five iterations.)

3.5 Results

Argument candidates extraction: For adverb phrases, we only extract the maximum adverb phrase. For noun phrases, we tested different extraction depths and finally choose to extract to the depth of 3 for the latter experiment. Combining

Only Parser (depth=1)		Only Parser (depth=2)		Only Parser (depth=3)		Only Parser (depth=4)		Only NER		Combine	
P	R	P	R	P	R	P	R	P	R	P	R
40.86	18.25	57.23	16.55	64.97	16.26	**67.89**	16.10	14.59	18.73	**69.92**	15.40

Table 2: performance of different argument candidate extraction on validation set.

Methods	Datasets	Argument Classification			Argument Candidates Extraction + Classification		
		P	R	F_1	P	R	F_1
Our models	Valid.	42.03	32.59	36.71	7.76	5.10	6.15
	Test.	40.59	34.29	37.18	9.39	6.98	8.00

Table 3: Performance of argument classification.

NER and parser, we are able to correctly extract 69.92% of the arguments.

Note that the recall of the extraction is very low, since we did not adopt any strategies and restrictions to improve the recall. However, this will have little effect on the final performance if the classification model can discard those incorrect arguments during the classification of arguments.

Argument classification: The performance of the model under gold standard argument candidates is shown in the third column of Table 2. As you can see the result is not very good (However it is still comparable with (Hong et al., 2011)).

Overall performance of part 2: The overall performance is terrible (under 10%). This shows that our model does not have enough ability to discard the incorrect argument candidates. A possible improvement is to train not with the gold standard argument, but with the extracted argument candidates. However, this will further increase the training cost.

4 Overall Performance of the Pipeline

The overall performance is shown in Table 4. Althougn we made a great performance on the identification and classification of trigger, the second part of argument classification is not that great.

Method	Dataset	Precision	Recall	F1
Our pipeline	Valid.	3.38	4.35	3.80
	Test.	4.45	4.76	4.60

Table 4: Overall performance of our pipeline.

5 Conclusions

In this assignment, we complete the task of event extraction with three perceptrons. For improvement, more refined argument extraction is required. At the same time, the pipelined paradigm also makes it easy for errors to accumulate. And the information of the later stage cannot be used for the previous stage.

6 Acknowledge

Thanks to Teacher Feng and T.A. Zhang Chen for their help this semester.

References

David Ahn. 2006. The stages of event extraction. In *Proceedings of the Workshop on Annotating and Reasoning about Time and Events*, pages 1–8.

Yu Hong, Jianfeng Zhang, Bin Ma, Jianmin Yao, Guodong Zhou, and Qiaoming Zhu. 2011. Using cross-entity inference to improve event extraction. In *Proceedings of the 49th annual meeting of the association for computational linguistics: human language technologies*, pages 1127–1136.

Qi Li, Heng Ji, and Liang Huang. 2013. Joint event extraction via structured prediction with global features. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 73–82.