Report for EMNLP HW3:
Event Extraction

Abstract

In this homework, the event extraction task is split into 4 subtasks: trigger detection, trigger classification, argument identification, argument classification. I use RNN in the second stage and feature-based Linear Model in other stages.

1 Introduction

Event extraction is a challenging task which requires to combine semantic and syntactic information of a sentence. In last homework, I implement a linear model trained with Perceptron Algorithm for trigger detection. Subsequent models will make prediction based on its results. In this homework, I implement a pipeline containing 4 modules to extract and classify event triggers and arguments. The dataset includes 33 types of triggers and 35 types of arguments. A strength of pipeline structure is that we can use the most appropriate model for each part respectively. One outstanding related work is that (Li et al., 2013) propose a joint framework for the task and achieve SOTA results without relying on external knowledge.

2 Methods

2.1 Trigger Classification

For trigger classification algorithm, I build a RNN and train it with gold standards, considering that the classification relies more on semantic information rather than syntactic information. I use pretrained word embeddings from GloVe (Pennington et al., 2014) as the features of words here. In training time, every word will be given to the model but only the loss of triggers will be backpropogated.

2.2 Argument Identification

I complete the argument identification task in 2 stages. In the first stage, argument candidates are proposed. Given that the argument span in the dataset is relatively long compared with named entities, I choose dependency parsing rather than pure Named Entity Recognition(NER). For every subtree in the dependency tree, the span is taken as an argument candidate. Therefore, the candidate proposal may achieve a better recall. In the second stage, a feature-based linear model predicts whether a candidate is an argument of a given event. The feature selection partly refers to (Li et al., 2013) and details are shown in Table 1. Candidates are used to train to reduce the gap between training and inference. The limitation is that the gold standard arguments excluded by the candidate set will not be used to train the model.

Feature Description
1. token number of the argument candidate
2. unigrams of the argument candidate
3. trigger word, type and subtype
4. the relative position between the argument candidate and the trigger: {before, after, overlap, or separated by punctuation}

Table 1: Description of features for argument identification and classification.

2.3 Argument Classification

After identifying arguments from candidates, they are classified into 35 role types by a Linear Model. The features of argument are the same as what is described in Section 2.2. The model is trained with gold standard data as well as Perceptron Algorithm and tested with results from the last step.

3 Experiment

3.1 Modules

First, I separately build the 4 modules, train the 4 models and test them. For trigger detection, the settings are the same as last homework. I set `n_epoch=40` and `feature_size=9715`. For

	Stage	acc	precision	recall	(macro-)F1
Modules	Trigger Detection	0.962	0.633	0.561	0.595
	Trigger Classification	0.605	0.545	0.537	0.512
	arg identification	0.952	0.441	0.105	0.169
	arg classification	0.439	0.464	0.304	0.327
Pipeline	Trigger Detection	0.936	757	0.797	0.733
	Trigger Classification	0.925	0.423	0.393	0.356
	arg identification	–	0.079	0.068	0.072
	arg classification	–	0.047	0.040	0.042

Table 2: Performance of the 4 stages when the modules are run separately and gathered in a pipeline.

trigger classification, I choose the `100d` GloVe embeddings and set `n_epoch=7`, `lr=0.005` and `hidden_size=256`. For argument identification and classification, the settings are `n_epoch=8`. Their respective performance is shown in Table 2.

3.2 Pipeline

After building the modules and training the models, I gather them together and complete the event extraction task in a pipeline. At each step, the input data is the output of the last data(i.e. The pipeline can extract events from raw texts.). And at each step the current performance is recorded. I run the pipeline for 3 times and the average results are in Table 2.

It can be inferred easily from the respective performance of the 4 modules and the gap between modules and pipeline that argument identification is the bottleneck of the pipeline. That's because the features are limited and much syntactic information is missed, compared to (Li et al., 2013). Another challenging part is that there are various types of triggers and argument roles.

4 Conclusion

In this homework I implement 4 modules for event extraction task and package them into a pipeline. And the argument identification seems to be the most challenging part. As the event is extracted in multiple steps, the correlation and cooperation between the steps become important. That's the advantage of joint framework.

References

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, Sofia, Bulgaria. Association for Computational Linguistics.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.