Formal Report

Luyang Huang
14307110020

Ⅰ. Introduction

Event extraction is important in extracting structured information from raw texts. In this task, I adopt Hidden Markov Model (HMM) and Conditional Random Field (CRF) to extract and label events. In addition, I analyze results of the experiments to evaluate two models.

Ⅱ. Model & Algorithm

i. HMM Model

Normal Hidden Markov Model is written in the following form:

$$\mathrm{P}(x_1, \dots, x_n, y_1, \dots, y_n) = \prod_i P(y_i|y_{i-1}, y_{i-2})P(x_i|y_i)$$

We can use transition probability to generate labels for each word in the sequence. In my experiment, specially, I use two input sequences, which respectively denote word and part-of-speech tagging. To modify the probability, I finally get:

$$\mathrm{P}(x_1, \dots, x_n, pos_1, \dots, pos_n, y_1, \dots, y_n) = \prod_i P(y_i|y_{i-1}, y_{i-2})P(x_i|y_i)P(pos_i|pos_i)$$

The toolkit I use for part-of-speech tagging is *pyltp*.

The algorithm I use to implement HMM is:

Algorithm 1 Hidden Markov Model

```
x1, y = input sequence
x2 = Postagger(x1)
count_dict_list = Calculate_frequency(x1, x2,y)
trigram_dict, postag_prob_dict, word_prob_dict = Normalize(count_dict_list)
model.predict(test_file)
```

ii. CRF Model

Another model can be used in this task is called Conditional Random Field model. In this model, we would like to find the most likely underlying state sequence under the model, that is,

$$\arg\max p(s|x, w)$$

Here x denotes features and w denotes weights, both of which are vectors.

This expression is equivalent to:

$$\arg\max \sum_{j=1}^{m} w \cdot \varphi(x, j, s_{j-1}, s_j)$$

We can imagine more features will probably return to better results in this model. Some parameters may be useful in using CRF++, which are feature cutoff, L2 regular terms. Feature cutoff can eliminate some features whose frequency is low in train set. L2 regular terms can prevent from overfitting. Features can be various in CRF. For example, current word, the word before or after current word, two continuous words and the same adopted to pos tag etc.

III. Experiment

i. Toolkit

The toolkit I use for pos tagging is *pyltp.* The toolkit used for CRF is CRF++ 0.54.

ii. Results

Results are shown in the following table:

Dataset	Methods	Features	Type_correct	precision	recall	F-score
trigger	HMM	Words	**0.9685**	0.6949	**0.774**	**0.7323**
trigger	HMM	Words+pos	0.9673	0.6701	0.7537	0.7094
trigger	CRF	Words	0.6023	0.7023	0.4036	0.5126
trigger	CRF	Words+pos	0.8177	**0.8366**	0.5831	0.6872
argument	HMM	Words	0.3676	0.4524	0.7528	0.5652
argument	HMM	Words+pos	0.3708	0.5174	0.7206	0.6023
argument	CRF	Words	0.4026	0.4528	0.4023	0.4261
argument	CRF	Words+pos	**0.7289**	**0.8479**	**0.7847**	**0.8151**

The HMM works better on trigger because the trigger dataset is small and contains less features in CRF compared to argument dataset. (about 100.000 versus about 1,000,000) Surprisingly, using pos tagging doesn't improve the performance. Possibly because the train set is not big enough so that both precision and recall declines.

The Argument train set contains some problems. Some words are separated wrongly, especially when it comes to entity. For example, "塞尔维亚民主反对联盟" should be an entity rather than some separate words. So when it comes to HMM which considers context of labels and the word itself, it doesn't work good enough. Adding pos tagging outperformances simple words in HMM because number of pos tagging is small to train and often doesn't contain some errors compared to word segmentation.

The CRF model works rather good on argument. Because it doesn't simply rely on the context of labels like HMM. Instead, it has more than 1,000,000 features (threshold is 3). In addition, a larger train set helps a lot. We can simply come to a conclusion that in a more complex task, CRF usually works better than HMM since it considers more situations than HMM.

IV. Discussion

In the experiment, I find that the segmentation sometimes prevents the better performance. Thus we need a special term called entity. We should see entity as a single word instead of separate words. This is harder in Chinese I think because in English words have clear boundary. For example, "贝尔格勒" should be a word rather than "贝尔格"

and “勒”. The entity should be added as input in this task so that we can use the context more precisely. It seems HMM is more relied on the quality of the dataset itself and CRF is influenced more by a larger dataset.

The results of HMM can get from running HMM.py and use evaluate function in eval.py to evaluate. The template and results of CRF are provided in the zip and can be evaluated by evaluate_postag in eval.py.