

Chinese event extraction

Yixiong Chen 16307110231

2019.12.12

1.Introduction

In this paper, we will talk about a Chinese event extraction task. The goal of this task is to distinguish whether a word in a Chinese sentence belongs to a set of event words, and more exactly, which category it belongs to. There are two types of words that describe events: a trigger is a word describing how an event happens; an argument is a word representing an entity, temporal expression or value that plays a certain role in the event. Hidden Markov Model (HMM) and Condition Random Field (CRF) model are used to do this task.

2.Methods

HMM.

Hidden Markov Model is a kind of generative model based on Markov Chain (MC). Markov Chain model consists a series of nodes, denoting a series of states, and every node is conditional independent given its previous node. HMM has only a little differences with the simple MC model. That is, every state node determines an evidence node. The evidence node is what we can observe instead of the state node. The aim of this model is to estimate the state nodes according to visible evidence nodes.

In the task of Chinese sentence labeling, the transition probability (how likely a state node is a successor of a particular state node) and emission

probabilities (how likely we observe an evidence node given a particular state node) of the HMM can be easily calculated by counting training samples.

$$P(s|v) = \frac{c(v, s)}{c(v)}, \quad P(x|s) = \frac{c(s, x)}{c(s)}, \quad P(tagA) = \frac{c(tagA)}{c(all\ tags)}$$

Where v, s are two labels, and x is a word, c(v,s) denotes the number of times that label s appears next to v, c(s,x) is used to denote the number of times that x appears as the evidence of state s.

Though holding the HMM model, it is not enough to do the inference job on a Chinese sentence. We have to do forward inference with Viterbi algorithm. Viterbi algorithm is a kind of dynamic programming algorithm, after calculating the probability of the first state, the probability of later states are based on all state probabilities of their predecessors.

The implementation of HMM is based on pure python code.

CRF.

Conditional random fields (CRFs for short) is a kind of discriminant probability model and one of random field models, which is often used to label or analyze sequence data, such as natural language text or biological sequence. It represents the Markov random field of another group of output random variables y given a set of input random variables X, that is to say, CRF is characterized by assuming that the output random variables constitute a Markov random field. The conditional random field can be regarded as the extension of the Maximum Entropy Markov model in the labeling problem.

The application of CRF is based on CRF++ package.

3.Results

HMM.

The HMM achieved accuracy of 95.52% and 70.30% on the trigger classification and argument classification respectively. Detailed results are in the following:

	accuracy	precision	recall	F1
trigger	95.52%	78.52%	65.11%	71.19%
argument	70.3%	69.82%	45.75%	55.28%

CRF.

The CRF achieved accuracy of 95.26% and 73.22% on the trigger classification and argument classification respectively. Detailed results are in the following:

	accuracy	precision	recall	F1
trigger	95.26%	80.12%	58.83%	67.85%
argument	73.22%	69.53%	59.17%	63.93%

4. Analysis

From the result part we can see the HMM performs quite well on the Chinese event extraction task. Is it the whole truth? Actually, only the accuracy index reaches high scores. If we take a look at other indices, we can find that the recall rate on both the trigger data and argument data are pretty low. The poor recall rate is due to how HMM works to find events in a sentence.

The HMM calculates the prior probabilities of different categories by counting the frequency of their occurrence. Although it would be slightly different between training data and testing data but often won't affect the performance of the model. However, the transition probabilities and emission probabilities differ to a large extent between the two datasets. The model calculates transition probabilities also by counting, but unlike prior probabilities, transition form has lots of possibilities but the real occurrence types are few. Once the classifier meets unseen transition forms from the testing data, the probability would be zero and all predictions of words from

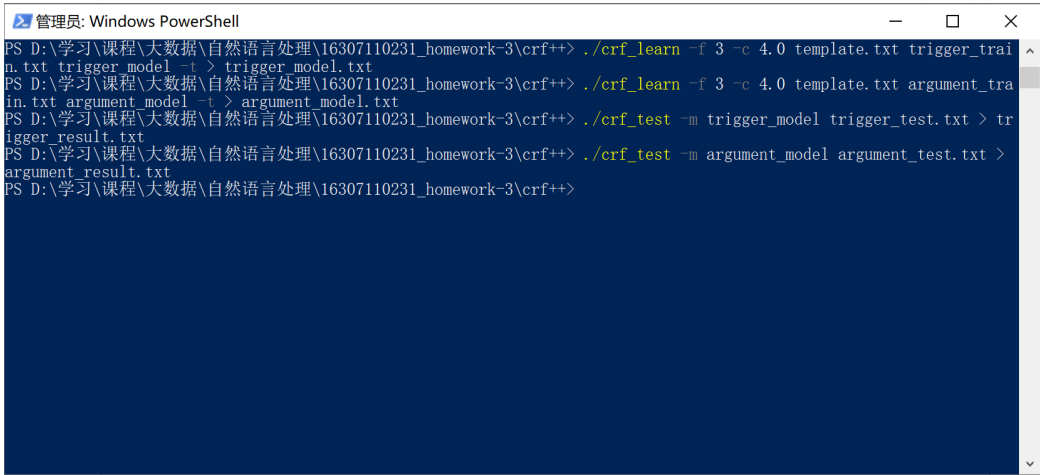
this particular word in the same sentence may make mistakes. The same principle can be adapted to emission probability, once a new word appears the error will follow.

Above are reasons why HMM has such low recall rates. Almost all new words or new transition forms cannot be classified correctly.

The CRF model depends on designed features. Although the simplest features are used in the implementation of this paper, we can still see an obvious improvement of recall rate on argument dataset. If more effort can be put into feature designing, it's not astonishing that CRF can reach above 90% F1 score on large datasets.

5. Appendix

The screen-shots of commands and results for training and testing CRF are in the following:

A screenshot of a Windows PowerShell terminal window titled "管理员: Windows PowerShell". The window has a dark blue background. The terminal shows the following commands and their outputs:

```
PS D:\学习\课程\大数据\自然语言处理\16307110231_homework-3\crf++> ./crf_learn -f 3 -c 4.0 template.txt trigger_train.txt trigger_model -t > trigger_model.txt
PS D:\学习\课程\大数据\自然语言处理\16307110231_homework-3\crf++> ./crf_learn -f 3 -c 4.0 template.txt argument_train.txt argument_model -t > argument_model.txt
PS D:\学习\课程\大数据\自然语言处理\16307110231_homework-3\crf++> ./crf_test -m trigger_model trigger_test.txt > trigger_result.txt
PS D:\学习\课程\大数据\自然语言处理\16307110231_homework-3\crf++> ./crf_test -m argument_model argument_test.txt > argument_result.txt
PS D:\学习\课程\大数据\自然语言处理\16307110231_homework-3\crf++>
```

Figure 1: commands for CRF++

```
====crf++/trigger labeling result====  
type_correct:  0.9231  
accuracy:  0.9526  
precision:  0.8012  
recall:  0.5883  
F1:  0.6785  
====crf++/argument labeling result====  
type_correct:  0.4135  
accuracy:  0.7322  
precision:  0.6953  
recall:  0.5917  
F1:  0.6393
```

Figure 2: results of CRF++