FUDAN UNIVERSITY

NLP Homework 3: Chinese Event Extraction

Deng Wen 17307130171

2019/12/4

1 Introduction

In this homework, I am going to do Chinese Event Extraction.

There are two different ways for me to do this work: one is using Hidden Markov Model to extract the event, the other is to use Conditional Random Fields.

For HMM, I plans to use Supervised learning methods to bulid the models. And for the decoding part, I use greedy decoding and viterbi decoding to respectively.

For CRF, I use the open-source toolkit $CRF++$ to do extraction.

2 Hidden Markov Model

2.1 Model Defination

From the lecture in class, HMM model are defined as following:

- States $X = \{q_1, \cdots, q_M\}$
- Observations $E = \{o_1, \cdots, o_N\}$
- Initial Distribution π_i
- Transitions $P(X_t|X_{t-1})$
- Emissions $P(E_t|X_t)$

2.2 Model Learning Algorithm

Since in the training data files, we are given the observations(Word) and the corresponding states(Event marks). So we are going to use the supervised learning methods to learn the parameters of the HMM models.

Recall the steps to learn the parameters:

2.2.1 *Initial Distribution:*

$$\hat{\pi}_i = \frac{\#\ of\ Sequences\ begin\ with\ q_i}{\#\ of\ Sequences}$$

2.2.2 *Transitions:*

$$P(X_t|X_{t-1}) = \frac{\#\ of\ (X_{t-1}, X_t)'s}{\#\ of\ (X_{t-1}, q_i)'s} \quad i = 1, \cdots, M$$

2.2.3 *Emissions:*

$$P(E_t|X_t) = \frac{\#\ of\ (X_t, E_t)'s}{\#\ of\ (X_t, o_i)'s} \quad i = 1, \cdots, N$$

2.3 HMM decode

In class, we have learned several decoding methods and in this section I am going to implement two decoding ways:

Also to do add-k smoothing, I added a parameter $alpha$ to avoid 0 probability, to make the model more robust. And in later test, I have test a few alhpas to get a relatively good alpha.

2.3.1 Greedy Decode:

This method is greedy because it just start at the left, and use our classifier at each position to assign a label. The classifier can depend on previous labeling decisions as well as observed data. It is Fast with no extra memory requirements and easy to implement. But is is an local soulution which means it is not optimal.

$$\hat{x}_t = \arg\max_x P(x|\hat{x}_{t-1})P(e_t|x)$$

2.3.2 Viterbi Decode:

This method uses dynamic programming and foward and backward algorithm to compute the optimal sequences.

Steps:

1. $M_1(q_i) = \pi_i P(e_1|q_i)$
2. $M_t(q_i) = \max\{M_{t-1}(q_j)P(q_i|q_j)\}P(e_t|q_i)$
3. $Pick \max M_k(q_i)\ and\ backtrack$

2.4 Results

For each task(trigger or argument) with each decoding method(greedy or viterbi), I set alpha equals to $[1e-5, 8e-6, 5e-6, 1e-6, 5e-7, 1e-7, \cdots, 1e-16]$ respectively and choose the one maximize the accuracy.

Here shows the results:

Method	Data	trained alpha	type correct	accuracy	precision	recall	F1
Greedy	Trigger	8e-6	0.9684	0.9578	0.7864	0.6934	0.737
Greedy	Argument	1e-16	0.2871	0.7463	0.6653	0.7399	0.7006
Viterbi	Trigger	8e-6	0.9686	0.9578	0.7833	0.6964	0.7372
Viterbi	Argument	1e-16	0.2981	0.7518	0.6754	0.7344	0.7036

2.5 Analysis

- In this result we can find that the accuracy of Trigger data can achieve about 95.78% and the Argument data can achieve about 75.18%. The reason that Trigger data is higher may be that it has only 8 types(Argument has 35 types), so it is clearer for the Model to classify.
- Viterbi decode are indeed more accurate than greedy decode, but greedy decode is still useful if the two methods' accuracy are similar because greedy decode takes less time!
- Type correct rate is very low in Argument data, while it is much higher in trigger data. It may also due to the number of types, but they both have a relatively good score in recall and F1(about 70%).

3 Conditional Random Field

CRF is a whole-sequence conditional model rather than a chaining of local models. Comparing to HMM, it is more complex and doesn't need the hypothesis of conditional independence and Markov property. It may work more reasonably.

The following is the main principles of the model and steps to use the open-source toolkit CRF++ to do event extraction:

$$p(x_1 \dots x_n | e_1 \dots e_n) = \frac{\exp(W \cdot F(x_1 \dots x_n, e_1 \dots e_n))}{Z_W(e_1 \dots e_n)}$$

where $F(x_1 \dots x_n, e_1 \dots e_n)$ is the features, and W is the weights

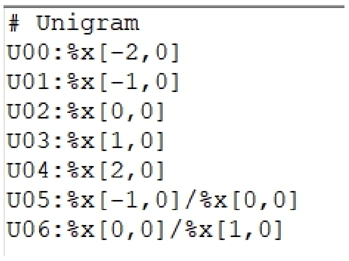
$$F(x_1 \dots x_n, e_1 \dots e_n) = (f_1(x_1 \dots x_n, e_1 \dots e_n), f_2(x_1 \dots x_n, e_1 \dots e_n), \dots, f_K(x_1 \dots x_n, e_1 \dots e_n))$$

$$f_k(x_1 \dots x_n, e_1 \dots e_n) = \sum_{i=1}^{n} f_k(x_{i-1}, x_i, e_1 \dots e_n, i)$$

Using Viterbi decoding to get the optimal label sequence!

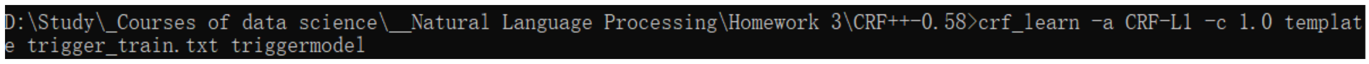
3.1 Feature design(template)

I have tried a few templates can choose this one which works relatively better.

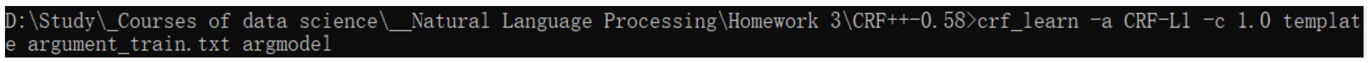


```
# Unigram
U00:%x[-2,0]
U01:%x[-1,0]
U02:%x[0,0]
U03:%x[1,0]
U04:%x[2,0]
U05:%x[-1,0]/%x[0,0]
U06:%x[0,0]/%x[1,0]
```

3.2 Model training



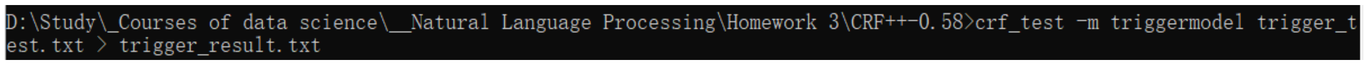
```
D:\Study\_Courses of data science\__Natural Language Processing\Homework 3\CRF++-0.58>crf_learn -a CRF-L1 -c 1.0 templat
e trigger_train.txt triggermodel
```



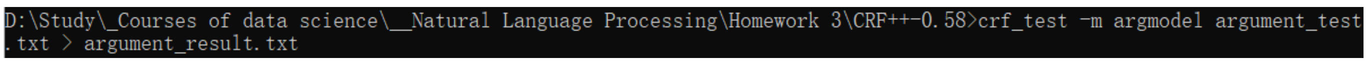
```
D:\Study\_Courses of data science\__Natural Language Processing\Homework 3\CRF++-0.58>crf_learn -a CRF-L1 -c 1.0 templat
e argument_train.txt argmodel
```

After several tests I have found that the when the regularlize parameter is "CRF-L1" and hyper-parameter is "1.0", the accuracy is relatively higher.

3.3 Model test



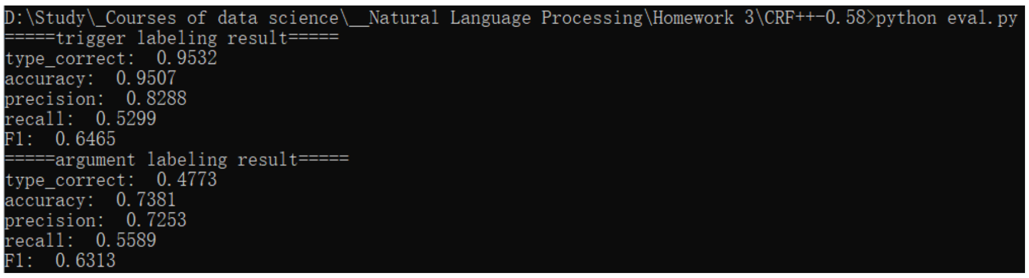
```
D:\Study\_Courses of data science\__Natural Language Processing\Homework 3\CRF++-0.58>crf_test -m triggermodel trigger_t
est.txt > trigger_result.txt
```



```
D:\Study\_Courses of data science\__Natural Language Processing\Homework 3\CRF++-0.58>crf_test -m argmodel argument_test
.txt > argument_result.txt
```

Excute the above code in command line.

3.4 Result and analysis



```
D:\Study\_Courses of data science\__Natural Language Processing\Homework 3\CRF++-0.58>python eval.py
=====trigger labeling result=====
type_correct:  0.9532
accuracy:  0.9507
precision:  0.8288
recall:  0.5299
F1:  0.6465
=====argument labeling result=====
type_correct:  0.4773
accuracy:  0.7381
precision:  0.7253
recall:  0.5589
F1:  0.6313
```

We can see that the CRF model behaves similar or a litter worse than HMM model, because the CRF model are dependent on Feature designs, I may improve it if I can design a better Model template.

4 Conclusion

In this homework, I have got a deep understanding of Hidden Markov Models by implementing it myself, as well as the idea of doing Event extraction. Besides, I also have known the advanced model Conditional Random Field by using the CRF++ toolkit. But this work still has many aspects to improve such as trying more advanced smoothing techniques(HMM), trying trigram(even higer) HMM, designing more comprehensive features(CRF) and so on. This will be done in the future.