NLP HW2 Report

Jiachuan DENG, PUID:0029985722

November 2017

1 Task Description

Named-entity recognition (NER) (also known as entity identification, entity chunking and entity extraction) is a subtask of information extraction that seeks to locate and classify named entities in text into pre-defined categories such as the names of persons, organizations, locations, expressions of times, quantities, monetary values, percentages, etc.
In this task, we are given preprocessed data, which contains both sentence words, and name entity tags. Part of Data is shown in Fig1.

```
sentence (in id): [554  23 241 534 358 136 193  11 208 251 104 502 413 256 104]

sentence (raw): what aircraft is used on delta flight DIGITDIGITDIGITDIGIT from kansas city t
o salt lake city

corresponding tags: O O O O O B-airline_name O B-flight_number O B-fromloc.city_name I-fromlo
c.city_name O B-toloc.city_name I-toloc.city_name I-toloc.city_name
```

Figure 1: Part of Data

2 Algorithm

2.1 BiLSTM+MLP

Firstly, I tried to use MLP to solve this problem. Regarding this problem as a classification problem. The input is: previous tag + current word (both with embedding, and using BiLSTM to extract words' features), the output is: next tag's probability, with Cross-entropy as Loss Function.

Neural Network structure is shown in Fig2.

However, the result is not ideal, the best performance I can get is around 59 in precision.

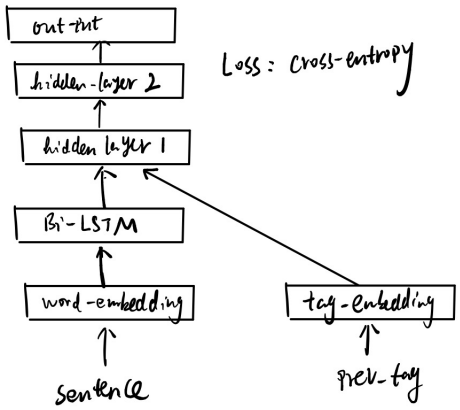

Figure 2: BiLSTM+MLP

2.2 BiLSTM + MEMM(as objective function)

Then I tried to regard this problem not a classification problem, but similarly as a generative model. Which using BiLSTM to extract sentence information, and using MEMM: $p(tag_i|tag_{i-1}, x_1, x_2, ..., x_m) = \frac{exp(p(x,tag_{i-1},tag_i))}{\sum_{tag'} exp(p(x,tag_{i-1},tag'))}$ as objective function. Where $p(x, tag_{i-1}, tag_i)$ can be calculated as $p(tag_i|x)p(tag_i|tag_{i-1})$, and $p(tag_i|x)$ can be presented as features extracted from BiLSTM, and we can construct a extra matrix **tag_trans** to represent $p(tag_i|tag_{i-1})$. Viterbi is used to extract tag sequences. Structure can be found in Fig3.

3 Feature Extraction

Word Embedding + BiLSTM. As is shown in Fig3, I add a word embedding layer, and then feed word embedding into a BiLSTM, and using the output of BiLSTM as features.

4 Parameters

I tried different parameters on Embedding Dim, Hidden Layer Dim, and number of Epoch, and get following results, as shown in Fig4.

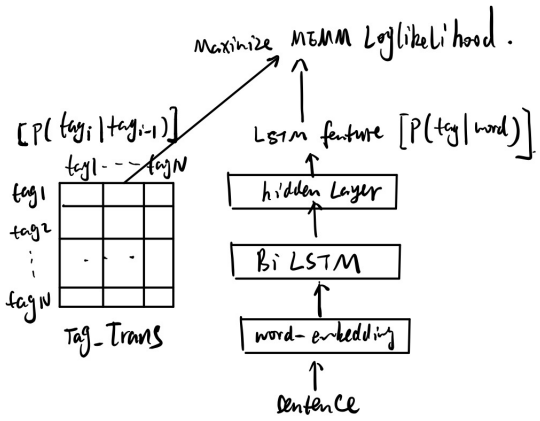

Figure 3: BiLSTM+MEMM(Objective Function)

I finally choose Embedding Dim = 15, Hidden Dim=10, and run it on 16 epochs.

```
EMBEDDING_DIM = 5
HIDDEN_DIM = 4
77.97 71.98 74.85

EMBEDDING_DIM = 15 (V3 epoch =8)
HIDDEN_DIM = 10
with Normal
42.74 34.65 38.27

EMBEDDING_DIM = 15 (V2 epoch=8)
HIDDEN_DIM = 10
without Normal
85.52 82.02 83.74

EMBEDDING_DIM = 15 (V4 epoch=8+8)
HIDDEN_DIM = 10
89.01 86.46 87.72

EMBEDDING_DIM = 20 (V5 epoch=8)
HIDDEN_DIM = 10
84.99 82.02 83.48

EMBEDDING_DIM = 20 (V6 epoch=8+4)
HIDDEN_DIM = 10
86.57 83.61 85.06


EMBEDDING_DIM = 30 (V7 epoch=8)
HIDDEN_DIM = 10
85.21 82.66 83.91
```

Figure 4: Parameters & Results

5 Experimental Results

The metrics are:

1. **Precison**= $\frac{TruePositive}{TruePositive+FalsePositive}$
2. **Recall**= $\frac{TruePositive}{TruePositive+FalseNegative}$
3. **F1**= $\frac{2xPrecisonxRecall}{Precision+Recall}$

Experiment is run on 100% of given training set, and test on 100% of given test set.

6 Conclusion

88.1,85.09, 86.57 are achieved in precision, recall and F1 Score respectively.