

# 预训练模型学习情况周报 7

姚凯

## 一、 本周学习：

### 项目实践

完成任务二：命名实体识别任务

**数据集：**CLUENER2020

**方案：**基于预训练模型 RoBERTa，改进自 BERT

#### 1) 数据预处理

将原始标签转换为 BIOS 标注：

```
{"text": "彭小军认为，国内银行现在走的是台湾的发卡模式，先通过跑马圈地再  
在圈的地里面选择客户，", "label": {"address": {"台湾": [[15, 16]]}, "name": {"彭小  
军": [[0, 2]]}}
```

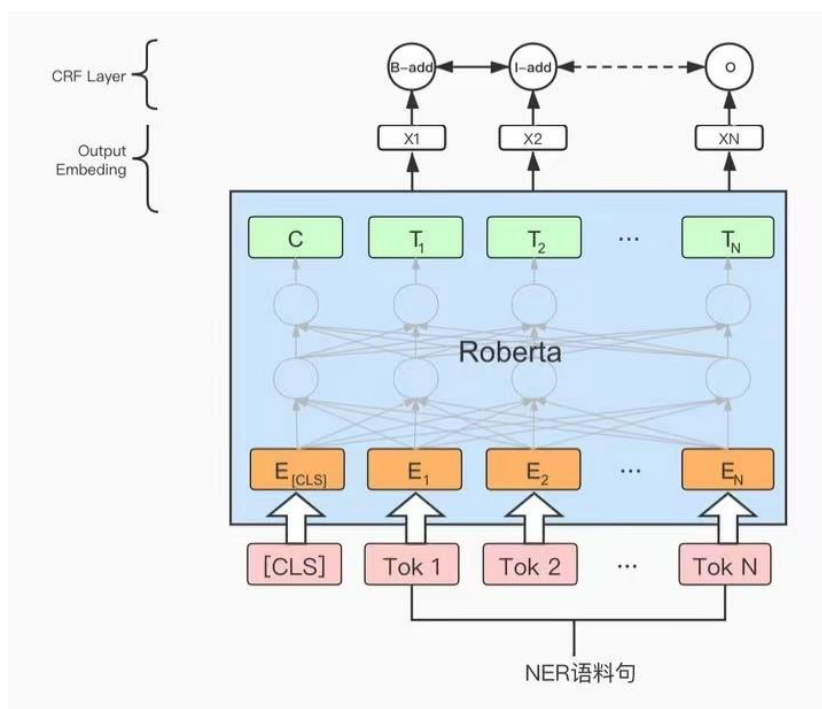
```
['B-name', 'I-name', 'I-name', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'B-  
address', 'I-address', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O',  
'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O']
```

#### 2) 构建数据集

第一步得到文本和标签序列，分别通过自建词表转为索引序列，并将每个 batch 中的句子和标签补全到同一长度。数据集按 9: 1 序列分割为训练集和验证集

#### 3) 模型结构

使用的预训练模型为中文 RoBERTa-wwm-ext-large，解决序列元素的多分类问题，条件随机场 CRF 则精细标签之间的依赖关系



整个模型将处理好的文本序列（索引格式）输入 Roberta 中，得到预测标签，再输入到 CRF 中修改预测标签，经过词表映射，与真实标签计算评测分数，来微调模型 weights

## 训练结果

```
root@281f96d0eb1f:/CLUEN X train.log X
```

```
-----Save best model!-----  
100% ██████████ 303/303 [02:31<00:00, 2.00it/s]  
Epoch: 42, train loss: 0.508043660582489  
Epoch: 42, dev loss: 691.8631125057445, f1 score: 0.7984496124031009  
100% ██████████ 303/303 [02:30<00:00, 2.01it/s]  
Epoch: 43, train loss: 0.3310048320505879  
Epoch: 43, dev loss: 685.5977944766773, f1 score: 0.7973110613159503  
100% ██████████ 303/303 [02:30<00:00, 2.01it/s]  
Epoch: 44, train loss: 0.3919056933311739  
Epoch: 44, dev loss: 684.6573818431181, f1 score: 0.7967380224260957  
100% ██████████ 303/303 [02:31<00:00, 2.01it/s]  
Epoch: 45, train loss: 0.4729228507567554  
Epoch: 45, dev loss: 677.6162351720474, f1 score: 0.7974709361615337  
100% ██████████ 303/303 [02:31<00:00, 2.00it/s]  
Epoch: 46, train loss: 0.40824869993102825  
Epoch: 46, dev loss: 679.368007884306, f1 score: 0.7984496124031009  
100% ██████████ 303/303 [02:31<00:00, 2.00it/s]  
Epoch: 47, train loss: 0.2209453016224474  
Epoch: 47, dev loss: 677.8815621768726, f1 score: 0.7973856209150326  
100% ██████████ 303/303 [02:31<00:00, 2.01it/s]  
Epoch: 48, train loss: 0.3171855561410633  
Epoch: 48, dev loss: 681.50926657284, f1 score: 0.7978766843609636  
100% ██████████ 303/303 [02:31<00:00, 2.00it/s]  
Epoch: 49, train loss: 0.3545302967033764  
Epoch: 49, dev loss: 680.9198150634766, f1 score: 0.7986111111111112  
100% ██████████ 303/303 [02:31<00:00, 2.00it/s]  
Epoch: 50, train loss: 0.33892178928891425  
Epoch: 50, dev loss: 681.285052131204, f1 score: 0.7986111111111112  
Best val f1: 0.8010631772643632  
Training Finished!
```

```
Saving completed root@281f96d0eb1f:/CLUENER2020-main/BERT
```

参考：

[https://blog.csdn.net/weixin\\_44388679/article/details/106871992](https://blog.csdn.net/weixin_44388679/article/details/106871992)

<https://zhuanlan.zhihu.com/p/346828049>

<https://github.com/hemingkx/CLUENER2020>

调试 bug：

If reserved memory is >> allocated memory try setting max\_split\_size\_mb to avoid fragmentation.

解决：增大内存，或者逐步减半 batch\_size

## 论文阅读

继续看完了《Pre-train, Prompt, and Predict: A Systematic Survey of Prompting Methods in Natural Language Process》这篇论文

### 对比

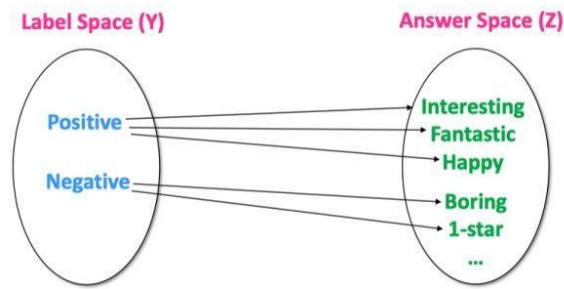
预训练+微调 (pre-train+Fine tune)：一个具有固定架构的模型通过预训练作为语言模型，当应用到下游任务时，引入额外参数，使用针对具体任务的目标函数微调模型参数。需要精心设计预训练和微调阶段使用的训练目标。

预训练+提示+预测 (pre-train,prompt,predict) :选择适当的 prompt 方法用预训练后的语言模型本身即可完成所需的预测，甚至无需额外的特定任务训练。

### 提示学习步骤

- 设计一个 prompt 映射函数（即模板 template），把原始文本  $x$  转变为基于 prompt 的模型需要的格式  $x'$ 。文本  $x$  "I love this movie."，其转变之后的文本  $x'$  为 "[ $x$ ]. Overall it was a [ $Z$ ] movie."。 [ $Z$ ]代表需要预测的答案文本  $z$ 。
- 输入  $x'$ ，在合适的语言模型上寻得具有最大概率的 [ $z$ ]
- 构造 Prompt 答案空间映射 (Verbalizer)，对应预测的  $Z$  与实际需要的标签

y。并通过映射得到所需的预测标签 y。



### Prompt 设计:

- 1) 手工设计，基于人类实验、经验和语言专业知识
- 2) 自动学习模板：
  - A. 离散 prompts: 自动生成由自然语言的词组成的 prompt，其搜索空间是离散的。
  - B. 连续 Prompts: 生成的 prompt 只用机器理解，直接作用在模型的向量空间

### Answer 设计:

根据任务选择答案[Z]的形状（字符/一句话/一篇文章），在答案空间搜索以及到目标 Y 的映射也有手工设计和自动设计两种。

参考:

[https://blog.csdn.net/qq\\_42393368/article/details/121229804](https://blog.csdn.net/qq_42393368/article/details/121229804)