本周报告

> 2020/05/07 1652792 罗吉皓

本周主要做了机器学习模型的调整以及相关项目的搭建，分为以下几个部分：

1. 对比实验
2. 前后端展示平台搭建
3. 论文准备

对比实验

在对比实验中主要针对CNN_LSTM模型，BiLSTM模型以及BiLSTM_CRF三个模型进行对比实验。对比实验中 数据预处理及特征化的提取都是使用的Bert，选取的CNN，BiLSTM以及BiLSTM_CRF的模型都是模仿目前在Ner领域相对标注效果较好的网络结构进行搭建。目前跑下来的结果如下所示：

1. CNNLSTM网络

在CNNLSTM网络中，经过Bert处理后，我使用的是简单的卷积层累加，而后接LSTM网络，加入Dropout防止过拟合。由于是序列的学习问题，在最后我都加上了time_distributed层，其结构大致如下：

layer_conv (Conv1D)	(None, 400, 32)	294944	non_masking_layer[0][0]
layer_lstm (LSTM)	(None, 400, 128)	82432	layer_conv[0][0]
layer_dropout (Dropout)	(None, 400, 128)	0	layer_lstm[0][0]
layer_time_distributed (TimeDis	(None, 400, 7)	903	layer_dropout[0][0]
activation (Activation)	(None, 400, 7)	0	layer_time_distributed[0][0]

其训练参数如下图所示：

```
==================================================================================================
Total params: 101,969,319
Trainable params: 378,279
Non-trainable params: 101,591,040
__________________________________________________________________________________________________
```

最后跑出来的结果如下所示，一般来说Places的准确率相对会更高一点，而Target Places是从Places里面进行挑选准确率肯定相对较低。

	precision	recall	f1-score	support
V	0.5444	0.5104	0.5269	96
P	0.6583	0.7401	0.6968	177
T	0.5571	0.5342	0.5455	73
micro avg	0.6100	0.6329	0.6213	346
macro avg	0.6054	0.6329	0.6177	346

2. BiLSTM网络

BiLSTM的引入帮助模型进一步理解上下文语境，使从后往前的信息编码成为可能，比较适合像提取核心地点这种更为细粒度的分类需求。网络结构图以及训练参数如下所示：

```
_________________________________________________________________________________________________
layer_blstm (Bidirectional)     (None, 400, 256)     3277824     non_masking_layer[0][0]
_________________________________________________________________________________________________
layer_dropout (Dropout)         (None, 400, 256)     0           layer_blstm[0][0]
_________________________________________________________________________________________________
layer_time_distributed (TimeDis (None, 400, 7)       1799        layer_dropout[0][0]
_________________________________________________________________________________________________
activation (Activation)         (None, 400, 7)       0           layer_time_distributed[0][0]
_________________________________________________________________________________________________
==================================================================================================
Total params: 104,870,663
Trainable params: 3,279,623
Non-trainable params: 101,591,040
__________________________________________________________________________________________________
```

可以看出整体的参数和训练参数都比原来多很多，相对出来的效果（以核心地点T为例子）也在原来的基础上提高了将近5%的准确率。

	precision	recall	f1-score	support
T	0.5976	0.6712	0.6323	73
P	0.7514	0.7684	0.7598	177
V	0.5773	0.5833	0.5803	96
micro avg	0.6694	0.6965	0.6827	346
macro avg	0.6706	0.6965	0.6831	346

3. BiLSTMCRF模型

在BiLSTM模型的基础上，我添加了CRF条件岁机场，biLSTM只能够预测文本序列与标签的关系，而不能预测标签与标签之间的关系，而通过CRF中的转移矩阵可以帮助了解标签之间的相互关系，即核心动词与核心地理位置之间的关系。相关示意图如下：

layer_blstm (Bidirectional)	(None, 400, 256)	3277824	non_masking_layer[0][0]
layer_dense (Dense)	(None, 400, 64)	16448	layer_blstm[0][0]
layer_crf_dense (Dense)	(None, 400, 7)	455	layer_dense[0][0]
layer_crf (CRF)	(None, 400, 7)	49	layer_crf_dense[0][0]

```
==================================================================================================
Total params: 104,885,816
Trainable params: 3,294,776
Non-trainable params: 101,591,040
```

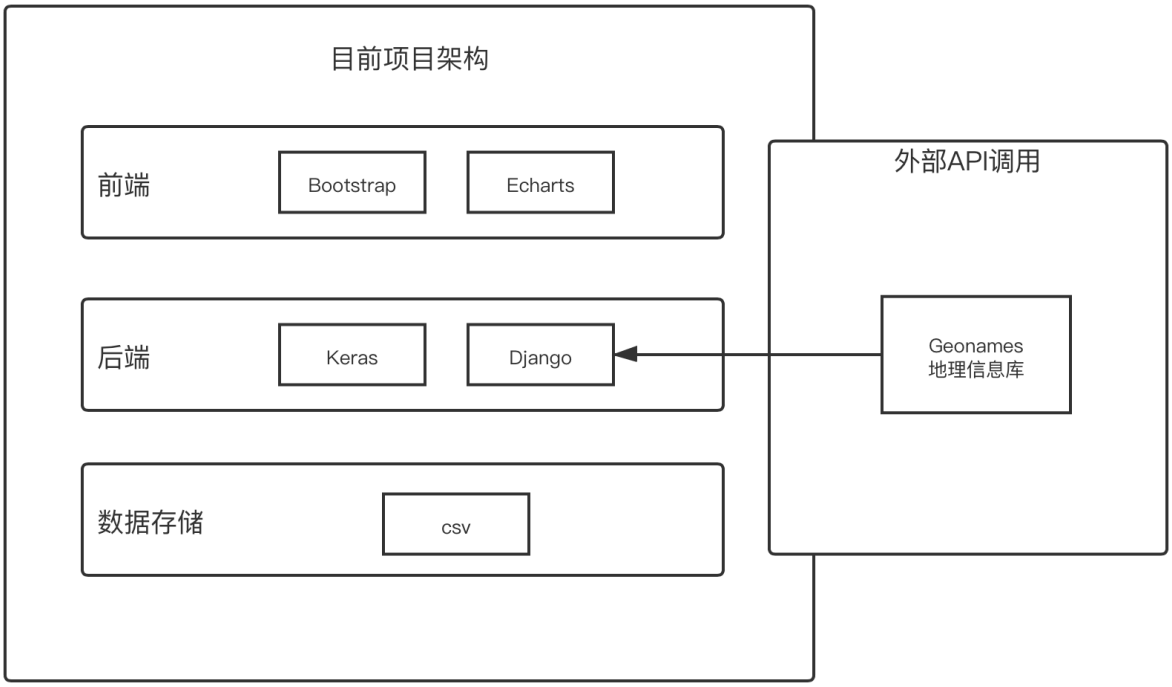
	precision	recall	f1-score	support
T	0.6563	0.6734	0.6647	73
P	0.7384	0.7175	0.7278	177
V	0.5859	0.6042	0.5949	96

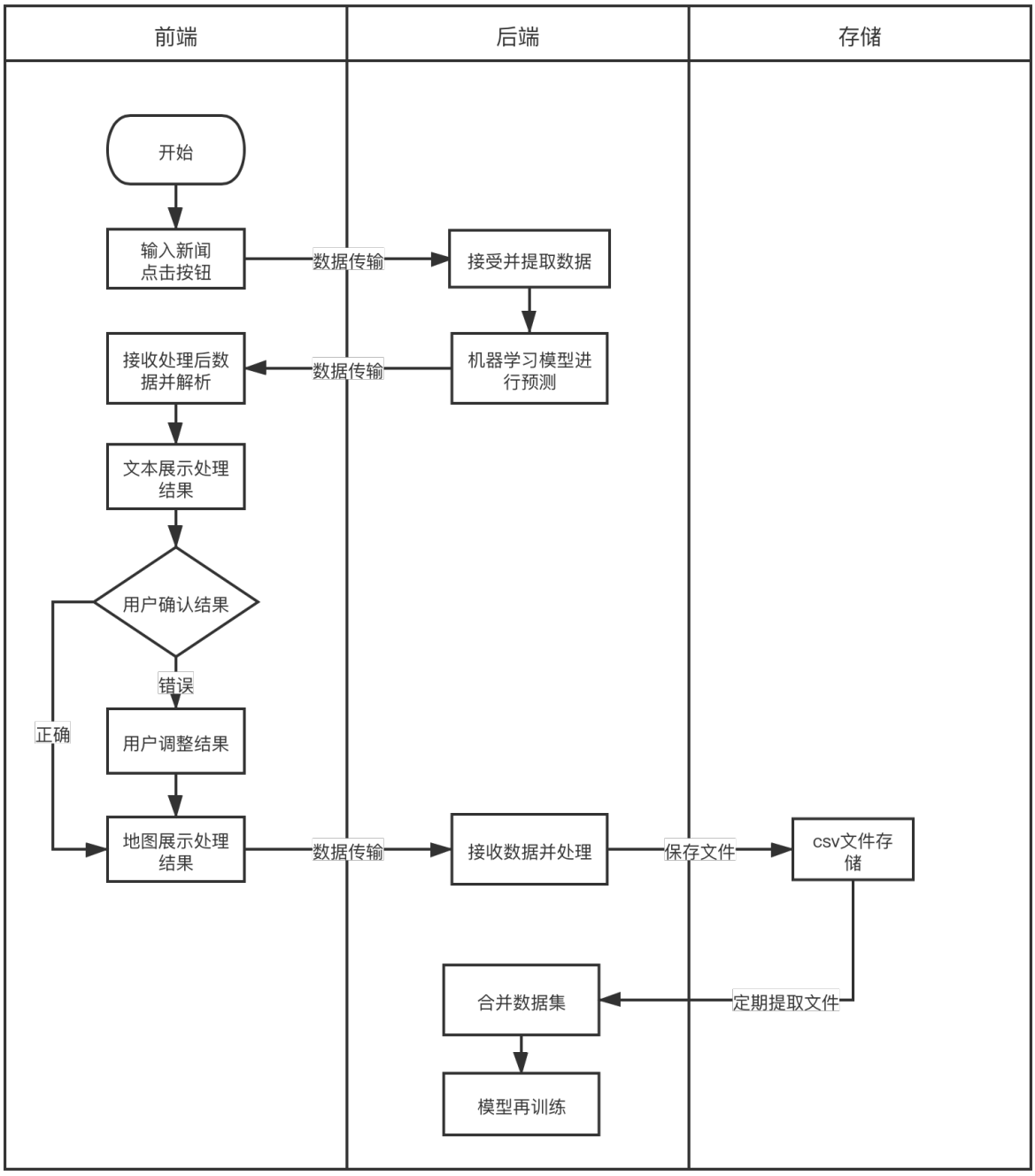
可以发现加入CRF层后核心动词与核心地理位置的准确率都有所提升。其中核心地理位置在BiLSTM的基础上又提高了5%左右，证明三种模型中确实BiLSTMCRF的效果是最好的。

下次改进

原本还想做Bert RandomInt和ERNIE 三种文本预处理及特征提取方法的对比实验，可以上三种模型对比已经花了10天多，后续可能按照时间安排适当调整这一块的对比实验的进行。

展示标注平台的搭建


目前项目架构
前端
Bootstrap
Echarts
后端
Keras
Django
数据存储
csv
外部API调用
Geonames
地理信息库


前端
后端
存储
开始
输入新闻
点击按钮
数据传输
接受并提取数据
接收处理后数据并解析
数据传输
机器学习模型进行预测
文本展示处理结果
用户确认结果
错误
正确
用户调整结果
地图展示处理结果
数据传输
接收数据并处理
保存文件
csv文件存储
合并数据集
定期提取文件
模型再训练

目前前端的效果图如下：

请输入新闻 确认

“initial”: [[“P” 贝,“P” 尔,“P” 格,“P” 莱,“P” 德,“O” 电,“O” 台,“O” 报,“O” 道,“O” ，,“O” 多,“O” 达,“O” 1,“O” 0,“O” 0,“O” 名,,“P” 阿,“P” 尔,“P” 巴,“P” 尼,“P” 亚,“O” 族,“O” 人,“O” 在,“T” 科,“T” 索,“T” 沃,“V” 举,“V” 行,“O” 示,“O” 威,]],

“targetPlace”: " 科索沃",“places”: " 贝尔格莱德 阿尔巴尼亚",“verbs”: " 举行"

相关接口规范

API1
http://127.0.0.1:8000/Geolocating/
请求类型：POST
说明：获取新闻中的核心地理位置

请求参数

参数名	参数类型	描述
sen	string	示例：贝尔格莱德电台报道，多达 100 名阿尔巴尼亚族人在科索沃举行示威。

响应参数

参数名	参数类型	描述
initial	list	
targetPlace	list	科索沃
places	list	贝尔格莱德 阿尔巴尼亚
verbs	list	举行

DEMO

```
输入示例
{
“sen”:“贝尔格莱德电台报道，多达 100 名阿尔巴尼亚族人在科索沃举行示威。”
}

输出示例
{
“initial”: [
[
“P”,
“P”,
“P”,
“P”,
“P”,
“O”,
“O”,
“O”,
“O”,
“O”,
“O”,
“O”,
“O”,
“O”,
“O”,
“O”,
“O”,
“O”,
“P”,
```

```
“P”,
“P”,
“P”,
“P”,
“O”,
“O”,
“O”,
“T”,
“T”,
“T”,
“V”,
“V”,
“O”,
“O”,
“O”
]
],
“targetPlace”: " 科索沃",
“places”: " 贝尔格莱德 阿尔巴尼亚",
“verbs”: " 举行"
}
```

API2
http://127.0.0.1:8000/CalculateLocating/
请求类型：POST
说明：地理位置经纬度

请求参数

参数名	参数类型	描述
location	string	示例：科索沃

响应参数

参数名	参数类型	描述
latitude	string	
longitude	string	

DEMO

```
输入示例：
{
    "location":"科索沃"
}
输出示例：
{
    "latitude": "42.58333",
    "longitude": "20.91667"
}
```

API3
http://127.0.0.1:8000/StoreResult
请求类型：POST
说明：标注内容保存

请求参数

参数名	参数类型	描述
result	list	
sen	string	

响应参数

参数名	参数类型	描述
statuscode	int	1

DEMO

```
输入示例
{
“sen”:“贝尔格莱德电台报道，多达 100 名阿尔巴尼亚族人在科索沃举行示威。”,
“result”: [
[
“P”,
“P”,
“P”,
“P”,
“P”,
“O”,
“O”,
“O”,
“O”,
“O”,
“O”,
“O”,
“O”,
“O”,
“O”,
“O”,
“O”,
“P”,
“P”,
“P”,
“P”,
“P”,
“O”,
“O”,
“O”,
“T”,
“T”,
“T”,
“V”,
“V”,
“O”,
“O”,
“O”
]
]
}

输出示例：
{
“statuscode”: 1
}
```

相关具体实现说明：

1. 后端预测

理论上来说，机器学习的效果应该是好于动词地名树的效果。机器学习的数据集是基于动词地名树生成的，相对来说对于地理位置的判断应该更为准确。另外一方面，动词地名树受限于规则，规则的不完善导致其预测结果并不准确，而模型的预测基于机器学习，跳出了规则的限制相对来说更为准确，但是在实际预测的过程中，可能是由于数据集的不完善以及训练不充分，可能会出现如下的情况：

```
错误预测：
“targetPlace”: " 科索沃",
“places”: " 尔格莱德 阿尔巴尼亚",
“verbs”: " 举行"

正确预测：
“targetPlace”: " 科索沃",
“places”: " 贝尔格莱德 阿尔巴尼亚",
“verbs”: " 举行"
```

在这种情况中，地理名词并没有被完全的提取出来，因此在后端预测方面我采用了动词地名树与机器学习相结合的方法，以机器学习为主，动词地名树作为纠正工具来规避这样的错误，并将正确的数据放入csv中，方便下次训练。

2. 地理定位

原本地理定位是选择使用百度地图api，后来发现百度地图只能定位国内的所有地点，国外的地点无法获取经纬度。因此转战谷歌地图，结果谷歌地图出现ssl请求失败情况，可能是由于python本身自带的openssl库没有办法满足谷歌服务本身的安全性下限。另外一方面网络不稳定，谷歌服务还是需要翻墙使用的，我写到一半服务器挂了，最后还是放弃了GoogleMap的服务。

最后选择的是Geoname的API来获取信息，这个是参考的当初modecai的整体项目实现。目前使用的是Geoname的在线API，未来也可以尝试使用docker+elasticSearch服务启动的方式，将整个地理相关库保存在本地方便我们使用。比较巧的是Geoname也是通过postcode来对于地理名词进行统一化处理，也正好和之前的编码工作进行合并。

3. 模型再训练

目前设定的模型再训练周期为一个月，自动化脚本将原有数据集与新建的数据集进行合并帮助模型训练。在训练完成后，新模型会自动覆盖原有模型，提高整体解决方案的性能。