Web 实验报告 Lab2

金哲欣 PB17111663

许世晨 PB17030846

0. 分工

姓名	分工
金哲欣	纯 CRF 模型、规则、融合
许世晨	Bert + BiLSTM + CRF

1. Bert + BiLSTM + CRF

1.1 数据预处理

把非实体的字标记为 `0`，实体的首字以 `B-` 开头，后面跟 `ILL`、`MDC`、`OPR`、`DSC`、`PHT`、`LAB`，分别对应疾病和诊断、药物、手术、解剖部位、影像检查、实验室检验，实体的其余部分使用 `I-` 开头，后面跟实体标签。

数据示意：

```
门 O
诊 O
以 O
“ O
直 B-ILL
肠 I-ILL
癌 I-ILL
术 I-ILL
后 I-ILL
” O
收 O
入 O
院 O
。 O
```

1.2 Bert

Bert 是词向量的预处理模型，其中提供了中文的预处理词向量，也支持 fine-tune 进一步进行训练。

使用 `bert-as-server` 在本机上部署 bert 服务器，然后在程序中使用 bert 的客户端获取文本（包括训练集文本和测试集文本）中出现的所有字的词向量，将其作为词向量 embedding 的初始设置。

在训练过程中，可以选择更新 embedding 或者不更新，最终经过测试，更新 embedding 的效果更好。

我们也尝试使用 bert 的 fine-tune 对 bert 模型进行训练，但是由于训练需要大量的资源和时间（光是 3 个 epoch 就训练了 2 个小时），我们仅仅尝试了一次，效果不错。

1.3 BiLSTM

LSTM的全称是Long Short-Term Memory，它是RNN（Recurrent Neural Network）的一种。LSTM由于其设计的特点，非常适合用于对时序数据的建模，如文本数据。BiLSTM是Bi-directional Long Short-Term Memory的缩写，是由前向LSTM与后向LSTM组合而成。两者在自然语言处理任务中都常被用来建模上下文信息。

这里选择使用biLSTM来处理Bert产生的向量序列。使用tensorflow构建双向LSTM，输入Bert产生的向量，输出被biLSTM提取出的向量序列。这个向量序列将被送往crf进行下一步处理。

构建biLSTM的核心代码：

```
cell_fw = LSTMCell(self.hidden_dim)
            cell_bw = LSTMCell(self.hidden_dim)
            (output_fw_seq, output_bw_seq), _ =
tf.nn.bidirectional_dynamic_rnn(
                cell_fw=cell_fw,
                cell_bw=cell_bw,
                inputs=self.word_embeddings,
                sequence_length=self.sequence_lengths,
                dtype=tf.float32)
            output = tf.concat([output_fw_seq, output_bw_seq], axis=-1)
            output = tf.nn.dropout(output, self.dropout_pl)
```

1.4 CRF

条件随机场(CRF)由Lafferty等人于2001年提出，结合了最大熵模型和隐马尔可夫模型的特点，是一种无向图模型，近年来在分词、词性标注和命名实体识别等序列标注任务中取得了很好的效果。

BiLSTM 提取了原序列的特征后，需进行进一步处理。这里使用 sklearn 包中的 crf 作为这个模型的最后一层，输出结果。

2. 纯 CRF 模型

我们选择使用 `sklearn-crfsuite` 工具包来构建纯粹的 CRF 模型。

由于训练集比较少，这种 CRF 模型远比 Bert + biLSTM + CRF 的神经网络模型效果好。

2.1 特征工程

我们设计了以下特征：

- 当前字本身（如果是英文则转换为小写）。
- 前一个字本身（如果是英文则转换为小写）。
- 后一个字本身（如果是英文则转换为小写）。
- 前一个字与当前字的拼接。
- 当前字与后一个字的拼接。
- 如果当前字是文本第一个字，标记为 `BOS` 。
- 如果当前字的文本最后一个字，标记为 `EOS` 。
- 当前字、前一个字、后一个字分别是否为数字。
- 当前字、前一个字、后一个字分别是否为标点。
- 当前字、前一个字、后一个字分别是否为英文字母。

代码示意：

```
def word2features(sent, i):
    punctuations = [' ', '+', ', ', '-', ': ', '、', '.', '; ', '。', '? ',
        '/', '*', '\\', '(', ')', ' (', ') ', '〞', '“', '"']
    english = 'qwertyuiopasdfghjklzxcvbnm'

    word = sent[i][0]
    features = {
        'bias': 1.0,
        'word.lower()': word.lower(),
        'word.isdigit()': word.isdigit(),
        'word.ispunc()': word in punctuations,
        'word.isenglish()': word.lower() in english
    }

    if i > 0:
        word1 = sent[i-1][0]
        features.update({
            '-1:word.lower()': word1.lower(),
            '-1:word.isdigit()': word1.isdigit(),
            '-1:word.ispunc()': word1 in punctuations,
            '-1:word.isenglish()': word1.lower() in english,
            '-1+0:words.lower()': word1.lower() + word.lower()
        })
    else:
        features['BOS'] = True

    if i < len(sent)-1:
        word1 = sent[i+1][0]
        features.update({
            '+1:word.lower()': word1.lower(),
            '+1:word.isdigit()': word1.isdigit(),
            '+1:word.ispunc()': word1 in punctuations,
```

```
            '+1:word.isenglish()': word1.lower() in english,
            '0+1:words.lower()': word.lower() + word1.lower()
        })
    else:
        features['EOS'] = True

    return features
```

2.2 参数设置

代码示意及注释：

```
crf = sklearn_crfsuite.CRF(
        algorithm='lbfgs',  # 使用 L-BFGS 梯度下降
        c1=0.2,   # L1 正则化参数
        c2=0.1,   # L2 正则化参数
        max_iterations=600,   # 迭代次数
        all_possible_transitions=True,    # 所有转移（降低速度，提升精度）
        all_possible_states=True,   # 所有状态（降低速度，提升精度）
        verbose=True    # 显示训练过程
    )
```

3. 规则

使用正则表达式来定义规则，基于规则的模型主要追求精度，用于最后的模型融合。

3.1 自动规则

部分规则：

```
(?P<疾病和诊断>脑梗死后遗症)
(?P<影像检查>冠脉CTA)
(?P<疾病和诊断>酒精性脂肪肝)
(?P<疾病和诊断>（升结肠）粘液性腺癌)
(?P<疾病和诊断>食管癌（T3N2M0IIIB期）)
(?P<手术>宫颈癌根治术\(II型子宫切除\+两侧附件切除\+盆腔淋巴结清扫术\+腹主动脉旁淋巴结活检术）)
(?P<实验室检验>Lac)
(?P<疾病和诊断>骶1-3水平椎管局限性膨大)
(?P<实验室检验>C反应蛋白)
(?P<影像检查>脑、肺CT)
(?P<疾病和诊断>慢性肾脏病5期)
(?P<疾病和诊断>肠隆起型绒毛状-管状腺癌II级)
(?P<药物>奈达铂)
(?P<手术>肠粘连松解术\+空肠造瘘术)
...
```

方法：

1. 根据训练集已有的实体名称，生成一个词表。
2. 对词表中的每个词，生成正则表达式。
3. 在训练集中进行验证每一条正则表达式，如果精度大于阈值（比如 0.9），则保留该规则，否则删去该规则。

3.2 手动规则

手动设置规则：

```
“(?P<疾病和诊断>.{1,10}癌)”
“(?P<疾病和诊断>.{1,10}术后)”
护(?P<解剖部位>肝)
(?P<解剖部位>腹)痛
(?P<解剖部位>腹)胀
(?P<解剖部位>腹腔)内
(?P<解剖部位>腹)水
(?P<解剖部位>肠壁.{1,30}淋巴结)癌转移
(?P<解剖部位>(小|中|大)弯.?淋巴结)
(?P<解剖部位>幽门(上|下)淋巴结)
行“(?P<手术>.{1,20}术)”
保(?P<解剖部位>肝)
(?P<解剖部位>肝)组织
(?P<解剖部位>肝)被膜
(?P<解剖部位>胸)闷
保(?P<解剖部位>胃)
。(?P<疾病和诊断>.{1,10}炎)。
```

主要用于弥补自动规则中泛化性不足，目测一些显而易见的规则，同时也经过在训练集中精确度的验证。

4. 融合与纠正

将规则模型、纯 CRF 模型进行融合：

- 如果CRF模型中有实体名与规则模型的实体名有部分重叠，则以规则模型的实体名为准，删去其他的重叠项。（因为规则模型精度高）
- 对于每个实体名，有如下要求，否则删去：
  - 中英文括号必须成对出现。
  - 中英文引号必须成对出现。
  - 实体名的第一个和最后一个字符不能是标点（可以是括号、引号之类）。
- 去除重复项。

5. 结果

截至 2019.12.17 14:08 ，我们的排名为第2名：

Filename	Precision	Recall	F1
lwx_sub5.csv	0.7613694957203259	0.7160314227524004	0.7380047980807677
PB17111663_lab2_submission_10.csv	0.754573331923443	0.6920764232373193	0.7219749089437475
PB17111586_submission11.csv	0.7493973378052615	0.6934341964891864	0.7203304452951845
PB17111586_submission12.csv	0.7302618428942634	0.708660653670837	0.719299109120441

6. 代码运行

6.0 环境

Ubuntu 18.04

Python 3.6.9

6.1 安装相关包

```
$ pip3 install -r requirements --user
```

6.2 数据预处理

请把训练集、测试集分别改名为 `train.txt` 和 `test.txt` ，放入 `Data` 目录下。

运行：

```
$ python3 prepocess.py
```

会生成 `Data/test.json` 、 `Data/train_word.txt` 和 `Data/test_word.txt` 。

6.3 规则模型

手动规则见 `Data/manual-rules.txt` 。

自动生成规则，运行：

```
$ python3 rule.py
```

会生成 `Data/auto-rules.txt` 。

同时生成基于规则的结果 `Data/rule.csv` 。

6.4 纯 CRF 模型

直接运行：

```
$ python3 crf.py
```

会生成基于 CRF 的结果 `Data/submit.csv` 。

6.5 模型融合

运行：

```
$ python3 fuse.py
```

得到最终结果 `Data/fuse.csv` ，提交即为 0.62197 。

PB17111663_lab2_submission_10.csv	0.754573331923443	0.6920764232373193	0.7219749089437475
PB17111663_lab2.csv	0.754573331923443	0.6920764232373193	0.7219749089437475

结果应该完全相同。