

命名实体识别项目说明

课程名称: 知识图谱

学号: 201828018670073

姓名: 刘浩林

2019 年 5 月 27 日

目录

一、	NER 数据集的构建.....	3
二、	NER 方法.....	5
三、	NER 项目运行流程	6
四、	NER 项目测试性能指标	7
五、	附录：代码结构图	10

一、NER 数据集的构建

本项目实现的功能是对中文命名实体进行识别,采用神经网络的方法进行识别(具体方法见 NER 识别方法),因此需要构建基于中文语料库的命名实体识别数据集,本项目通过爬取网页的新闻数据和已有的人民日报和 MSRA 中文语料库的基础上构建自己的命名实体识别数据集,常用的命名实体识别的标注方式有 BIO 和 BME0 格式的标注方法,两种标注方法分别如图 1 和图 2 所示:

中	B-LOC
国	I-LOC
很	O
大	O
句	O
子	O
结	O
束	O
是	O
空	O
行	O

图 1. BIO 标注格式

美	B-LOC
国	E-LOC
的	O
华	B-PER
莱	I-PER
士	E-PER
我	O
跟	O
他	O
谈	O
笑	O
风	O
生	O

图 2. BME0 标注格式

考虑到构建的数据集较大,且 BME0 标注更为精细,所以项目采用 BME0 格式对采集的数据进行标注。

使用 python 中的 pandas 进行爬取搜狐和新华网的新闻数据,网页设置如下图所示:(详见目录 NewCrawler\utils\news_crawler.py 文件)

```
sohu_template_url = 'http://v2.sohu.com/public-api/feed?scene=CHANNEL&sceneId=15&page=1&size={}'  
xinhuanet_template_url = 'http://qc.wa.news.cn/nodeart/list?nid=11147664&pgnum={}&cnt={}&tp=1&orderby=1'
```

将爬取的数据按照 BME0 标注格式进行标注得到命名实体数据集,因为爬取的数据较少,构成的命名实体个数较少,因此在此基础上下载了人民日报和 MSRA 的语料库,将语料库中的数据首先进行预处理,然后得到以 BME0 命名实体标注的数据,为了实现不同算法和数据标注之间性能的比较,同样将上面的数据集以 BIO 格式进行标注了一份。得到的数据文件 Mydata 文件目录如下图所示:

```
├── boson  
│   ├── data_util.py  
│   ├── fyz.test.embs  
│   ├── origindata.txt  
│   └── wordtagBIO.txt  
└── MSRA
```

```
|      fyz.train.embs
|      test1.txt
|      testright1.txt
|      train1.txt
|      train2.pkl.py
|      trainBIO.txt
|      wordtag.txt
|
└─renMinRiBao
    data_renmin_word.py
    devBIO.txt
    fyz.dev.embs
    renmin.txt
    renmintag.txt
```

图 3：数据集目录

说明：其中以 BIO 格式结尾的文件为 BIO 标注数据，以 embs 结尾的为 BME0 标注的数据，Py 文件为进行数据处理的 python 文件。

制作的数据集的结果：项目中总共标注的实体有三类：人名、地名、机构名：
总共的实体数目统计如下面三个图所示：

```
C:\Users\Administrator\Desktop\ChineseNER-master\ChineseNER-master\data\boson>python data_util.py
14801
finished tag
```

图 4. 14801 个实体

```
67653
finish tagging the data
```

图 5.MSAR 67653 个实体

```
Bao>python data_renmin_word.py
43900
data tag finished
```

图 6. 人民日报 43900 个实体

上面的三个部分别作为测试集、训练集、验证集。

二、NER 方法

本次采用两种方法进行命名实体识别，一种采用常规的 BiLSTM+CRF 模型，但是加上 BERT 方法产生词向量，提升模型的性能，另外一种方法则采用 Lattice-CRF 模型来实现命名实体识别。

BERT+BiLSTM+CRF 方法介绍

常规的 BiLSTM+CRF 模型在老师的课件中有相关介绍，因此这里不再赘述，这里采用加入 BERT 方法来实现 BiLSTM+CRF 模型，BERT 方法主要用来产生词向量。大致的实现流程是

- 1) 首先先将数据集制作成 BIO 格式，如图 1 所示。
- 2) 然后将数据集转换成 BERT 需要的格式，如图 7、图 8 所示。
- 3) 将数据封装成 record 形式，调用 BERT 官方的 API 实现数据读取、模型定义、模型训练。

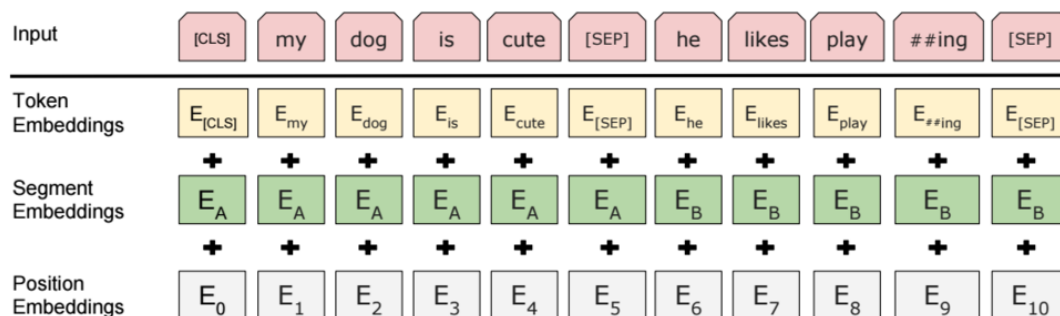


图 7. BERT 的数据格式

```
INFO:tensorflow:*** Example ***
INFO:tensorflow:guid: train-4
INFO:tensorflow:tokens: 以家乡的历史文献、特定历史时期书刊、某
一名家或名著的多种出版物为专题，注意精品、非卖品、纪
念品，集成系列，那收藏的过程就已经够您玩味无穷了。
INFO:tensorflow:input_ids: 101 809 2157 740 4638 1325 1380 3152 4346 510 4294 21
37 1325 1380 3198 3309 741 1149 510 3378 671 1399 2157 2772 1399 5865 4638 1914
4905 1139 4276 4289 711 683 7579 8024 3800 2692 5125 1501 510 7478 1297 1501 510
5279 2573 1501 8024 7415 2768 5143 1154 8024 6929 3119 5966 4638 6814 4923 2218
2347 5307 1916 2644 4381 1456 3187 4956 749 511 102 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0
```

图 8. BERT 实际转换数据结果

Lattice-CRF 模型方法介绍:

Lattice-CRF 为 2018ACL 上面的文章，主要的设计思路为在中文实体识别时加入分词特征，使用预训练的分词特征来进行匹配语料库中的命名实体，主要设计思路如下图 9 所示:

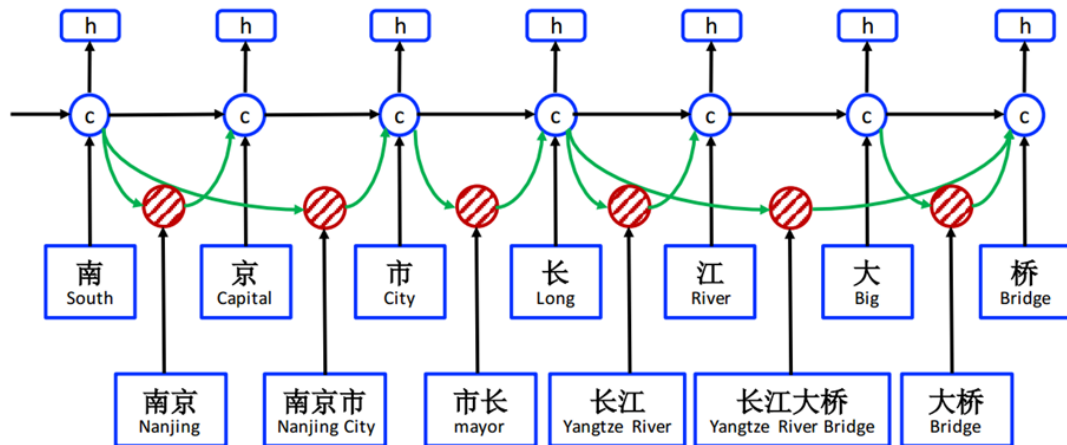


图 9. Lattice 模型设计思路

- 1) 模型实现的大致步骤为首先将数据标注为图 2 所示的 BEMO 格式。
- 2) 将预训练的分词特征加入进行词语间的匹配送入 RNN 模型当中。
- 3) 训练、测试模型。

三、NER 项目运行流程

BERT+BiLSTM+CRF 方法

实验平台：Ubuntu18.0 +k80 GPU

软件及环境要求：

TensorFlow 版本：

```
>>> import tensorflow
>>> import tensorflow as tf
>>> tf.__version__
'1.12.0'
```

Python==3.6

训练及测试方法：

Ubuntu 下：python3 bert_lstm_ner.py

Lattice-CRF 模型方法：

实验平台：Ubuntu18.0 +k80 GPU

软件及环境要求：

Pytorch 版本：

```
>>> print(torch.__version__)
1.0.0
>>>
```

Python==3.6

训练及测试方法：

Ubuntu 下：python3 main.py

四、NER 项目测试性能指标

BERT+BiLSTM+CRF 方法:

1) 在一个网上的较小的数据集上训练的结果

训练及测试结果如图 10 所示,红色的为测试的总的实体的准确度、召回率、F1 值,黄色的分别为三类不同的实体的识别的准确度、召回率、F1 值。

```
ROM_CKPT*
INFO:tensorflow: name = bert/pooler/dense/bias:0, shape = (768,), *INIT_FROM_CK
PT*
INFO:tensorflow: name = rnn_layer/bidirectional_rnn/fw/basic_lstm_cell/kernel:0
, shape = (896, 512)
INFO:tensorflow: name = rnn_layer/bidirectional_rnn/fw/basic_lstm_cell/bias:0,
shape = (512,)
INFO:tensorflow: name = rnn_layer/bidirectional_rnn/bw/basic_lstm_cell/kernel:0
, shape = (896, 512)
INFO:tensorflow: name = rnn_layer/bidirectional_rnn/bw/basic_lstm_cell/bias:0,
shape = (512,)
INFO:tensorflow: name = project/hidden/W:0, shape = (256, 128)
INFO:tensorflow: name = project/hidden/b:0, shape = (128,)
INFO:tensorflow: name = project/logits/W:0, shape = (128, 11)
INFO:tensorflow: name = project/logits/b:0, shape = (11,)
INFO:tensorflow: name = crf_loss/transitions:0, shape = (11, 11)
INFO:tensorflow:Done calling model_fn.
INFO:tensorflow:Graph was finalized.
INFO:tensorflow:Restoring parameters from /home/liuhaolin/spring_work/knowledge_
graph/NRE/BERT_BC_NER/output/model.ckpt-1956
INFO:tensorflow:Running local_init_op.
INFO:tensorflow:Done running local_init_op.
['processed 214542 tokens with 7450 phrases; found: 7589 phrases; correct: 7019.
\n', 'accuracy: 99.27%; precision: 92.49%; recall: 94.21%; FB1: 93.34\n', '
    LOC: precision: 93.69%; recall: 94.72%; FB1: 94.20 3502\n', '
    ORG: precision: 87.02%; recall: 90.35%; FB1: 88.65 2249\n', '
    PER: precision: 96.90%; recall: 97.86%; FB1: 97.38 1838\n']
INFO:tensorflow:prediction loop marked as finished
```

图 10. 测试结果图

具体的测试指标如下表 1 所示:

表 1.模型测试结果

实体类别	precision	recall	FB1
Total	92.49%	94.21%	93.34%
LOC	93.69%	94.72%	94.20%
ORG	87.02%	90.35%	88.65%
PER	96.90%	97.86%	97.38%

2) 在自己制作的数据集上训练同一个测试集上测试的结果

训练及测试结果如图 11 所示, 红色的为测试的总的实体的准确度、召回率、F1 值, 黄色的分别为三类不同的实体的识别的准确度、召回率、F1 值。

```
INFO:tensorflow: name = rnn_layer/bidirectional_rnn/fw/basic_lstm_cell/bias:0,
shape = (512,)
INFO:tensorflow: name = rnn_layer/bidirectional_rnn/bw/basic_lstm_cell/kernel:0
, shape = (896, 512)
INFO:tensorflow: name = rnn_layer/bidirectional_rnn/bw/basic_lstm_cell/bias:0,
shape = (512,)
INFO:tensorflow: name = project/hidden/W:0, shape = (256, 128)
INFO:tensorflow: name = project/hidden/b:0, shape = (128,)
INFO:tensorflow: name = project/logits/W:0, shape = (128, 11)
INFO:tensorflow: name = project/logits/b:0, shape = (11,)
INFO:tensorflow: name = crf_loss/transitions:0, shape = (11, 11)
INFO:tensorflow:Done calling model fn.
INFO:tensorflow:Graph was finalized.
INFO:tensorflow:Restoring parameters from /home/liuhaolin/spring_work/knowledge_
graph/NRE/BERT_BC_NER/output/model.ckpt-17386
INFO:tensorflow:Running local_init_op.
INFO:tensorflow:Done running local_init_op.
INFO:tensorflow:list index out of range
INFO:tensorflow:香港高尔夫球爱好者资助失学儿童
INFO:tensorflow:B-LOC I-LOC 0 0 0 0 0 0 0 0 0 0 0 0
INFO:tensorflow:list index out of range
INFO:tensorflow:● 定位——一个不能缺少关怀的空间
INFO:tensorflow:0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
['processed 214497 tokens with 7451 phrases; found: 7456 phrases; correct: 7207.
\n', 'accuracy: 99.69%; precision: 96.66%; recall: 96.73%; FB1: 96.69\n', '
LOC: precision: 97.77%; recall: 97.49%; FB1: 97.63 3454\n', '
ORG: precision: 93.71%; recall: 94.18%; FB1: 93.94 2177\n', '
PER: precision: 98.08%; recall: 98.30%; FB1: 98.19 1825\n']
INFO:tensorflow:prediction loop marked as finished
liuhaolin@iie-SYS-7048GR-TR:~/spring_work/knowledge_graph/NRE/BERT_BC_NER$
```

图 11. 测试结果图

具体的测试指标如下表 2 所示:

表 2. 模型测试结果

实体类别	precision	recall	FB1
Total	96.66%	96.73%	96.69%
LOC	97.77%	97.49%	97.63%
ORG	93.71%	94.18%	93.94%
PER	98.08%	98.30%	98.19%

实验结果分析: 对于该方法更换大的数据集以后模型的 F1 值显著提升, 提升大约 3 个百分点, 其他各项指标也显著提升。

Lattice-CRF 模型方法测试结果:

训练及测试结果如图 11 所示, 红色方框中的迭代 100 个 epoch 测试中文实体的准确度、召回率、F1 值, 以及速度。


```
Instance: 3000; Time: 61.42s; loss: 260.6708; acc: 95642.0/97488.0=0.9811
Instance: 3500; Time: 64.83s; loss: 277.8168; acc: 111682.0/113863.0=0.9808
Instance: 3821; Time: 41.10s; loss: 133.3419; acc: 121742.0/124099.0=0.9810
Epoch: 98 training finished. Time: 489.59s, speed: 7.80st/s, total loss: 2066.0
93017578125
gold_num = 1497 pred_num = 1507 right_num = 1405
Dev: time: 21.93s, speed: 21.12st/s; acc: 0.9676, p: 0.9323, r: 0.9385, f: 0.935
4
gold_num = 1630 pred_num = 1650 right_num = 1532
Test: time: 23.82s, speed: 20.03st/s; acc: 0.9630, p: 0.9285, r: 0.9399, f: 0.93
41
Epoch: 99/100
Learning rate is setted as: 9.348204032106312e-05
Instance: 500; Time: 61.16s; loss: 247.2622; acc: 15562.0/15856.0=0.9815
Instance: 1000; Time: 65.58s; loss: 284.8423; acc: 31873.0/32515.0=0.9803
Instance: 1500; Time: 65.07s; loss: 313.2357; acc: 47986.0/48962.0=0.9801
Instance: 2000; Time: 63.13s; loss: 247.9102; acc: 63675.0/64933.0=0.9806
Instance: 2500; Time: 63.46s; loss: 339.8730; acc: 79658.0/81281.0=0.9800
Instance: 3000; Time: 63.44s; loss: 226.6986; acc: 95521.0/97446.0=0.9802
Instance: 3500; Time: 64.26s; loss: 265.1793; acc: 111721.0/113945.0=0.9805
Instance: 3821; Time: 40.91s; loss: 193.1791; acc: 121665.0/124099.0=0.9804
Epoch: 99 training finished. Time: 486.99s, speed: 7.85st/s, total loss: 2118.1
80404663086
gold_num = 1497 pred_num = 1506 right_num = 1405
Dev: time: 22.17s, speed: 20.90st/s; acc: 0.9681, p: 0.9329, r: 0.9385, f: 0.935
7
gold_num = 1630 pred_num = 1648 right_num = 1531
Test: time: 24.20s, speed: 19.72st/s; acc: 0.9636, p: 0.9290, r: 0.9393, f: 0.93
41
```

具体的测试指标如下表 2 所示：

表 2.模型测试结果

实体类别	precision	recall	FB1
Total	92.90%	93.93%	93.41%

思考与比较：

Lattice-CRF 模型方法在 F1 值上要略微高于 BERT+BiLSTM+CRF 方法，但是它的模型更复杂训练所需的时间更长，而 BERT+BiLSTM+CRF 方法则因为使用了 BERT 词向量的方法使得模型的性能上升，并且训练的时间也相对于 Lattice-CRF 模型更少，因此从工程上来说，BERT+BiLSTM+CRF 方法无疑更加占优势。

五、附录：代码结构图

```

├──BERT_BC_NER
│   │   bert_lstm_ner.py
│   │   conlleval.pl
│   │   conlleval.py
│   │   data.conf
│   │   lstm_crf_layer.py
│   │   tf_metrics.py
│   │
│   └──bert
│       │   CONTRIBUTING.md
│       │   create_pretraining_data.py
│       │   extract_features.py
│       │   LICENSE
│       │   modeling.py
│       │   modeling_test.py
│       │   multilingual.md
│       │   optimization.py
│       │   optimization_test.py
│       │   README.md
│       │   requirements.txt
│       │   run_classifier.py
│       │   run_pretraining.py
│       │   run_squad.py
│       │   sample_text.txt
│       │   tokenization.py
│       │   tokenization_test.py
│       │   __init__.py
│       │
│       └──__pycache__
│           │   modeling.cpython-36.pyc
│           │   optimization.cpython-36.pyc
│           │   tokenization.cpython-36.pyc
│           │   __init__.cpython-36.pyc
│           │
│       └──checkpoint
│           └──chinese_L-12_H-768_A-12
│               bert_config.json
│               bert_config.json
│
└──NERdata
    │   dev.txt
    │   test.txt

```

NER-刘浩林

```
| | | train.txt
| | |
| | | └─data
| | |     dev.txt
| | |     train(1).txt
| | |
| | | └─ori
| | |     dev.txt
| | |     test.txt
| | |     train.txt
| | |
| | └─__pycache__
| |     conllevel.cpython-36.pyc
| |     lstm_crf_layer.cpython-36.pyc
| |     tf_metrics.cpython-36.pyc
| |
| └─lattice_NER
| |   fyz_run_decode.sh
| |   main.py
| |
| |   └─data
| |       └─model
| |   └─model
| |       bilstm.py
| |       bilstmcrf.py
| |       charbilstm.py
| |       charcnn.py
| |       crf.py
| |       latticelstm.py
| |       __init__.py
| |
| |   └─__pycache__
| |       bilstm.cpython-36.pyc
| |       bilstmcrf.cpython-36.pyc
| |       charbilstm.cpython-36.pyc
| |       charcnn.cpython-36.pyc
| |       crf.cpython-36.pyc
| |       latticelstm.cpython-36.pyc
| |       __init__.cpython-36.pyc
| |
| └─ResumeNER
| |   .demo.raw.out.swo
| |   .test.char.bmes.swp
| |   dev.char.bmes
```

NER-刘浩林

```
| | test.char.bmes
| | train.char.bmes
| |
| | └─test_data
| |     |
| |     |   fyz.dev.embs
| |     |   fyz.test.embs
| |     |   fyz.train.embs
| |     |
| |     └─utils
| |         |
| |         |   alphabet.py
| |         |   data.py
| |         |   functions.py
| |         |   gazetteer.py
| |         |   metric.py
| |         |   trie.py
| |         |   __init__.py
| |         |
| |         └─__pycache__
| |             |
| |             |   alphabet.cpython-36.pyc
| |             |   data.cpython-36.pyc
| |             |   functions.cpython-36.pyc
| |             |   gazetteer.cpython-36.pyc
| |             |   metric.cpython-36.pyc
| |             |   trie.cpython-36.pyc
| |             |   __init__.cpython-36.pyc
| |             |
| |             └─Mydata
| |                 |
| |                 |   └─boson
| |                 |       |
| |                 |       |   data_util.py
| |                 |       |   fyz.test.embs
| |                 |       |   origindata.txt
| |                 |       |   wordtag.txt
| |                 |       |
| |                 |       └─MSRA
| |                 |           |
| |                 |           |   fyz.train.embs
| |                 |           |   test1.txt
| |                 |           |   testright1.txt
| |                 |           |   train1.txt
| |                 |           |   train2pkl.py
| |                 |           |   trainBIO.txt
| |                 |           |   wordtag.txt
| |                 |           |
| |                 |           └─renMinRiBao
| |                 |               |
| |                 |               |   data_renmin_word.py
```

NER-刘浩林

```
|      dev.txt
|      fyz.dev.embs
|      renmin.txt
|      renmintag.txt
|
└─NewsCrawler
    |   main.py
    |
    └─news
        |   latest.txt
        |   latest_news.csv
        |   sohu.txt
        |   sohu_latest_news.csv
        |   xinhua.txt
        |   xinhuanet_latest_news.csv
        |
        └─utils
            |   news_crawler.py
            |   preprocessing.py
            |
            └─__pycache__
                |   news_crawler.cpython-37.pyc
                |   preprocessing.cpython-37.pyc
```