

智能医疗：中文电子病历命名实体识别（存档）

环境说明

- Keras == 2.2.0
- keras-contrib == 2.0.8
- numpy == 1.16.4
- h5py == 2.10.0
- Tensorflow == 1.10.0

注：此版本为 2020 年演示视频版本的代码复现最终版，采用 TensorFlow 1.10 版本实现。全部代码在以上系统环境下，均测试通过，运行记录可参考 PDF 文档。案例已升级至 2.0 版本，此存档版本不再进行框架升级、代码及程序报错、更新维护等 issues 支持。敬请谅解！

- 演示视频链接：https://www.bilibili.com/video/BV1uh411f7mC?spm_id_from=333.999.0.0
(https://www.bilibili.com/video/BV1uh411f7mC?spm_id_from=333.999.0.0)

1.数据转换

1.1 加载库

In [1]:

```
import os
from collections import Counter
```

1.2 设置路径

In [14]:

```
origin_path = 'data_origin'
train_filepath='train.txt'
train_path = 'data/train.txt'
vocab_path = 'model/vocab.txt'
embedding_file = 'model/token_vec_300.bin'
model_path = 'model/tokenvec_bilstm2_crf_model_20.h5'
```

1.3 序列标记

序列标记中，O非实体部分,TREATMENT治疗方式, BODY身体部位, SIGN疾病症状, CHECK医学检查, DISEASE疾病实体。

In [3]:

```
label_dict = {
    '检查和检验': 'CHECK',
    '症状和体征': 'SIGNS',
    '疾病和诊断': 'DISEASE',
    '治疗': 'TREATMENT',
    '身体部位': 'BODY' }

cate_dict = {
    '0': 0,
    'TREATMENT-I': 1,
    'TREATMENT-B': 2,
    'BODY-B': 3,
    'BODY-I': 4,
    'SIGNS-I': 5,
    'SIGNS-B': 6,
    'CHECK-B': 7,
    'CHECK-I': 8,
    'DISEASE-I': 9,
    'DISEASE-B': 10
}
```

1.4 数据转换

In [4]:

```
f = open(train_filepath, 'w+', encoding='utf-8')
count = 0
for root,dirs,files in os.walk(origin_path):
    for file in files:
        filepath = os.path.join(root, file)
        if 'original' not in filepath:
            continue
        label_filepath = filepath.replace('.txtoriginal', '')
        print(filepath, '\t\t', label_filepath)
        content = open(filepath, encoding='utf-8').read().strip()
        res_dict = {}
        for line in open(label_filepath, encoding='utf-8'):
            res = line.strip().split(' ')
            start = int(res[1])
            end = int(res[2])
            label = res[3]
            label_id = label_dict.get(label)
            for i in range(start, end+1):
                if i == start:
                    label_cate = label_id + '-B'
                else:
                    label_cate = label_id + '-I'
                res_dict[i] = label_cate

        for indx, char in enumerate(content):
            char_label = res_dict.get(indx, 'O')
            print(char, char_label)
            f.write(char + '\t' + char_label + '\n')
f.close()
```

data_origin\一般项目\一般项目-1.txtoriginal.txt
项目\一般项目-1.txt

女 0
性 0
, 0
8 0
8 0
岁 0
, 0
农 0
民 0
, 0
双 0
滦 0
区 0
应 0
营 0
子 0
村 0
' ^

data_origin\一般

2.模型建立

2.1 导入库

In [5]:

```
import numpy as np
from keras import backend as K
from keras.preprocessing.sequence import pad_sequences
from keras.models import Sequential
from keras.layers import Embedding, Bidirectional, LSTM, Dense, TimeDistributed, Dropout
from keras_contrib.layers.crf import CRF
import matplotlib.pyplot as plt
import os
```

Using TensorFlow backend.

2.2 读取训练集并构建数据

In [10]:

```

datas = []
sample_x = []
sample_y = []
vocabs = {'UNK'}

for line in open(train_path, encoding='utf-8'):
    line = line.rstrip().split('\t')
    if not line:
        continue
    char = line[0]
    if not char:
        continue
    cate = line[-1]
    sample_x.append(char)
    sample_y.append(cate)
    vocabs.add(char)
    if char in ['。', '，', '？', '！', '！', '，', '？']:
        datas.append([sample_x, sample_y])
        sample_x = []
        sample_y = []

word_dict = {wd:index for index, wd in enumerate(list(vocabs))}

with open(vocab_path, 'w+', encoding='utf-8') as f:
    f.write('\n'.join(list(vocabs)))

```

2.3 参数设置

In [11]:

```
class_dict = {
    'O': 0,
    'TREATMENT-I': 1,
    'TREATMENT-B': 2,
    'BODY-B': 3,
    'BODY-I': 4,
    'SIGNS-I': 5,
    'SIGNS-B': 6,
    'CHECK-B': 7,
    'CHECK-I': 8,
    'DISEASE-I': 9,
    'DISEASE-B': 10
}
EMBEDDING_DIM = 300
EPOCHS = 5
BATCH_SIZE = 128
NUM_CLASSES = len(class_dict)
VOCAB_SIZE = len(word_dict)
TIME_STAMPS = 150
```

2.4 转换数据为合适keras的格式

In [12]:

```
x_train = [[word_dict[char] for char in data[0]] for data in datas]
y_train = [[class_dict[label] for label in data[1]] for data in datas]
x_train = pad_sequences(x_train, TIME_STAMPS)
y = pad_sequences(y_train, TIME_STAMPS)
y_train = np.expand_dims(y, 2)
```

2.5 加载预训练词向量

In [15]:

```
embeddings_dict = {}
with open(embedding_file, 'r', encoding='utf-8') as f:
    for line in f:
        values = line.strip().split(' ')
        if len(values) < 300:
            continue
        word = values[0]
        coefs = np.asarray(values[1:], dtype='float32')
        embeddings_dict[word] = coefs
print('Found %s word vectors.' % len(embeddings_dict))
```

Found 20028 word vectors.

2.6 加载词向量矩阵

In [16]:

```
embedding_matrix = np.zeros((VOCAB_SIZE + 1, EMBEDDING_DIM))
for word, i in word_dict.items():
    embedding_vector = embeddings_dict.get(word)
    if embedding_vector is not None:
        embedding_matrix[i] = embedding_vector
```

2.7 使用预训练向量进行模型训练

In [19]:

```
model = Sequential()
embedding_layer = Embedding(VOCAB_SIZE + 1,
                             EMBEDDING_DIM,
                             weights=[embedding_matrix],
                             input_length=TIME_STAMPS,
                             trainable=False,
                             mask_zero=True)

model.add(embedding_layer)
model.add(Bidirectional(LSTM(128, return_sequences=True)))
model.add(Dropout(0.5))
model.add(Bidirectional(LSTM(64, return_sequences=True)))
model.add(Dropout(0.5))
model.add(TimeDistributed(Dense(NUM_CLASSES)))
crf_layer = CRF(NUM_CLASSES, sparse_target=True)
model.add(crf_layer)
model.compile('adam', loss=crf_layer.loss_function, metrics=[crf_layer.accuracy])
model.summary()
```

Layer (type)	Output Shape	Param #
=====		
embedding_2 (Embedding)	(None, 150, 300)	527700

bidirectional_3 (Bidirection	(None, 150, 256)	439296

dropout_3 (Dropout)	(None, 150, 256)	0

bidirectional_4 (Bidirection	(None, 150, 128)	164352

dropout_4 (Dropout)	(None, 150, 128)	0

time_distributed_2 (TimeDist	(None, 150, 11)	1419

crf_1 (CRF)	(None, 150, 11)	275
=====		
Total params: 1,133,042		
Trainable params: 605,342		
Non-trainable params: 527,700		

2.8 模型训练和保存

In [20]:

```
history = model.fit(x_train[:, y_train:], validation_split=0.2, batch_size=BATCH_SIZE, epochs=EPOCHS)
model.save(model_path)
```

Train on 6268 samples, validate on 1568 samples

Epoch 1/5

6268/6268 [=====] - 169s 27ms/step - loss: 18.4812 - crf_viterbi_accuracy: 0.7240 - val_loss: 15.8485 - val_crf_viterbi_accuracy: 0.7916

Epoch 2/5

6268/6268 [=====] - 182s 29ms/step - loss: 17.8324 - crf_viterbi_accuracy: 0.9184 - val_loss: 15.6522 - val_crf_viterbi_accuracy: 0.8235

Epoch 3/5

6268/6268 [=====] - 185s 29ms/step - loss: 17.7190 - crf_viterbi_accuracy: 0.9538 - val_loss: 15.6046 - val_crf_viterbi_accuracy: 0.8323

Epoch 4/5

6268/6268 [=====] - 187s 30ms/step - loss: 17.6835 - crf_viterbi_accuracy: 0.9639 - val_loss: 15.5773 - val_crf_viterbi_accuracy: 0.8402

Epoch 5/5

6268/6268 [=====] - 193s 31ms/step - loss: 17.6664 - crf_viterbi_accuracy: 0.9687 - val_loss: 15.5512 - val_crf_viterbi_accuracy: 0.8470

3. 模型调用

3.1 导入库

In [1]:

```
import numpy as np
from keras.preprocessing.sequence import pad_sequences
from keras.models import Sequential
from keras.layers import Embedding, Bidirectional, LSTM, Dense, TimeDistributed, Dropout
from keras_contrib.layers import CRF
import os
```

Using TensorFlow backend.

In [2]:

```
vocab_path = 'model/vocab.txt'
embedding_file = 'model/token_vec_300.bin'
model_path = 'model/tokenvec_bilstm2_crf_model_20.h5'
```

3.2 加载词表

In [3]:

```
vocabs = [line.strip() for line in open(vocab_path, encoding='utf-8')]
word_dict = {wd: index for index, wd in enumerate(vocabs)}
```

3.3 设置参数

In [4]:

```
class_dict = {
    'O': 0,
    'TREATMENT-I': 1,
    'TREATMENT-B': 2,
    'BODY-B': 3,
    'BODY-I': 4,
    'SIGNS-I': 5,
    'SIGNS-B': 6,
    'CHECK-B': 7,
    'CHECK-I': 8,
    'DISEASE-I': 9,
    'DISEASE-B': 10
}
label_dict = {j:i for i,j in class_dict.items()}
EMBEDDING_DIM = 300
EPOCHS = 10
BATCH_SIZE = 128
NUM_CLASSES = len(class_dict)
VOCAB_SIZE = len(word_dict)
TIME_STAMPS = 150
```

3.4 给定输入并转换成匹配格式

In [5]:

```
s = input('enter an sent:').strip()

ss = []
for char in s:
    if char not in word_dict:
        char = 'UNK'
    ss.append(word_dict.get(char))
ss = pad_sequences([ss], TIME_STAMPS)
#口腔溃疡可能需要多吃维生素
```

enter an sent:口腔溃疡可能需要多吃维生素

3.5 加载预训练词向量

In [6]:

```
embeddings_dict = {}
with open(embedding_file, 'r', encoding='utf-8') as f:
    for line in f:
        values = line.strip().split(' ')
        if len(values) < 300:
            continue
        word = values[0]
        coefs = np.asarray(values[1:], dtype='float32')
        embeddings_dict[word] = coefs
print('Found %s word vectors.' % len(embeddings_dict))
```

Found 20028 word vectors.

3.6 加载词向量矩阵

In [7]:

```
embedding_matrix = np.zeros((VOCAB_SIZE + 1, EMBEDDING_DIM))
for word, i in word_dict.items():
    embedding_vector = embeddings_dict.get(word)
    if embedding_vector is not None:
        embedding_matrix[i] = embedding_vector
```

3.7 使用预训练矩阵进行模型搭建

In [8]:

```
model = Sequential()
embedding_layer = Embedding(VOCAB_SIZE + 1,
                            EMBEDDING_DIM,
                            weights=[embedding_matrix],
                            input_length=TIME_STAMPS,
                            trainable=False,
                            mask_zero=True)

model.add(embedding_layer)
model.add(Bidirectional(LSTM(128, return_sequences=True)))
model.add(Dropout(0.5))
model.add(Bidirectional(LSTM(64, return_sequences=True)))
model.add(Dropout(0.5))
model.add(TimeDistributed(Dense(NUM_CLASSES)))
crf_layer = CRF(NUM_CLASSES, sparse_target=True)
model.add(crf_layer)
model.compile('adam', loss=crf_layer.loss_function, metrics=[crf_layer.accuracy])
model.summary()
```

D:\Anaconda3\envs\tf1\lib\site-packages\keras_contrib-2.0.8-py3.6.egg\keras_contrib\layers\crf.py:346: UserWarning: CRF.loss_function is deprecated and it might be removed in the future. Please use losses.crf_loss instead.
D:\Anaconda3\envs\tf1\lib\site-packages\keras_contrib-2.0.8-py3.6.egg\keras_contrib\layers\crf.py:353: UserWarning: CRF.accuracy is deprecated and it might be removed in the future. Please use metrics.crf_accuracy

Layer (type)	Output Shape	Param #
embedding_1 (Embedding)	(None, 150, 300)	527700
bidirectional_1 (Bidirection	(None, 150, 256)	439296
dropout_1 (Dropout)	(None, 150, 256)	0
bidirectional_2 (Bidirection	(None, 150, 128)	164352
dropout_2 (Dropout)	(None, 150, 128)	0
time_distributed_1 (TimeDist	(None, 150, 11)	1419
crf_1 (CRF)	(None, 150, 11)	275
Total params: 1,133,042		
Trainable params: 605,342		
Non-trainable params: 527,700		

3.8 加载模型

In [9]:

```
model.load_weights(model_path)
```

3.9 利用模型进行划分显示

In [10]:

```
raw = model.predict(ss)[0][-TIME_STAMPS:]
result = [np.argmax(row) for row in raw]
chars = [i for i in s]
tags = [label_dict[i] for i in result][len(result)-len(s):]
res = list(zip(chars, tags))
print(res)
```

```
[('口', 'BODY-B'), ('腔', 'BODY-I'), ('溃', 'O'), ('疡', 'O'), ('可', 'O'), ('能', 'O'), ('需', 'O'), ('要', 'O'), ('多', 'O'), ('吃', 'O'), ('维', 'DISEASE-B'), ('生', 'DISEASE-I'), ('素', 'DISEASE-I')]
```