包方法

自定义方法/类

文件夹

文件

变量

**embedding\_model.py:**

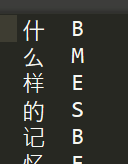
MySentences读取corpus中每个文件夹中的文件（内容为已经切分过的法律文档）

-->将文件中所有根据空格分开的词存入words变量并将其返回给变量sentences

-->gensim.models.Word2Vec将sentences中词语变为100维的向量，窗口设置为5，最小词频为1，最大迭代次数100，并存入model\_conll\_law.m

**bilstm\_cnn\_crf.py**

----🡪运行process\_train方法，参数设置为：语料路径corpus，迭代轮数nb\_epoch=1，base\_model\_weight = train\_model.hdf5

----🡪方法中：raw\_train\_file列表存放corpus中所有文件夹中的文件路径，运行process\_data( raw\_train\_file, ’train.data’)，将raw\_train\_file中每个字都上标注，存入train.data中

----🡪运行create\_documents('train.data')，读取的每一条序列保存在一个实例化的“documents”类中“documents”类包含三个属性，“chars”，“labels”是两个数组，分别存放序列中每个字的内容和标签。第三个属性“index”则保存序列的序列号。读取序列的方式是逐行读取，在遇到句子的切分标签“，”“。”“；”时，也算作一行序列的结束而进入下一次循环.将所有数据保存在“documents”数组类型的变量**train\_documents**中

----🡪 get\_lexicon(train\_documents)生成词典lexicon, lexicon\_reverse

----🡪embedding\_model读取model\_conll\_law.m，embedding\_size为100

----🡪create\_embedding

(embedding\_model, embedding\_size, lexicon\_reverse)得到**embedding\_weights**存储(字典长\*100)矩阵，embedding\_weights[i]存储字典lexicon\_reverse中i所对应字的100维向量

----🡪设置label\_2\_index和index\_2\_label

----🡪create\_matrix将train\_documents中的数据的字典index，标签的数字形式，所属document的index分别存入train\_data\_list，train\_label\_list，train\_index\_list（这个没用）

----🡪随后，使用padding\_sentences函数对长度不足的行数据按照最长的序列的长度max\_len(为188)，使用“0”作为填充值进行补长，最终的有价值的输出为：train\_data\_array，train\_label\_array

----🡪model =Bilstm\_CNN\_Crf(max\_len,len(lexicon),len(label\_2\_index),embedding\_weights)

----🡪判断base\_model\_weight是否为空（即是否第一次训练），若否则model导入base\_model\_weight

----🡪hist=model.fit(train\_data\_array,train\_label\_array,batch\_size=256,epochs=nb\_epoch,verbose=1)

----🡪score=model.evaluate(train\_data\_array,train\_label\_array,batch\_size=512)

----🡪 model.save\_weights('train\_model.hdf5')

----🡪pickle.dump([lexicon,lexicon\_reverse,max\_len,index\_2\_label],open('lexicon.pkl','wb'))