

网络工程

系统开发实训报告

题目名称：面向中文电子病历的命名实体识别

班 级：网络153班

学 号：201508030323

学生姓名：张涛

指导教师：刘小明

2018 年 7 月

# **中文摘要**

命名实体识别是自然语言处理任务的重要步骤。近年来，不依赖人工特征的神经网络在新闻等通用领域命名实体识别方面表现出了很好的性能。然而在生物医学领域 ，许多实验表明基于领域知识的人工特征对于神经网络模型的结果影响很大。因此，如何在不依赖人工特征的情况下获得较好的生物医学命名实体识别性能是有待解决的问题。该文主要是基于CRF++（Conditional Rondom Fields 条件随机场）和ANN（Artificial Neural Networks 人造神经网络）的模型。

首先利用CRF++对数据集进行分词（segment）和序列标注（label sequential），然后通过代码对标注好的数据集进行格式转换，再利用ANN模型训练出我们需要的model以及得到相应的测评指标，即：精确率（Precision）、召回率（Recall）以及F1-Measure。

实验结果表明，不依赖任何人工特征，该文方法在CCKS2018

官方给出的生物医学语料上得到的评测指标是Precision:89.35% ，Recall:87.23% ，F1:88.28%。

关键词：命名实体识别；CRF++；ANN；精确度（Precision）；召回率（Recall）；F1-Measure

# **目 录**

[中文摘要 I](#_Toc523672666)

[目 录 II](#_Toc523672667)

[第1章 相关基础技术 1](#_Toc523672668)

[1.1实训内容介绍 1](#_Toc523672669)

[1.2 CRF++理论和应用场景 1](#_Toc523672670)

[1.2.1 概率无向图模型 2](#_Toc523672671)

[1.2.2 概率无向图的因子分解 2](#_Toc523672672)

[1.2.3 条件随机场的定义 3](#_Toc523672673)

[1.2.4 条件随机场的参数化形式 4](#_Toc523672674)

[1.2.5 条件随机场的概率计算问题 5](#_Toc523672675)

[1.2.6 条件随机场的学习算法 5](#_Toc523672676)

[1.2.7 条件随机场的预算算法 7](#_Toc523672677)

[1.2.8 应用场景 8](#_Toc523672678)

[1.3 ANN（人工神经网络）模型 14](#_Toc523672679)

[1.3.1什么是ANN? 14](#_Toc523672680)

[1.3.2 ANN的基本结构 14](#_Toc523672681)

[1.3.3 ANN的类型 16](#_Toc523672682)

[1.3.4 ANN如何工作的？ 17](#_Toc523672683)

[1.3.5利用ANN机器学习 17](#_Toc523672684)

[1.3.6贝叶斯网络（Bayesian Networks） 18](#_Toc523672685)

[1.3.7神经网络的应用领域 19](#_Toc523672686)

[第2章 系统分析 20](#_Toc523672687)

[2.1系统需求分析 20](#_Toc523672688)

[2.2系统概要设计 20](#_Toc523672689)

[2.3系统详细设计 21](#_Toc523672690)

[2.3.1数据集预处理 21](#_Toc523672691)

[2.3.2测试结果检查 21](#_Toc523672692)

[2.3.3应用 22](#_Toc523672693)

[第3章 系统开发 23](#_Toc523672694)

[3.1 系统开发环境配置 23](#_Toc523672695)

[3.2 系统开发 23](#_Toc523672696)

[3.2.1开发详细过程 23](#_Toc523672697)

[3.2.2遇到的问题及解决方案 60](#_Toc523672698)

[第4章 系统测试 62](#_Toc523672699)

[4.1测试环境 62](#_Toc523672700)

[4.2功能测试 64](#_Toc523672701)

[4.3测试结果分析 66](#_Toc523672702)

[第5章 结 论 71](#_Toc523672703)

[总结 71](#_Toc523672704)

[参考文献 72](#_Toc523672705)

[致 谢 73](#_Toc523672706)

[附 录 74](#_Toc523672707)

[附录: 详细的代码清单 74](#_Toc523672708)

# 第1章 相关基础技术

## 1.1 实训内容介绍

本次实训要实现面向中文电子病历的命名实体识别，采用了基于CRF++（Conditional Rondom Fields 条件随机场）和ANN（Artificial Neural Networks 人造神经网络）的模型。具体任务是对于给定的一组电子病历纯文本文档，任务的目标是识别并抽取出与医学临床相关的实体提及（entity mention），并将它们归类到预定义类别（pre-defined categories），比如症状、药品、手术等。

结合数据源“现病史记录”的内容及特点，本任务实体类型聚焦在症状、药物、手术三大类。由于症状类型实体多表现为结构化形式，本次任务将症状类型进一步细化，划分为三类：解剖部位 （复合症状的主体）、症状描述（复合症状的描述）及独立症状。最终将本任务预定义类别限定为以下五类：

1. 解剖部位：由多种组织构成能行使功能的结构单位，例如“腹部”
2. 症状描述：指患者患病后对机体生理功能异常的自身体验和感觉，同时需与解剖部位联合输出，例如“不适”，需与腹部组合输出“腹部不适”
3. 独立症状：指患者患病后对机体生理功能异的自身体验和感觉，可独立输出，例如“眩晕”
4. 药物：用来治疗、预防或促进健康的一种化学物质
5. 手术：指医生用医疗器械对病人身体进行的切除、缝合等治疗

备注：

实验中5种特征实体对应的英文名称分别是：AnatomicSite(解剖部位)，SymptomsDescibed (症状描述)，IndependentSymptoms(独立症状)，Medicine(药物)，Operation(手术)

## 1.2 CRF++理论和应用场景

条件随机场（CRF）是给定一组输入随机变量条件下，求另一组输出随机变量的条件概率分布的模型；其特点是假设输出随机变量构成马尔科夫随机场（后面解释）。条件随机场可以用于不同的预测问题，对自然语言处理过程主要是线性（linear chain）条件随机场，这时，问题变成了由输入序列对输出序列预测的判别模型，形式为对数 线性模型，学习方法为极大似然估计或者正则化的极大似然估计。条件随机场和隐马类似，对应得得三个基本问题：**概率计算问题、学习问题和预测问题**。

### 1.2.1 概率无向图模型

概率无向图模型（probabilistic undirected graphical model）又称为马尔科夫随机场，是一个可以由无向图表示的联合概率分布。

定义：设有联合概率分布于P(Y)，由无向图G=(V,E)表示，在图G中，节点表示随机变量，边表示随机变量之间的依赖关系。如果联合概率分布P(Y)满足成对、局部或全局马尔科夫性，就称此联合概率分布为概率无线图模型

成对、局部或全局马尔科夫性，大白话就是说每一个节点的分布只和有边相连的节点有关系，如图1-1。

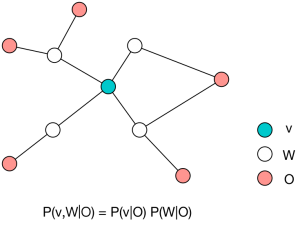


图1-1

### 1.2.2 概率无向图的因子分解

因子分解是建立在最大团的理论之上，这里先介绍最大团

**定义（团和最大团）**：无向图G中任何两个节点均有边链接的节点自己称为团（clique）。若C是无向图的一个团，并且不能再加入任何一个G的节点使其成为一个更大的团，则称此C为最大团（maximal clique）。

下图中有两个节点组成的团有五个：{A,B},{B,C},{A,C},{C,D},{B,D};有两个最大团：{A,B,C},{B,C,D}；而{A,B,C,D}不是一个团，因为A和D之间没有边，如图1-2。

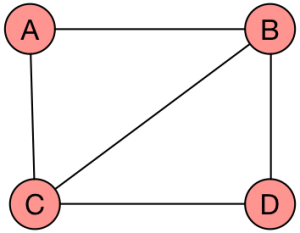


图1-2

**定义（因子分解）**：将概率无向图模型的联合概率分布表示为其上最大团的随机变量的函数的乘机的形式的操作，成为概率无向图模型的因子分解（factorization）。

给定无向图模型G，其中C表示的是最大团， YC 表示C对应的随机变量，联合概率分布P(Y)可写作图中所有最大团C上的函数 φ C(YC) 的乘积形式：

 公式1

其中Z是规范化因子：

 公式2

规范化因子保证P(Y)构成一个概率分布，函数φ C称为势函数，这里要求势函数是严格正的，通常定义的为指数函数

 公式3

### 1.2.3 条件随机场的定义

定义（条件随机场）：设X和Y是随机变量，P(Y|X)是在给定X的条件下的Y的条件分布，若随机变量Y构成一个无向图G=(V,E)表示的马尔科夫场，即：

公式4

对任意的节点v成立，则称条件概率分布P(Y|X)为条件随机场。其中 w∼v 表示在图G=(V,E)中，节点v有边链接的所有节点w， w≠v表示节点v以外的所有节点。

定义中并没有要求X和Y具有相同的图结构，现实中，一般假设X和Y具有相同的图结构。本文主要讨论下面两种线性链的情况，如图1-3。

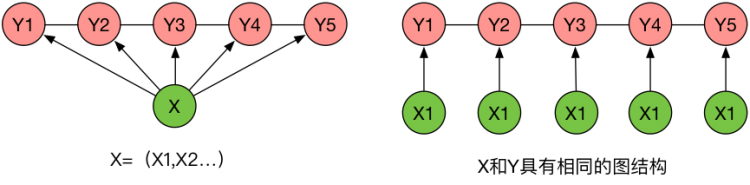


图1-3

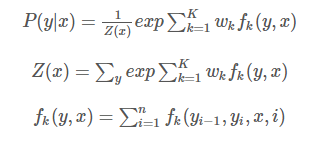
根据定义，线性链的的条件随机场可以表示为（请不要纠结边界）：

公式5

### 1.2.4 条件随机场的参数化形式

线性条件随机场中每个最大团都包含两个节点，当前节点为i前一个节点为i-1。特征粒度是在最大团的粒度，并且是两个相邻节点{i-1, i}。一般来说特征分为两类，作用在边上的特征和作用在节点上的特征，边上的第m维特征用 tm 表示，节点上的第l维特征用 sl 表示。每个团的特征向量为(t1,t2,...,tm,...,tM,s1,s2,...sl,...,sL) ，为了表示方便，通常用 (f1,f2,...,fM,fM+1,....,fM+L)表示；类似线性模型，每个特征都要学习出来一个权重 w=(w1,w2,...,wM+L)。

用公式表示即为：



公式6

### 1.2.5 条件随机场的概率计算问题

在已知参数和求给定观测序列的概率的时候，只需要依次从前到后计算每一个最大团的向量乘积即可。

### 1.2.6 条件随机场的学习算法

具体的实现算法有改进的迭代尺度法(Improved Iterative Scaling)、梯度下降法以及拟牛顿法。这里详述一下改进的迭代尺度法，其他算法如需了解学习可自行网上查阅。

改进的迭代尺度算法（IIS）：

改进的迭代尺度法，在很多模型求解中用到，比如最大熵、CRFs等，对模型是对数线性模型的似然都适用。这个算法的思想也很简单，通俗的理解就是通过两个不等式变形优化下界，从而迭代到收敛的算法。

用到两个不等式，对 α>0：

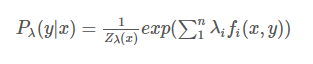
−logα≥1−α 公式7

对 p(x) 是一个概率密度函数

 公式8

为了阅读方便，可以先记住上面两个不等式，至于为什么用这两个不等式变换，文章末尾有我个人简单的思考。

假设模型为：

 公式9

其中 λ 是参数，Z(x)是归一化因子。

似然函数可以写成：

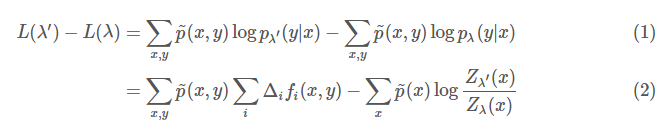
 公式10

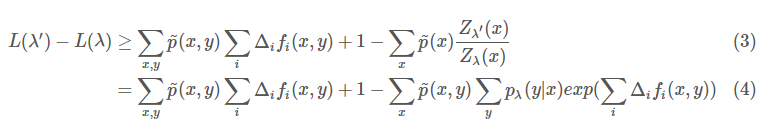
其中是样本（x,y）出现的频率。

接下来就是要找到合适的 λ ，找 λ 的过程可以看做是下面迭代过程：

 公式11

所以问题可以看做每次寻找一个 λ的移动向量，然后不断迭代，接下来就是确定每一步如何找到。一种容易想到的做法就是通过最大化两次迭代的差值，从而实现每一步得到最优的 。

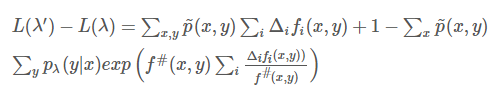
  
对上面式子通过不等式 −logα≥1−α可以改写为：



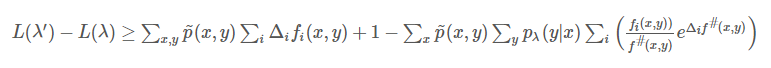
定义：



可以将上式修改为：



对上面等式通过不等式：可以改写为：



上式直接对 求导数，令导函数为0可以直接解出来 ，从而不断迭代达到收敛。

思考：

不等式的选择问题，对于第一个不等式 −logα≥1−α 画图之后的效果：如图1-4。

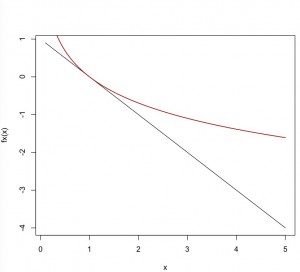


图1-4

在坐标逼近（1，0）处，他们两个的值是无限接近，当迭代收敛的时候，刚好这个点。

第二个不等式以前没有接触过，但是容易想到的是凸函数的期望不等式，『期望的函数』小于等于『函数的期望』:



### 1.2.7 条件随机场的预算算法

预测算法和HMM类似，是维特比算法。

预测问题，就是从多个候选标注中挑选出来一种标注概率最高的。由于归一化因子不影响值的比较，所以只需要比较分子部分的『非规范化概率』。这个求最优的过程可以通过数学表示为：



其中，



接下来就是使用维特比算法对每一个位置i，计算标注为 cj的最大概率，对于每一个位置有多个可能的标注，定义 ϱi(j)表示第i个位置，标注序列是 cj结束的最大概率，所以动态转移方程可以表示为：



对第一个位置可以表示为：



计算的时候记录一下路径就可以了。

### 1.2.8 应用场景

1.CRF++词性标注：

举一个具体应用的例子来说明：

训练和测试的语料都是人民日报98年标注语料，训练和测试比例是10：1，直接通过CRF++标注词性的准确率:0.933882。特征有一千多万个，训练时间比较长。机器cpu是48核，通过crf++，指定并线数量 -p为40，训练了大概七个小时才结束。

语料库、生成训练数据的python脚本、训练日志、模型、计算准确率脚本都上传到网盘，可以直接下载：[戳我下载 CRF++词性标注](http://pan.baidu.com/s/1bokPQLx)，程序在centos6.5+python2.7下面运行通过，如果在win下或者ubuntu下可能会有异常，通常都是编码、路径规范等小问题，通过逐行debug脚本应该很容易找到问题，同时要确定crf++在自己机器本身编译没有问题，下面说一下每一步的过程。

1. 生成训练和测试数据

生成训练和测试数据脚本：get\_post\_train\_test\_data.py，执行过程中会打印出来一些调试信息。

1. 执行训练和测试数据

设置模板为：

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | # Unigram  U00:%x[-2,0]  U01:%x[-1,0]  U02:%x[0,0]  U03:%x[1,0]  U04:%x[2,0]  U05:%x[-1,0]/%x[0,0]  U06:%x[0,0]/%x[1,0] |

训练的时候的-p参数根据自己机器情况设置

|  |  |
| --- | --- |
| 1  2 | crf\_learn -f 3 -p 4 -c 4.0 template train.data model > train.rst  crf\_test -m model test.data > test.rst |

1. 计算准确率

通过命令：python clc\_f.py  test.rst 执行python脚本

1. 实验结果

|  |  |
| --- | --- |
|  | WordCount from result: 98369  WordCount of correct post : 91865  准确率:0.933882 |

2.CRF++中文分词：

同样也是举一个具体应用的例子来说明：

使用人民日报的语料，为了方便切割，将其中的\t替换为了空格，语料直接下载：[人民日报语料](http://pan.baidu.com/s/1qXtmyde)。生成的crf测试和训练数据：[点击-训练和测试数据下载（6tag版本）](http://pan.baidu.com/s/1jHdP6aM)

对于语料有嵌套的标注，例如：[中央/n 电视台/n]nt，为了处理方便，只考虑最细粒度的分词结果，即当作是 中央/n 电视台/n 两个词进行处理。生成训练数据的python脚本是来自 [李海波的博客](http://blog.csdn.net/marising/article/details/5769653)，由于粘贴的时候缩进错乱导致不可用，所以对脚本进行了修改。

1. 生成训练和测试数据

通过下面python脚本，根据人民日报的语料库生成crf的测试和训练数据。原始数据中随机10%是测试数据，90%是训练数据。程序打印出来了不少调试信息，可以忽略。生成训练数据的时候，支持4tag和6tag两个格式，6tag的格式是：

**S，单个词；B，词首；E，词尾；M1/M2/M，词中**

4tag和6tag的区别就是没有词中顺序状态。

生成数据时，通过最后一个参数是4还是6区别生成4tag数据还是6tag数据。

生成6tag

python get\_crf\_train\_test\_data.py 6

生成4tag

python get\_crf\_train\_test\_data.py 4

1. 使用模板

# Unigram

U00:%x[-1,0]

U01:%x[0,0]

U02:%x[1,0]

U03:%x[-1,0]/%x[0,0]

U04:%x[0,0]/%x[1,0]

U05:%x[-1,0]/%x[1,0]

1. 训练和测试

|  |  |
| --- | --- |
| 2  3 | #!/bin/sh  crf\_learn -f 3 -c 4.0 template 6.train.data 6.model > 6.train.rst  crf\_test -m 6.model 6.test.data > 6.test.rst |

1. 计算F值

|  |  |
| --- | --- |
| 2  3  4  5  6  7 | #查全率      P = wc\_of\_correct/float(wc\_of\_test)      #查准率，召回率      R = wc\_of\_correct/float(wc\_of\_gold)      print "P = %f, R = %f, F-score = %f" % (P, R, (2\*P\*R)/(P+R)) |

1. 计算结果

执行： python crfeval.py 6.test.rst

1. 得到结果：

WordCount from test result: 55445

WordCount from golden data: 55487

WordCount of correct segs : 51687

P = 0.932221, R = 0.931515, F-score = 0.931868

## 1.3 ANN（人工神经网络）模型

人工智能的另一个研究领域是神经网络，它的灵感来自人类神经系统的自然神经网络。

### 1.3.1什么是ANN?

第一个神经计算机的发明者, Dr. Robert Hecht-Nielsen定义了一个神经网络：

"...a computing system made up of a number of simple, highly interconnected processing elements, which process information by their dynamic state response to external inputs.”

译文：“一种由许多简单的、高度互联的处理元素组成的计算系统，这些元素通过它们对外部输入的动态状态响应来处理信息。”

### 1.3.2 ANN的基本结构

人工神经网络的概念基于这样一种信念，即人类大脑通过制造正确的连接来工作，可以用硅和电线作为活的**神经元**（neurons）和**树突**（dendrites）进行模仿。

人类的大脑由860亿个叫做**神经元**（neurons）的神经细胞组成。它们通过**轴突**（Axons）与其他上千个细胞相连。来自外部环境的刺激或来自感觉器官的输入被树突接受。这些输入产生了电脉冲，电脉冲通过神经网络迅速传播。然后，神经元可以将消息发送给其他神经元来处理这个问题，或者不发送，如图1-4。

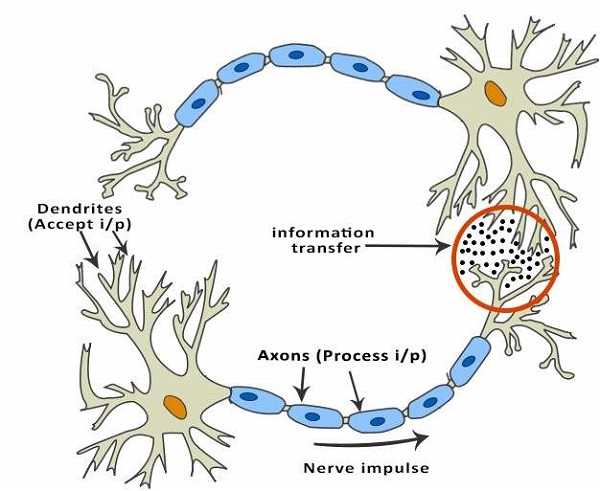


图1-5

人工神经网络由多个节点（nodes）组成，它们模仿人类大脑的生物神经元(neurons)。神经元是通过链接连接起来的，它们相互作用。节点可以接受输入数据并对数据执行简单的操作。这些操作的结果被传递给其他神经元。每个节点的输出称为其激活（activation）或节点值（node value）。

每个连杆都与权重（weight）有关。ANNs能够通过改变权重值来学习。下面的插图显示了一个简单ANN结构说明，如图1-5。

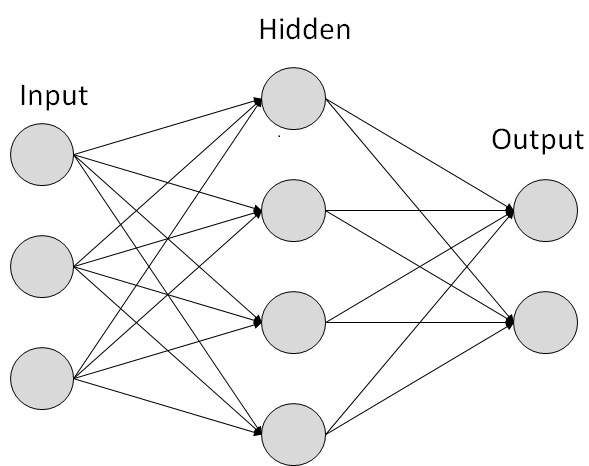


图1-6

### 1.3.3 ANN的类型

有两种人工神经网络拓扑：前馈（FeedForward）和反馈（Feedback）

1. FeedForward ANN

信息流是单向的。单位向不接收任何信息的其他单位发送信息。没有反馈环。它们用于模式生成/识别/分类。它们已经固定了输入和输出，如图1-6。

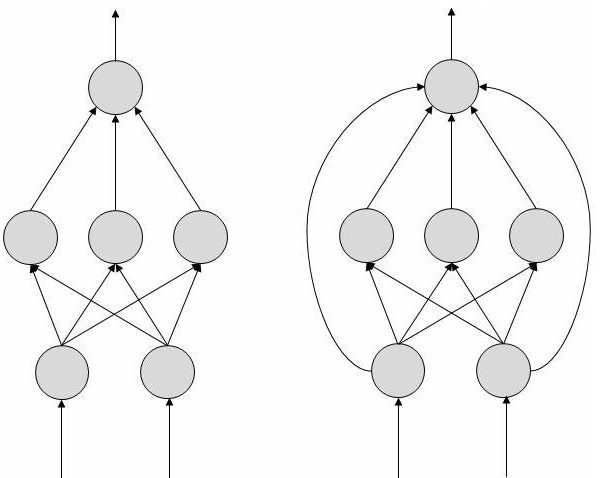


图1-7

1. FeedBack ANN

在这里，反馈循环是允许的。它们用于内容可寻址的记忆，如图1-7。

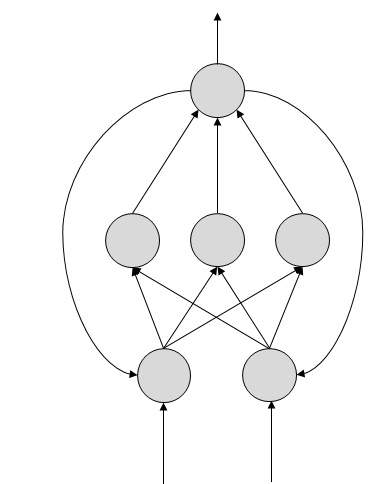


图1-8

### 1.3.4 ANN如何工作的？

在图中所示的拓扑图中，每个箭头表示两个神经元之间的连接，并指示信息流动的路径。每个连接都有一个权值，一个控制两个神经元之间信号的整数。

如果网络产生“好的或想要的”输出，则不需要调整权重。但是，如果网络产生“糟糕的或不希望的”输出或错误，那么系统就会改变权重，以改进后续的结果。

### 1.3.5利用ANN机器学习

ANNs有学习能力，需要训练。有几种学习策略：

1. 监督式学习

它涉及一个比ANN本身更有学问监督者。例如，监督者提供一些示例数据，这些数据是监督者已经知道答案的。

例如,模式识别。ANN在识别的同时给出了猜测。然后监督者给ANN答案。然后ANN将它的猜测与监督者的“正确”答案进行比较，并根据错误进行调整。

1. 无监督学习

这样做需要没有已知答案的示例数据集。例如，搜索一个隐藏的模式。在这种情况下基于现有的数据集，将一组元素按照某个未知的模式划分为组进行聚类。

1. 强化学习

这种策略建立在观察的基础上。ANN通过观察周围的环境来做决定。如果观察值为负，则ANN会调整其权重值，以便下次能够做出不同的必要决定。

1. 反响传播算法(Back Propagation Algorithm)：

它是训练或学习算法。它通过例子去学习。如果你向算法提交你想要网络做什么的例子，它会改变网络的权重，这样它就能在完成培训的时候产生期望的输出。

反向传播网络非常适合于简单的模式识别和映射任务。

### 1.3.6贝叶斯网络（Bayesian Networks）

它们是用来表示一组随机变量之间的概率关系图形结构。贝叶斯网络也称为贝氏网络（Belief Networks）或贝叶斯网络（Bayes Nets）。这也是BNs不确定域的原因。

在这些网络中，每个节点代表一个具有特定命题的随机变量。例如，在医学诊断领域，癌症节点代表病人患癌症的命题。

连接节点的边表示这些随机变量之间的概率依赖性。如果在两个节点中，一个影响另一个节点，那么它们必须直接连接到影响的方向上。变量之间关系的强度由与每个节点相关的概率来量化。

对于BN中的弧，只有一个约束，您不能仅仅通过跟随有向弧返回到节点。因此，BNs被称为有向无环图(DAGs)。

BNs能够同时处理多值变量。这里的BN变量由两个维度组成。即：一是介词的范围，二是给每个介词赋值的概率

考虑离散随机变量的有限集X = {X1, X2，…，Xn}，其中每个变量Xi可以从一个有限集合中取值，用Val(Xi)表示。如果变量Xi和变量Xj之间有一个有向链接，那么变量Xi将是变量Xj的父元素，表示变量之间的直接依赖关系。

BN的结构是结合先验知识和观测数据的理想结构。BN可以用来学习因果关系，理解各种问题领域，预测未来的事件，即使是在丢失数据的情况下。

### 1.3.7神经网络的应用领域

它们可以执行人类容易但机器难的任务：

航空航天−自动驾驶飞机、飞机故障检测。

汽车−汽车导航系统。

军事−武器方向和指导,目标跟踪,反对歧视,面部识别、信号/图像识别。

电子−代码序列预测、集成电路芯片布局、芯片失效分析、机器视觉、语音合成。

金融−房地产评估、贷款顾问,抵押贷款筛选,公司债券评级,投资组合交易计划,企业财务分析,货币值预测,文档的读者,信贷应用评估。

工业−生产过程控制、产品设计和分析,质量检验系统,焊接质量分析,论文质量预测,化学产品设计分析、化学过程的动态模拟系统,机器保养分析、项目投标、规划和管理。

医疗−癌症细胞分析、心电图和脑电图分析,假体设计、移植时间优化器。

语音−语音识别,语音分类,从文本到语音转换。

电信−图像和数据压缩、自动化信息服务,实时口语翻译。

运输−卡车制动系统诊断、车辆调度、路由系统。

面部识别软件−模式识别,光学字符识别等。

时间序列预测−人工神经网络用于预测股票和自然灾害。

信号处理−可以训练神经网络处理音频信号和筛选适当的助听器。

控制−ANN通常被用来指导决策的物理车辆。

异常检测−ANN专家识别模式,他们也可以被训练来生成一个输出当发生一些不寻常的东西,不适应模式。

# 第2章 系统分析

## 2.1系统需求分析

功能1：中文分词。

功能2：序列标注。

功能3：命名实体提取。

功能4：训练和测试过程利用matplotlib绘制结果图，并以pdf格式输出评估指标结果。

## 2.2系统概要设计

Project-Master

Function 4

Function 3

Function 2

Function 1

实体提取

序列标注

中文分词

指标评估

结果绘图

（1）中文分词：主要利用jieba+用户自定义词典对医学文本语料进行分词处理。

（2）序列标注：通过代码对分好的词进行实体标注，整理好训练需要的数据集格式。．

（3）实体提取：通过神经网络训练出本次实训使用的医学电子病历专用的model，然后利用测试文本可以对文本内指实体进行识别提取测试。

（4）结果绘制：放入测试数据，执行主方法获得相应的评测指标数据，即准确率，召回率，F值。

## 2.3系统详细设计

### 2.3.1数据集预处理

采用的编码语言是python3，操作系统是ubuntu16.0 64bit

以下是所需要的代码设计：

pos.py进行数据的预处理：

主要类和方法：

class CRF\_unit

def ReadFileUTF8(file)#读取文件

def get\_posTag(self, sentence)#分词

def get\_token(self, filename):#实体标注

def read\_type(self, itype):#判断5种实体特征标签，用此方法返回标签对应的英文名

def get\_type(self, filename)#获取实体特征，并把读到的特征标签添加到features列表中，更新features列表

#add\_method.py

def AddTrain(features, file, linetag="\n")#把特征标签写到训练文件里

### 2.3.2测试结果检查

NeuroNER/src/main.py文件进行测试结果的检查，并得到测评指标数据：

(1)true:标记正确的个数

(2)All:所有进行了标记的个数

(3)ＯｔｏＢＩ：将Ｏ错误标记称Ｂ或Ｉ的个数

(4)ＢＩtoO:将Ｂ或I错误标记为O的个数．

(5)Ｐ:准确率＝true/(true+OtoBI)

(6)Ｒ:召回率=true/(true+BItoO)

(7)F1:F值=(P\*R\*2)/(P+R)

### 2.3.3应用

FortmatChange.py用于将文本转换为conl2003的格式,然后把测试文本放在/data/conll2003/en/test.txt下，之后执行main.py,通过预先训练好的model可以对测试文本进行实体特征的识别提取，并通过过brat工具可以快速标记出实体。

# 第3章 系统开发

## 3.1系统开发环境配置

开发语言：python3

开发环境配置说明文件：

./code[/Instructions-Anaconda+CUDA+CUDNN+CRF+pycharm.docx](file:///E:\腾讯QQ\1030626864\FileRecv\安装教程.docx)

1.anaconda安装

2. pypi 镜像使用帮助

3. anaconda的配置与激活

4.安装Cuda

5.安装CUDNN

6.安装tensorflow

7.安装CRF++

8.安装pycharm

## 3.2 系统开发

### 3.2.1开发详细过程

用jieba分词对有些词不准，需要自己构建词典对已标注好的原始文本处理，只取第一列实体名提取txt文件某一列数据，并保存为txt文件。

代码实现：

#注：001.txt 和002.txt均为测试代码文件，稍后将以此方法对600个标注好的原始文本进行取词，然后制作自己的词典，进而使用jieba并引入专业领域词典（即自己准备的词典）分词，这能使分词更加准确。

import sys

path = "./001.txt" #数据来源

f = open(path , 'r')

line = f.readline()

list = []

while line:

a = line.split()

# print(a[0])

b = a[0]

list.append(b)

line = f.readline()

f.close

list.reverse()#反向列表中元素

#print(list)

f2 = open('002.txt', 'w') #提取后的数据文件

count = len(list)

while count > 0:

f2.write(list[count - 1]+"\n")

count = count - 1

f2.close()

现在需要合拼所有文本，以方便取词时读取文本

Linux下用cat命令进行600.txt文件合拼工作

cat ./origin-data2/\*.txt > ./cat2.txt

合拼后结果保存在了 cat2.txt文件了，格式依然是原文本，如图3-1。

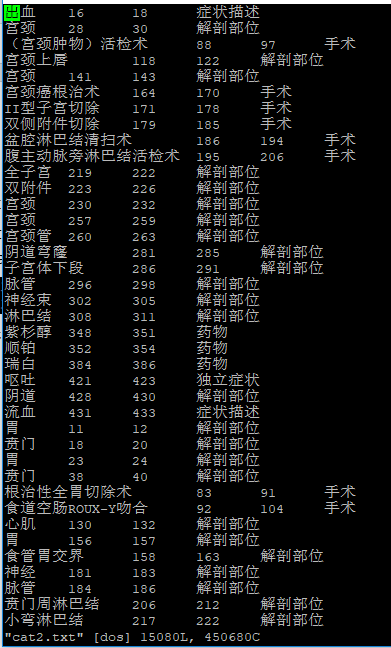


图3-1

现在开始取词，并将取好的词保存在dict.txt文件中，作为jieba词典使用。实现代码如图3-2。

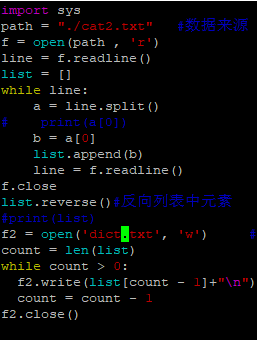


图3-2

python quci.py

结果：

打开dict.txt，如图3-3。

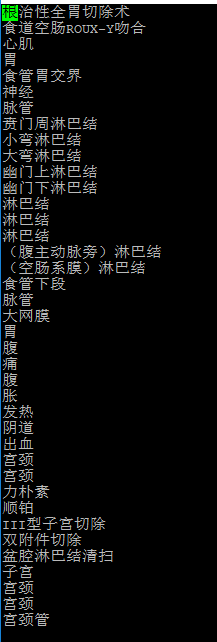


图3-3

开始结合自己的词典对原始文本进行处理

之前我已经用cat命令将原始文本进行了合拼，合拼后的结果保存在了/zhangtao/cat.txt里，即：如图3-4。



图3-4

接下来开始利用jieba+词典分词：

实现的代码我放在了./code/src/jieba/jieba\_seg.py

import jieba

jieba.load\_userdict("../dict.txt")

file\_input = open("../cat.txt","r").read()

file\_input = ' '.join(jieba.cut(file\_input))

file\_output = open("./origin-data\_seg.txt","w")

file\_output.write(file\_input)

分词后结果与词典对比：如图3-5。



图3-5

用自己的model进行分词测试：

在/deepnlp-master/test进行测试

新建一个bimedical.py文件，如图3-6。

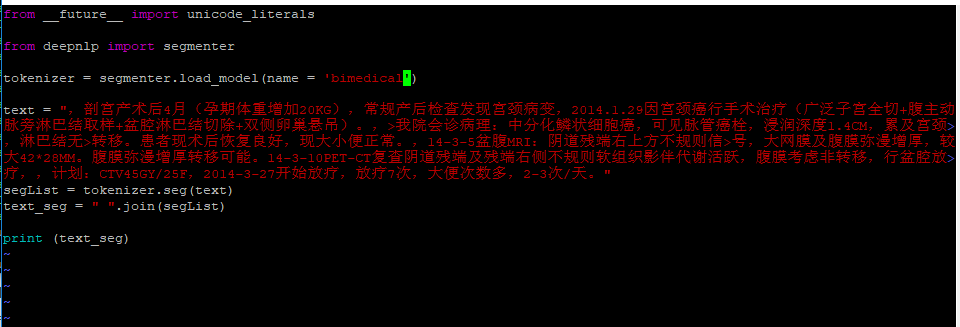


图3-6

Error：Please use deepnlp.register\_model('segment', 'bimedical')

此时在

/home/zutnlp/anaconda3/lib/python3.6/site-packages/deepnlp/model\_util.py注册一下自己的模板，如图3-7。

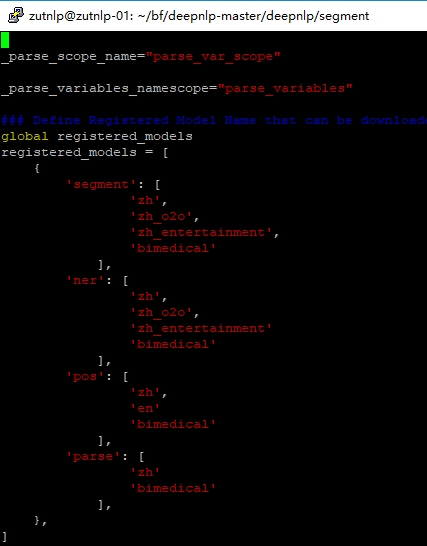


图3-7

同时需要将自己的模板copy到python36环境对应的model下，如图3-8。



图3-8

用自己的model测试分词的结果：如图3-9。

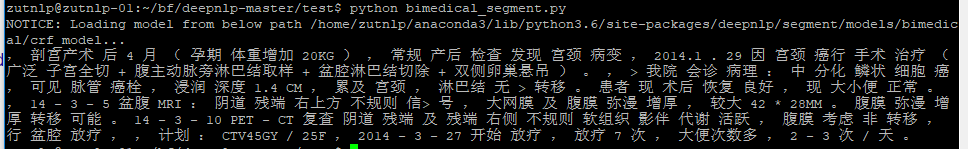


图3-9

分完词之后开始进行词性标注即POS阶段：(在词性标注好之后，才可以train pos model)注：这里的词性标注指的是文本特征标注。

如何对已经分词的文本进行词性标注呢（注意：这里做的是电子病历的命名实体识别，所以需要做的是把被识别的实体在训练文本中用5种特征词标注出来，特征词：解剖部位AnatomicSite、症状描述SymptomsDescribed、独立症状IndependentSymptoms、药品Medicine、手术Operation）

训练文件标注的结果：如图3-10。

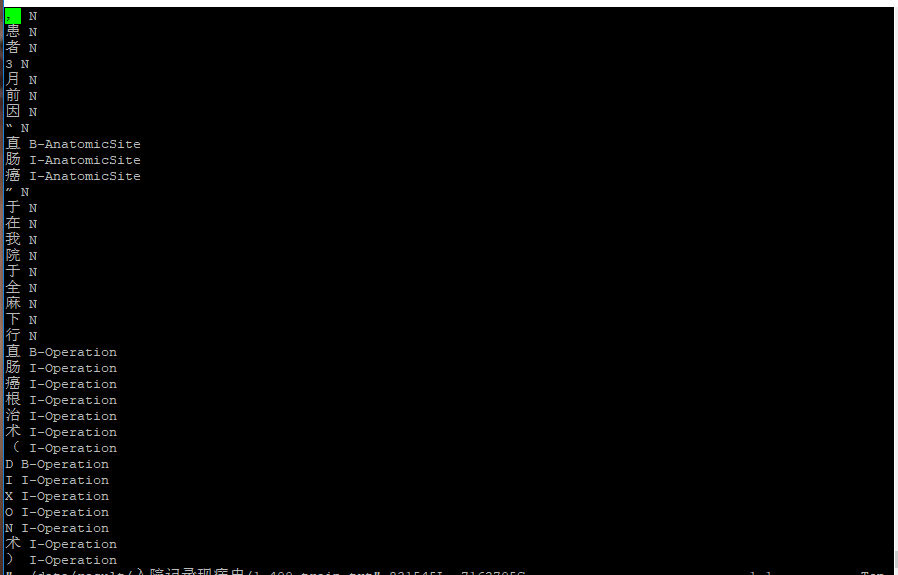


图3-10

接下来训练pos\_Tag model

需要准备的是：

exec.sh（执行脚本）//自己编写，这里是分步执行的，后面在写个执行脚本  
template（特征模板）//crf自带  
test.data（测试集）//这里准备了200个测试文件，和400个训练文件在同一目录里  
train.data（训练集)//这里的训练集是1-400\_train.txt(已经打标签了) 如图3-11-a和图3-11-b。

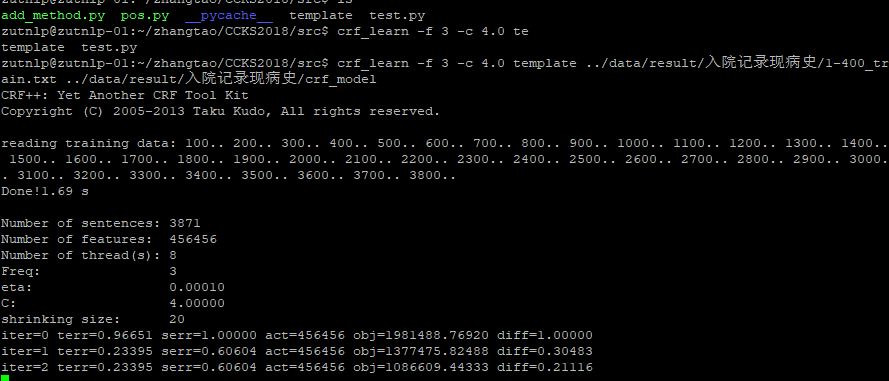


图3-11-a

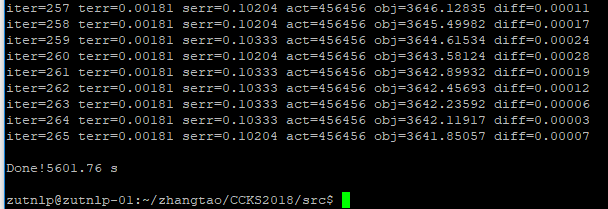


图3-11-b

到此400个文件的model已训练完毕。

写一个test脚本，用于200个测试数据处理：如图3-12。

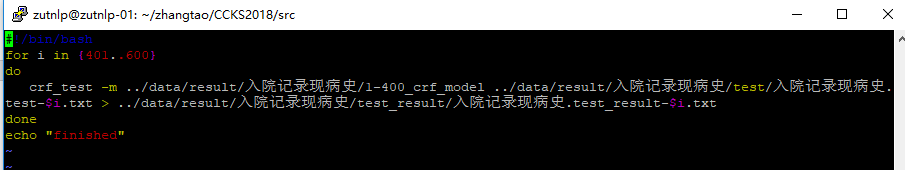


图3-12

执行脚本之后

zutnlp@zutnlp-01:~/zhangtao/CCKS2018/src$ bash ./test\_file.sh

这是测试数据实体特征识别出来的结果：

目录在：~/zhangtao/CCKS2018/data/result/入院记录现病史/test\_result，如图3-13。

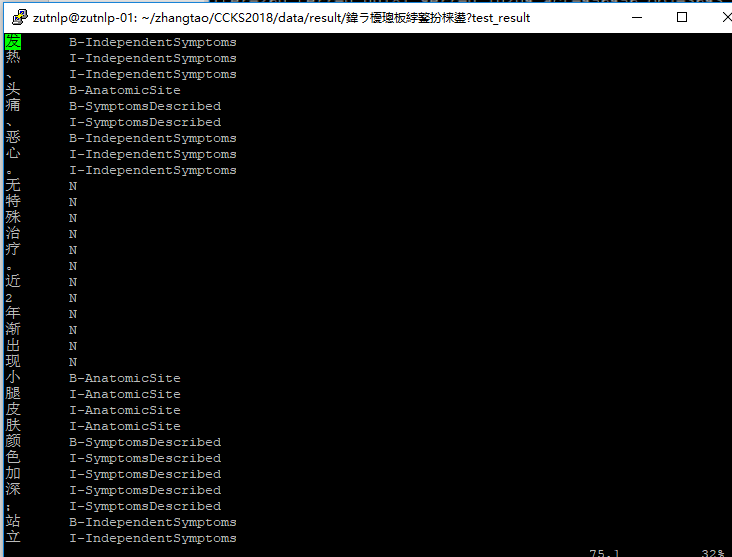


图3-13

从数据看，实体识别的结果还算理想。接下来要进行利用工具去评估几个参数：

采用精确率（Precision）、召回率（Recall）以及F1-Measure作为评测指标。

首先数据分类：原始数据里600个，前400个作为train数据，后200个作为valid数据，在/zhangtao/CCKS2018/data/origin-data\_backup里

后来给的400个未标注的文本数据是作为test数据的，在/zhangtao/CCKS2018/data/origin-data\_backup/test1-400

现在对代码进行修改，使其对数据进行实体标注

200 个valid文件个格式已转换，如图3-14-a和3-14-b。

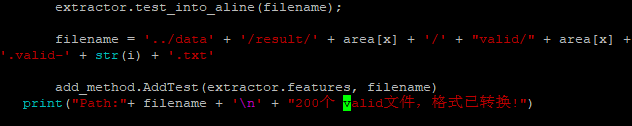


图3-14-a

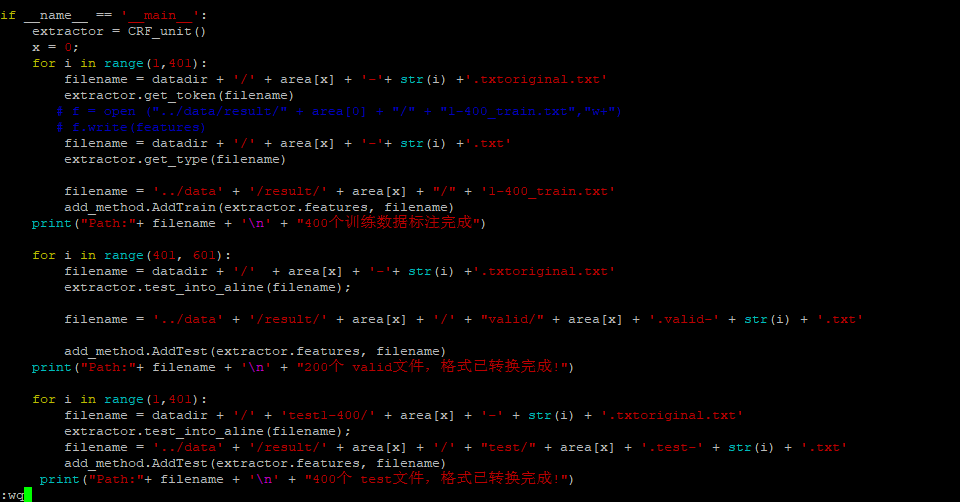


图3-14-b

python pos.py 后： 如图3-15。

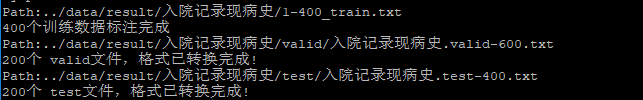


图3-15

之后执行标注脚本Tag.sh得到的valid和test文件加上实体标注：

修改Tag.sh代码：如图3-16-a和3-16-b。



图3-16-a

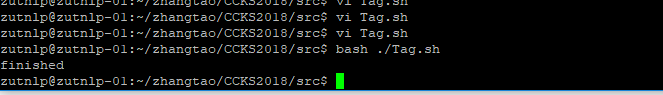


图3-16-b

看一下标注的结果：如图3-17。

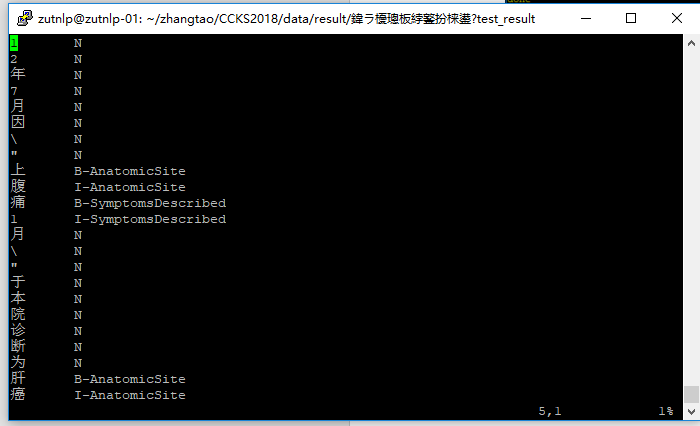


图3-17

之后把原始数据集，和标注好的数据集分别拷贝到NeuroNER工程对应的数据文件夹下，在这里写了两个文件cp脚本，用于拷贝文件，如图3-18-a。

cp.sh:



图3-18-a

bash 一下：如图3-18-b。

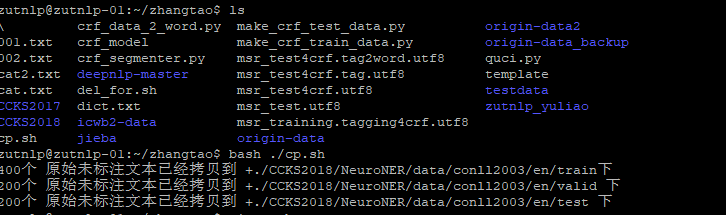


图3-18-b

cp2.sh:如图3-18-c。

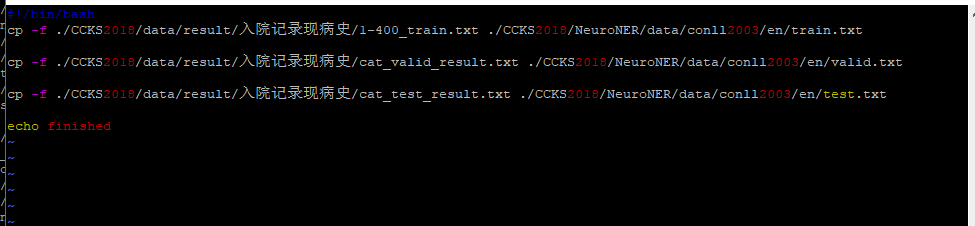


图3-18-c

bash一下:如图3-18-d。



图3-18-d

需要的文件都已经copy过来了，如图3-18-e。

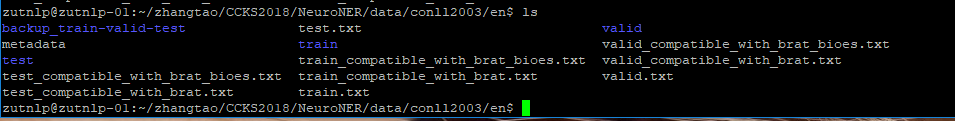


图3-18-e

准备的数据格式是两列，而要求格式是4列，所以报出了参数错误，无法兼容什么的，如图3-18-f。

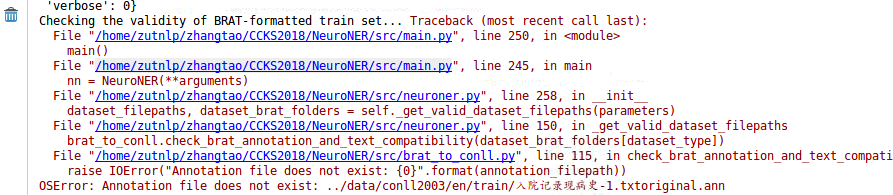


图3-18-f

现在要做的处理就是对 train.txt valid.txt test.txt 三个数据格式修改 ，随意添加两列，使其格式符合该工程，需要写个py脚本用于格式转换：如图3-19。



图3-19

在格式转换的过程中遇到list越界，经查明是文本里有多余空行，因此写了一个删除文本多余空行的方法：delSpaceLine.py,如图3-20。

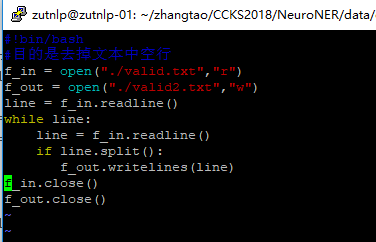


图3-20

以此类推把test文本做了删除空行的处理，这样上边的格式转换就不会出错。

在rnu main.py 是报错：如图3-21。

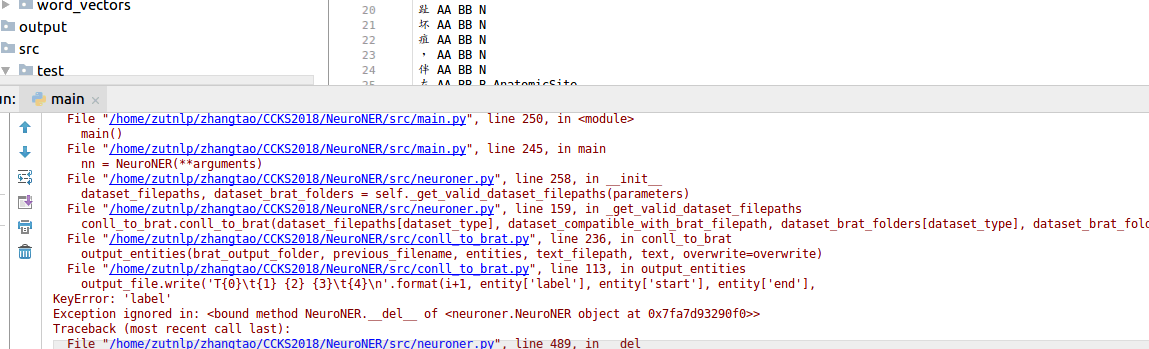


图3-21

格式问题：需要重新调整格式

问题原因：<https://github.com/Franck-Dernoncourt/NeuroNER/issues/62>

修改代码：把N替换掉成O，如图3-22。

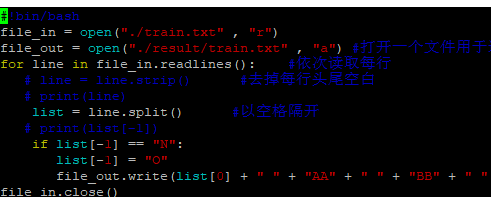


图3-22

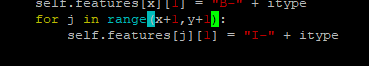
下面工作：

1. 调整src/pos.py 代码，生成需要的400个标注数据 ，同时也完成了200个valid数据和400个test数据格式的重整。结果放在了 data/result/入院记录现病史里
2. 拿这400个标注数据去训练 model
3. 执行Tag.sh去 标记 200个valid数据 ，和 400个test数据（其实在这里我疑惑一个问题，我之所以说拿这个model去标注这些数据，一是方便，直接用crf\_test一下就好了，二是我觉得用了这个model只是对实体识别特征的修改，没有其他影响。可是问题是，这样做的的话，数据集之间存是否在着交叉呢，用了train set训练出来的model去标记valid set 和test set，这样做需要的几个指标（精确率（Precision）、召回率（Recall）以及F1-Measure作为评测指标）现在有两个方案，即准备两套数据集，一套是用model标记的，另一套非model标记（即由准备的jieba词典+jieba分词+标注代码完成。）

方案一：

调整src/pos.py 代码，生成需要的400个标注数据 ，同时也完成了200个valid数据和400个test数据格式的重整。结果放在了 data/result/入院记录现病史里

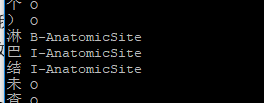
修改的代码块：如图3-23-a。



3-23-a

把y+1修改成 y

Run一下之后的结果：如图3-23-b



3-23-b

不过数据中有一点小小的错误，就是标注中会产生一个None的标签，在生成前面的数据时也遇到过，经过我仔细的查看推敲和对数据的检查，发现在识别药物特征出的错，故回到读取代码一看果真是打错了一个字，把“药物“打成了”药品“，如图3-23-c。



图3-23-c

修改过来后：如图3-23-d。



图3-23-d

然后执行代码，重新生成数据，如图3-23-e。

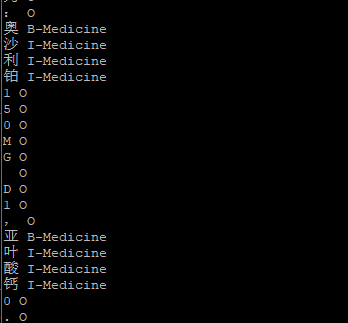


图3-23-e

拿这400个标注数据去训练 model。

执行Tag.sh去 标记 200个valid数据 ，和 400个test数据。

bash ./Tag.sh

得到结果保存在了 /zhangtao/CCKS2018/data/result/入院记录现病史/test\_result 和 /zhangtao/CCKS2018/data/result/入院记录现病史/valid\_result，如图3-24。

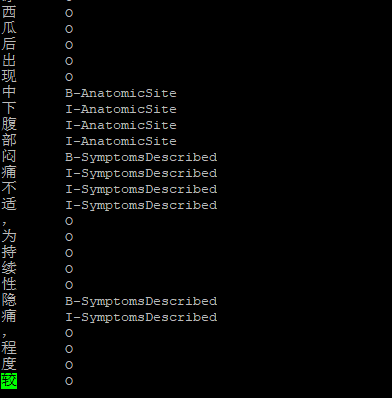


图3-24

接下来继续处理格式，处理成4列，以便在NeuroNER工程里参数格式能兼容处理过程就不在这里详述，大概处理过程会说一下。

先用cat把数据集合拼：

cat ./test\_result/\*.txt > ./cat\_test\_result.txt

cat ./valid\_result/\*.txt > ./cat\_valid\_result.txt

在执行FormatChange.py时会出错，这是因为用cat合拼之后，每个文件内容间会有个空白格，当读取到那里，自然就读不到指，会报越界错误，所以写了个delSpaceLine.py脚本，用于删去文件里的多余空行。切记在pyhton代码块里不允许用Tab补充空格，python严格按空格缩进的，因为用model得到的valid set和test set 其实我查出了它们列间是Tab填充的，所以在用list[1]牵引时会出错。在这里我没有返回到pos.py里去寻找原因，而是用list[-1]去从右侧做牵引，结果可以得到想要的数据格式。

记得修改FormatChange.py里面对应的文件名

现在查看一下得到的数据：如图3-25。

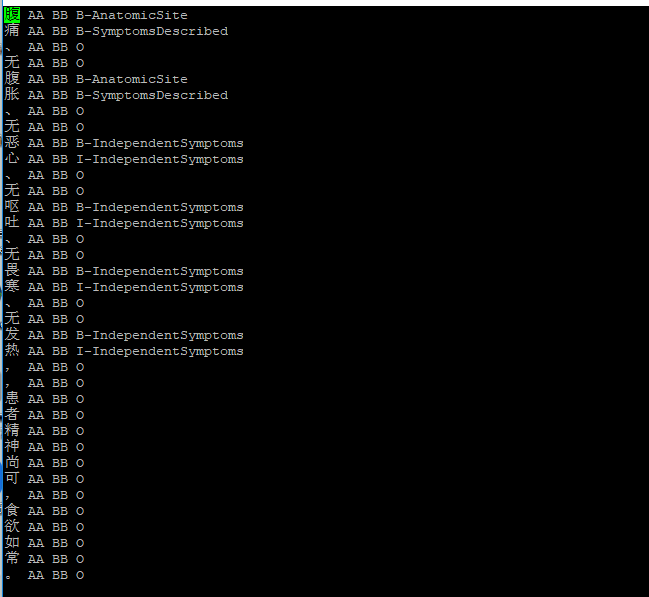


图3-25

接下来把重整好的data set 拷贝到NenroNER对应的目录下，然后开始run一下代码，测试得到的数据吧！之后就是静静地等待测试结果！但愿这次代码不要在报致命的错了。如图3-26。

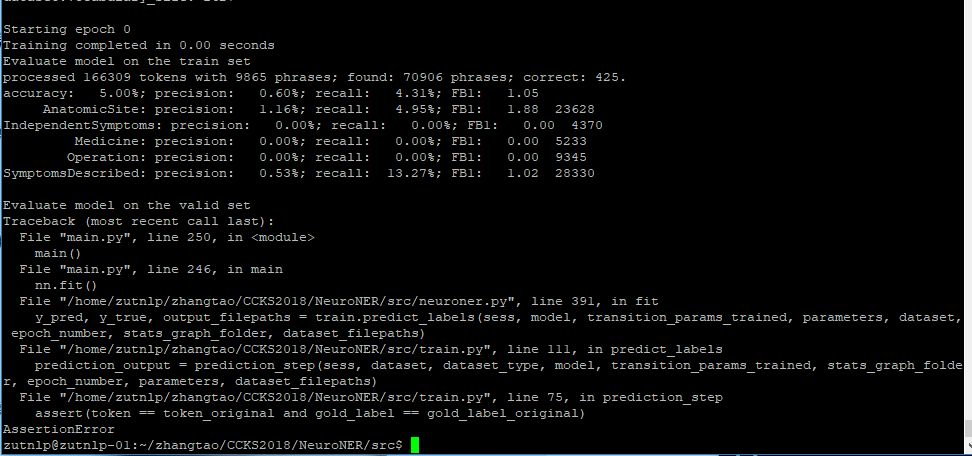


图3-26

还是弹出一个错误。

开始解决，经过推断分析，可能是准备的数据集有问题，因为valid set 和 test set 全是用train set 的model得到的，所以我怀疑这里处理不是很得当。

因此现在采用方案二：重整数据集，其中不在使用model进行训练标注了，直接用代码标注。

调整一下数据集： 500个 train set 100个valid set 400个test set

由于后来的400个测试文本未给对应的标记文本，所以这400个test数据依然需要model 来标注

同样方法来训练model，之后利用model 标记test2里的数据集

之后把Tag.sh 里的文件路径改一下：如图3-27。

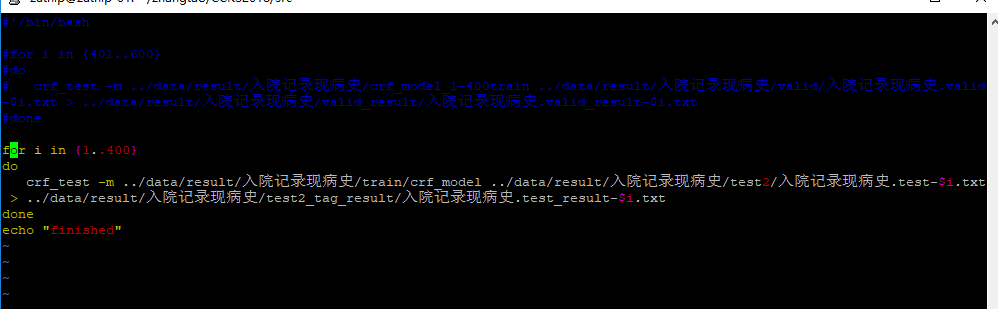


图3-27

bash 一下就完成了。

标注结果：如图3-28。

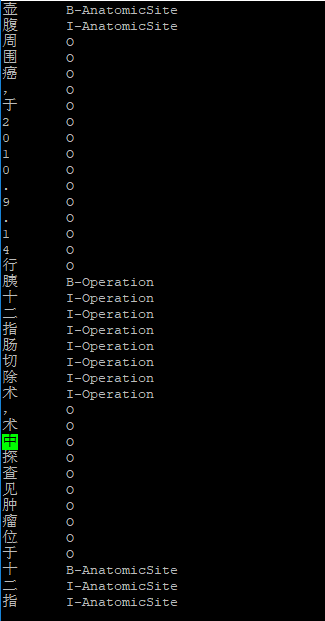


图3-28

就是有一点问题是，两列之间时tab填充的。所以后面需要对数据格式重整，只能用list[-1]牵引，前面已经讲过了。

合并文件，在去除空行，最后格式转换，转换好的数据都存在了 ultimate\_data下了。

有一个错误，因为文件名创建错了。

接下来，把数据集再次拷贝到NeuroNER对应的目录里，然后跑一下代码。

同样又得到了一个错误，不过知道问题的原因了，就是不能用model去标记test数据集，所有数据集都需要是代码标注的。应该是这样。

现在意识到一个错误，我想所谓的对数据集分类，应该是这样的，先从官方那里得到是600个原始文本+600个对应的标注文本 实质是对这600个文本进行分类，即：400个train +50个valid +150个test（test有点少）不过可以这样测试NeuroNER代码，先跑起来，看看结果，由这个NeuroNER训练出来的model再去对后来的400个test数据进行测试！把数据集重整一下！方法和上面一样，就不再累述了！如图3-29-a。

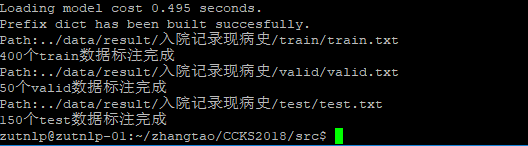


图3-29-a

格式修改：如图3-29-b。

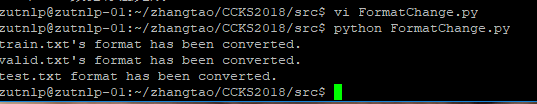


图3-19-b

预览一个：如图3-29-c。

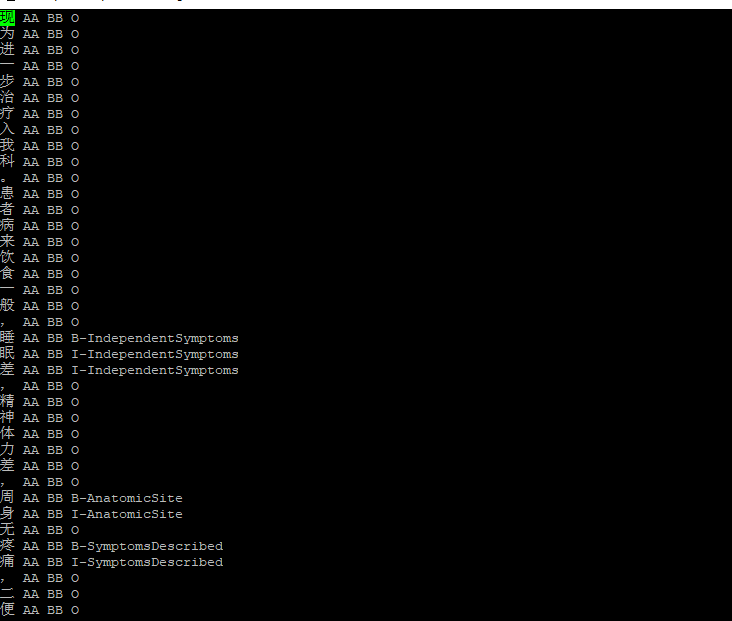


图3-29-c

接下来把数据集拷贝到NeuroNER对应的目录里，跑代码，看结果。

又弹出了一个问题：如图3-30。

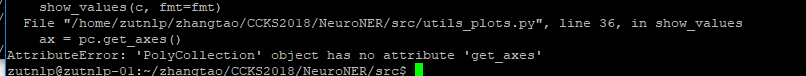


图3-30

这个问题遇见过，可以解决，就是方法名变了，改一下就好。

AttributeError: 'PolyCollection' object has no attribute 'get\_axes'

解决方案：change 'ax = pc.get\_axes()' to 'ax = pc.axes' will work !

改完之后再次运行，但愿代码能够跑起来。如图3-31。

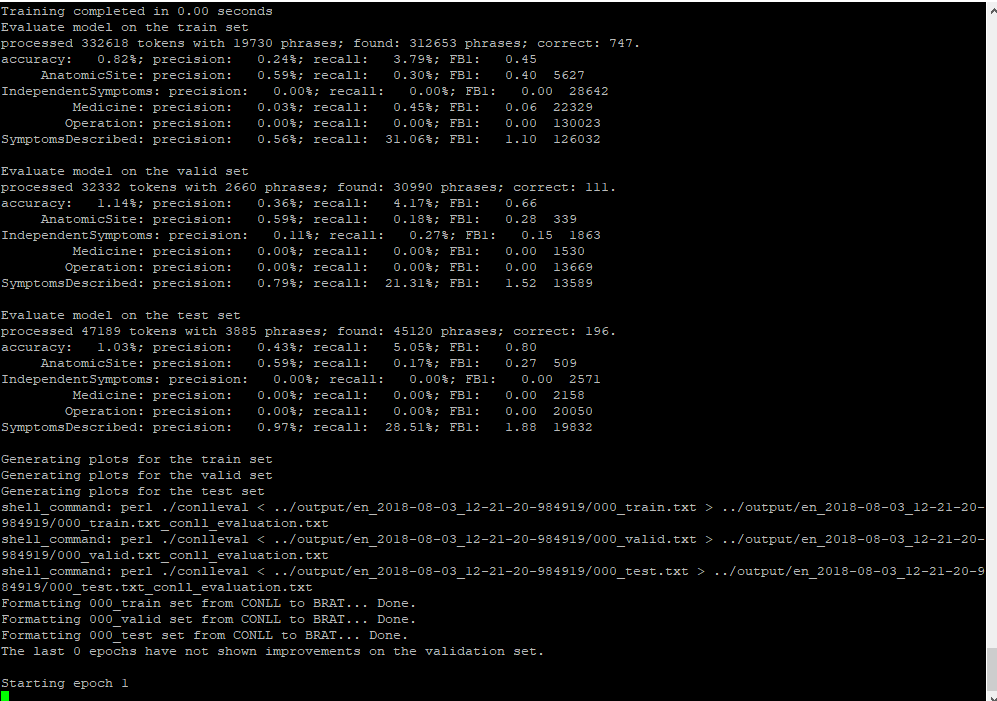


图3-31

经过了几个小时的漫长训练，验证，测试，和评估，100轮后实验结束，这是实验结果。如图3-32。

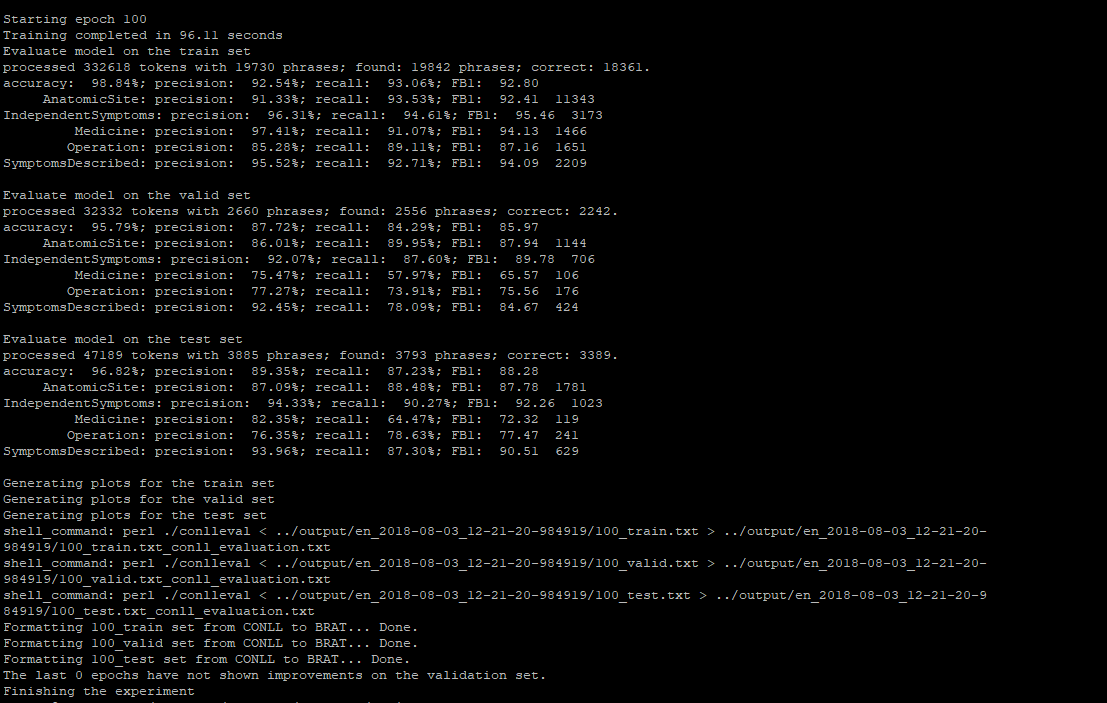


图3-32

### 3.2.2遇到的问题及解决方案

错误1：

当执行 python crf\_segmenter.py crf\_model ./icwb2-data/testing/msr\_test.utf8 msr\_test.seg.utf8 后报错：如图3-33。

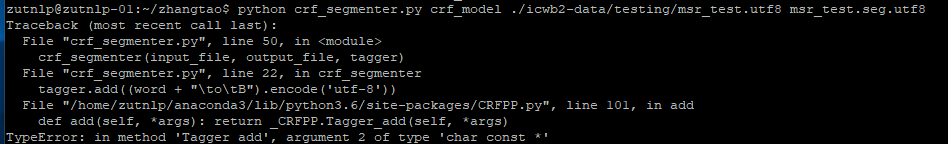


图3-33

How to fix it ?

原因是： 因为 2和3的str类型不同，原则上用python2可以运行，但由于py2里没有安装CRF++,所以这里我修改了一下代码：

在crf\_segmenter.py里去掉encode(‘utf-8’)，此时python3里可以运行。如图3-34。

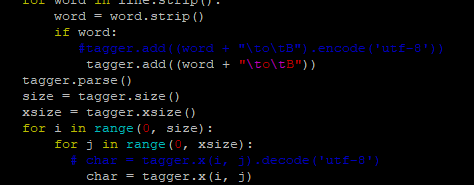


图3-34

在调试NeuroNER代码是遇到的问题及解决方案：

1.当 python3.5 main.py 后报错：

Trouble running main.py missing module attribute-distutils-util #75

解决方案：

It seems something has changed in distutils. Try changing  
import distutils to import distutils.util in neuroner.py,（即将import distutils 换成 import distutils.util）

相关链接：<https://github.com/Franck-Dernoncourt/NeuroNER/issues/75>

错误2：

AttributeError: 'PolyCollection' object has no attribute 'get\_axes'

解决方案：change 'ax = pc.get\_axes()' to 'ax = pc.axes' will work !

相关链接：<https://github.com/Franck-Dernoncourt/NeuroNER/issues/76>

# 第4章 系统测试

## 4.1测试环境

1. 下载anaconda

清华大学镜像站：<https://mirrors.tuna.tsinghua.edu.cn/anaconda/archive/>



下载Anaconda3-5.1.0-Linux-x86\_64.sh

2. 安装anaconda

打开终端进入到下载好的文件的目录下（或找到下载文件，在其文件夹内打开终端）

首先执行sudo chmod +x Anaconda3-5.1.0-Linux-x86\_64.sh



然后执行./Anaconda3-5.1.0-Linux-x86\_64.sh



根据提示默认输入回车和“yes”

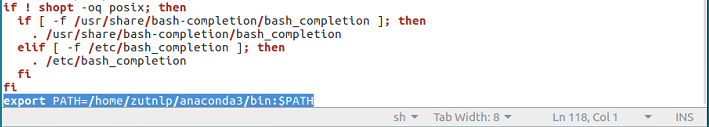
3. anaconda环境配置

执行sudo gedit ~/.bashrc进行环境配置



在末尾添加

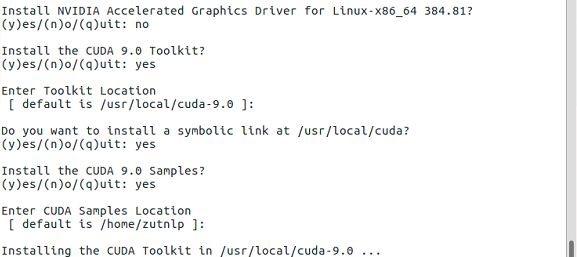
export PATH=/home/用户名/anaconda3/bin:$PATH



完成后许需重启计算机或注销方能生效

4.安装cuda

首先在以下好的文件的文件夹内，执行sudo sh cuda\_9.0.176\_384.81\_linux.run



5. 配置环境变量

执行sudo gedit  /etc/profile打开配置文件

在末尾添加

export  PATH=/usr/local/cuda-9.0/bin:$PATH

export  LD\_LIBRARY\_PATH=/usr/local/cuda-9.0/lib64$LD\_LIBRARY\_PATH

6.安装cudnn

进入到Downloads，也就是包含cuDNN Tar文件的目录。

用以下命令解压缩cuDNN软件包

tar -xzvf cudnn-9.0-linux-x64-v7.tgz

将以下文件复制到CUDA Toolkit目录中。

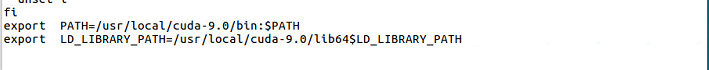
sudo cp cuda/include/cudnn.h /usr/local/cuda/include

sudo cp cuda/lib64/libcudnn\* /usr/local/cuda/lib64

授予权限

sudo chmod a+r /usr/local/cuda/include/cudnn.h

sudo chmod a+r /usr/local/cuda/lib64/libcudnn\*



保存退出即可。

重启或注销系统即可生效

7.安装tensorflow

在python36中执行pip install tensorflow

验证时进入python，import tensorflow即可

8.安装crf++

解压

tar -zxvf CRF++-0.58.tar.gz



进入CRF++0.58目录下

cd CRF++-0.58

./configure



make

sudo make install

进入子目录python中

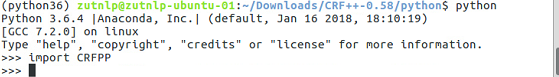
python setup.py build

python setup.py install

进入python执行import CRFPP

（若出现ImportError: libcrfpp.so.0: cannot open shared object file: No such file or directory

执行ln -s /usr/local/lib/libcrfpp.so.0 /usr/lib/）



## 4.2功能测试

功能一：中文分词

之前我已经用cat命令将原始文本进行了合拼，合拼后的结果保存在了/zhangtao/cat.txt里，即：如图4-1。



图4-1

接下来开始利用jieba+词典分词：（注：词典如何做的在3.2.1开发详细过程里有）

实现的代码我放在了/zhangtao/jieba/jieba\_seg.py

import jieba

jieba.load\_userdict("../dict.txt")

file\_input = open("../cat.txt","r").read()

file\_input = ' '.join(jieba.cut(file\_input))

file\_output = open("./origin-data\_seg.txt","w")

file\_output.write(file\_input)

功能二：序列标注

python pos.py ，如图4-2

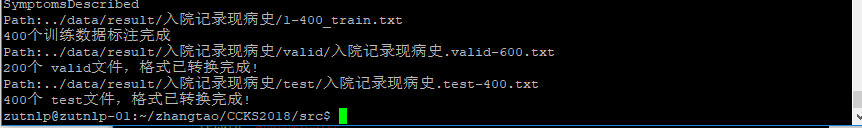


图4-2

功能三：实体提取

写一个test脚本，用于200个测试数据处理：如图4-3。

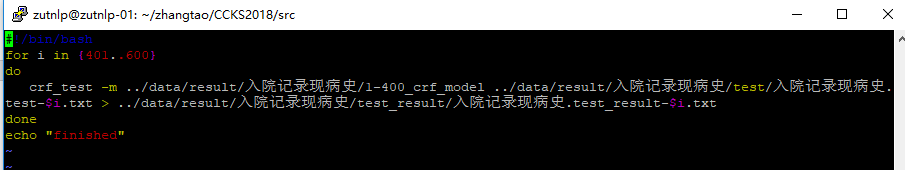


图4-3

执行脚本之后

zutnlp@zutnlp-01:~/zhangtao/CCKS2018/src$ bash ./test\_file.sh

这是测试数据实体特征识别出来的结果：

目录在：~/zhangtao/CCKS2018/data/result/入院记录现病史/test\_result

同时利用ANN提取出来的结果保存在了

~/zhangtao/CCKS2018/NeuroNER/data/conll2003/en/test

功能四：指标评估，结果绘图

把重整好的data set 拷贝到NenroNER对应的目录下，然后开始run一下代码，测试准备好的数据吧！之后就是静静地等待测试结果！

## 4.3测试结果分析

中文分词结果：如图4-4。



图4-4

分词后结果与词典对比，还算理想。

序列标注：

这是400个train文件标注的结果：如图4-5。



图4-5

标注的结果还算理想，把想要的实体给打上了特征标签。

实体提取：如图4-6。

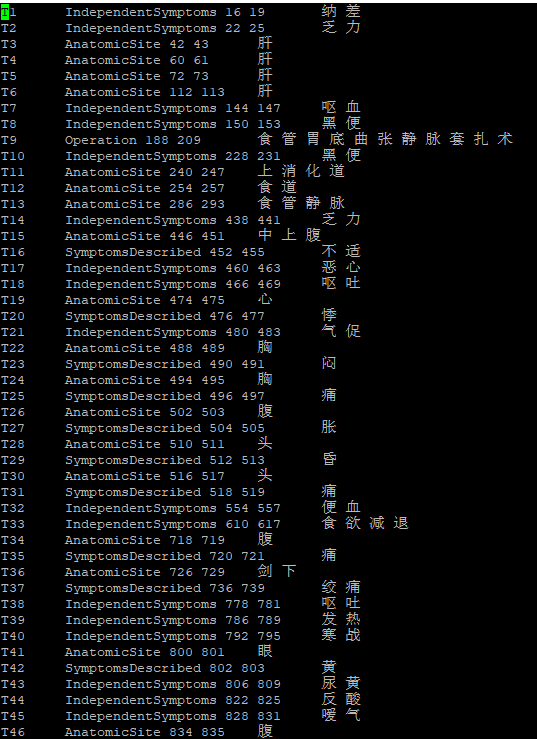


图4-6

从数据看，实体识别的结果还算理想。接下来要进行利用工具去评估几个参数：

采用精确率（Precision）、召回率（Recall）以及F1-Measure作为评测指标

指标评估，结果绘图：如图4-7。

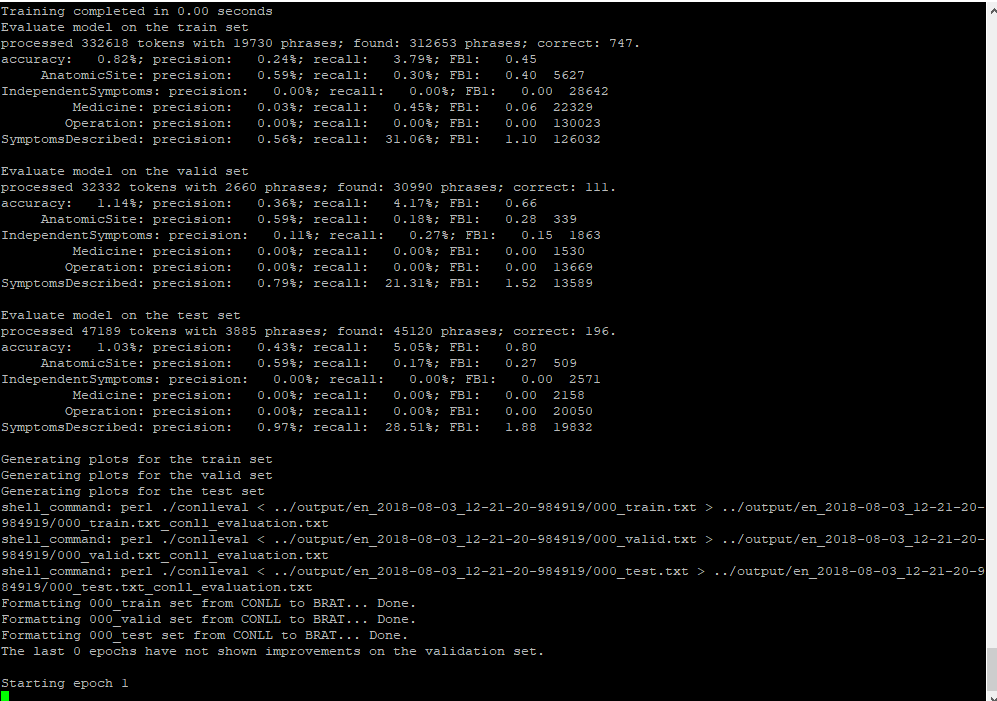


图4-7

经过了几个小时的漫长训练，验证，测试，和评估，100轮后实验结束，这是实验结果。

以下是以图表格式输出的测试结果：如图4-8。

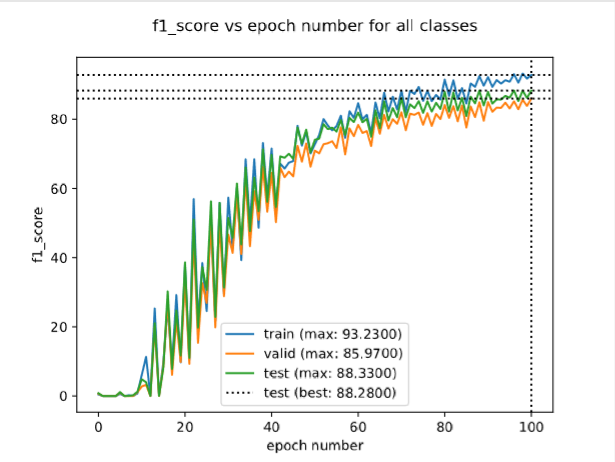


图4-8

通过测试的指标结果看，本次测试的最优F1的值是：88.33，而业界做的较好的F1值一般都是在90以上，这也意味着本次实验的指标结果还有待提高！也许和的测试集的数量有关，也许跟使用的方法有关。所以今后我要仔细思考和专研一下，争取能有更好的测试结果。

# 第5章 结 论

# 总结

主要结论如下:

(1)在生物医学命名实体识别任务中,人工特征和领域知识对于结果的影响很大。但是构建合适的人工特征需要大量的特征选择实验,导致了系统的成本提升,泛化能力下降。因而本文构建了CNN - BILSTM-CRF深层神经网络模型,在不使用任何人工特征的情况下,获得了比使用大量丰富特征和领域知识的浅层机器学习方法更好的结果。

(2)本文提出了利用ANN网络来获得表示单词形态特征的字符向量,用以补充词向量的不足。通过字符向量的加入,使得模型对于含有特殊字符,大小写混合这类实体能够更有效地识别,从而提高了模型的性能。

(3)为了获得更加准确的识别结果,我们通过CRF对CNN-BILSTM网络的输出进行解码,获得最优的标记序列。CRF的融人提升了对于含有多修饰词,边界模糊的生物医学实体的识别性能。

综上,在生物医学命名实体识别任务上,本文提出的通过CNN网络获得字符级特征来补充词向量,以及BILSTM与CRF模型的融合,都是有效提高识别性能的途径。

在这篇文章中，介绍了NeuroNER，一种基于CNN-BILSTM的NER工具，非专业用户也可以利用它去实验一些强大的文本标注与识别提取功能。

# 参考文献

[1] 吴军. 数学之美. 北京：人民邮电出版社，2014.11//书

[2] 罗刚. 自然语言处理原理和技术实现. 北京：电子工业出版社，2016.5//书

[3] 杨锦峰．电子病例命名实体识别和实体关系抽取研究综述．自动化学报，2014.8//论文

[4] 苏娅．在线医疗文本中的实体识别研究．　北京大学大学报（自然科学版），　2016.20//论文

[5] 郑捷．机器学习算法原理与编程实践．　北京：电子工业出版社　2015.11//书

[6] 宗成庆. 统计自然语言处理．　北京：清华大学出版社　2008//书

[7] MIT.aneasy-to-use programfor named-entityrecognitionbasedonneuralnetworks

[8] Jinhang Wu, Xiao Hu, Rongsheng Zhao, Feiliang Ren\*, Minghan Hu.Clinical Named Entity Recognition via Bi-directional LSTM-CRF Model

# 致 谢

这次实训做的是自然语言方面的应用实现，这是人工智能领域的一个重要分支，同时理论和技术难度系数也很大。

虽然实验过程有些艰难，遇到了好多困难，不过最终在老师和同学们的指导和帮助下很多问题得到了解决。在这个过程很感谢我的指导老师刘小明，细心的指导我学习，他真的是很负责任的一位老师，特别感谢！同时也感谢这次实训的主指导老师张俊宝老师，感谢他能给我二次验收的机会，在重做的过程中真的收获到了好多，学会了如何学习新领域新知识，如何解决工程实现中遇到的各种问题。

总之一句话，搞科研就是不断的发现问题，解决问题，创造出新的问题，逐步的提高自己的科研认知和水平。

# **附 录**

## 附录: 详细的代码清单

#pos.py 处理数据集的

# coding:utf-8

import add\_method

import codecs

import sys

import os

import jieba

import jieba.posseg as pseg

jieba.load\_userdict("../data/jieba\_dict/dict.txt")

datadir = "../data/origin-data\_backup"

area = ["入院记录现病史"]

class CRF\_unit:

def \_\_init\_\_(self):

self.features = []

def test\_into\_aline(self, filename):

self.features = []

sentences = add\_method.ReadFileUTF8(filename);

for sentence in sentences:

for token in sentence:

self.features.append(token)

def get\_posTag(self, sentence):

# words = ' '.join(jieba.cut(sentence))

words = pseg.cut(sentence)

return words

def get\_token(self, filename):

# f = open ("../data/result/" + area[0] + "/" + "1-400\_train.txt","w+")

self.features = []

sentences = add\_method.ReadFileUTF8(filename);

for sentence in sentences:

words = self.get\_posTag(sentence)

for w in words:

for token in w.word:

#print(token)

feature = [token , "O"]

# print(feature)

self.features.append(feature)

def read\_type(self, itype):

# itype = itype.encode('utf-8')

if itype == "解剖部位":

return "AnatomicSite"

if itype == "症状描述":

return "SymptomsDescribed"

if itype == "独立症状":

return "IndependentSymptoms"

if itype == "药物":

return "Medicine"

if itype == "手术":

return "Operation"

def get\_type(self, filename):

sentences = add\_method.ReadFileUTF8(filename);

for sentence in sentences:

words = sentence.split()

# print (words[-3] + words[-2])

x = int(words[-3])

y = int(words[-2])

itype = self.read\_type(words[-1])

itype = str(itype)

# print(itype)

# if itype == "None":

# print(filename)

# break

self.features[x][1] = "B-" + itype

for j in range(x+1,y):

self.features[j][1] = "I-" + itype

if \_\_name\_\_ == '\_\_main\_\_':

extractor = CRF\_unit()

x = 0;

for i in range(1,401):

filename = datadir + '/' + area[x] + '-'+ str(i) +'.txtoriginal.txt'

extractor.get\_token(filename)

# f = open ("../data/result/" + area[0] + "/" + "1-400\_train.txt","w+")

# f.write(features)

filename = datadir + '/' + area[x] + '-'+ str(i) +'.txt'

extractor.get\_type(filename)

filename = '../data' + '/result/' + area[x] + "/train/" + 'train.txt'

add\_method.AddTrain(extractor.features, filename)

print("Path:"+ filename + '\n' + "400个train数据标注完成")

for i in range(401,451):

filename = datadir + '/' + area[x] + '-'+ str(i) +'.txtoriginal.txt'

extractor.get\_token(filename)

filename = datadir + '/' + area[x] + '-'+ str(i) +'.txt'

extractor.get\_type(filename)

filename = '../data' + '/result/' + area[x] + "/valid/" + 'valid.txt'

add\_method.AddTrain(extractor.features, filename)

print("Path:"+ filename + '\n' + "50个valid数据标注完成")

for i in range(451,601):

filename = datadir + '/' + area[x] + '-' + str(i) + '.txtoriginal.txt'

extractor.get\_token(filename)

filename = datadir + '/' + area[x] + '-'+ str(i) +'.txt'

extractor.get\_type(filename)

filename = '../data' + '/result/' + area[x] + "/test/" + 'test.txt'

add\_method.AddTrain(extractor.features, filename)

print("Path:"+ filename + '\n' + "150个test数据标注完成")

"""

for i in range(401, 601):

filename = datadir + '/' + area[x] + '-'+ str(i) +'.txtoriginal.txt'

extractor.test\_into\_aline(filename);

filename = '../data' + '/result/' + area[x] + '/' + "valid/" + area[x] + '.valid-' + str(i) + '.txt'

add\_method.AddTest(extractor.features, filename)

print("Path:"+ filename + '\n' + "200个 valid文件，格式已转换完成!")

for i in range(1,401):

filename = datadir + '/' + 'test1-400/' + area[x] + '-' + str(i) + '.txtoriginal.txt'

extractor.test\_into\_aline(filename);

filename = '../data' + '/result/' + area[x] + '/' + "test/" + area[x] + '.test-' + str(i) + '.txt'

add\_method.AddTest(extractor.features, filename)

print("Path:"+ filename + '\n' + "400个 test文件，格式已转换完成!")

"""

#add\_method.py 相关函数的定义，在pos.py里调用

# coding:utf-8

import sys

import os

import codecs

import imp

#datadir = "./data"

def ReadFile(file):

f = open(file,'r')

lines = f.readlines()

f.close()

return [line.rstrip('\r\n') for line in lines]

def ReadFileUTF8(file):

f = codecs.open(file,'r', 'utf8')

lines = f.readlines()

f.close()

return [line.rstrip() for line in lines]

def SaveFeatures(features, file, linetag="\n"):

imp.reload(sys)

# sys.setdefaultencoding('utf8')

f = open(file, "w")

for [token,tag] in features:

f.write(token + " " + tag)

f.write(linetag)

f.flush()

f.close()

def AddTrain(features, file, linetag="\n"):

imp.reload(sys)

#sys.setdefaultencoding('utf8')

f = open(file, "a")

for [token,tag2] in features:

f.write(token + " " + tag2)

f.write(linetag)

f.flush()

f.close()

def AddTest(features, file, linetag="\n"):

imp.reload(sys)

#sys.setdefaultencoding('utf8')

f = open(file, "w")

for token in features:

f.write(token)

f.write(linetag)

f.flush()

f.close()

"""if \_\_name\_\_ == '\_\_main\_\_':

lines = ReadFileUTF8(datadir+'/病史特点/病史特点-1.txtoriginal.txt');

for line in lines:

print(line)

"""

# delSpaceLine.py 删除文本中多余空行

#!bin/bash

#目的是去掉文本中空行

"""

f\_in = open("../data/result/入院记录现病史/cat\_valid\_result.txt","r")

f\_out = open("../data/result/入院记录现病史/cat\_valid\_result2.txt","w")

line = f\_in.readline()

while line:

line = f\_in.readline()

if line.split():

f\_out.writelines(line)

f\_in.close()

f\_out.close()

"""

f\_in = open("../data/result/入院记录现病史/test/test.txt","r")

f\_out = open("../data/result/入院记录现病史/test/test2.txt","w")

line = f\_in.readline()

while line:

line = f\_in.readline()

if line.split():

f\_out.writelines(line)

f\_in.close()

f\_out.close()

print("Empty lines have been deleted !")

# FormatChange.py 数据集文本格式转换，新增两列后变为四列（符合conll2003的数据集格式）

#!bin/bash

file\_in = open("../data/result/入院记录现病史/train/train.txt" , "r")

file\_out = open("../data/result/入院记录现病史/ultimate\_data/train.txt" , "a") #打开一个文件用于追加

for line in file\_in.readlines(): #依次读取每行

# line = line.strip() #去掉每行头尾空白

# print(line)

list = line.split() #以空格隔开

# print(list[-1])

file\_out.write(list[0] + " " + "AA" + " " + "BB" + " " + list[-1] + "\n")

file\_in.close()

file\_out.close()

print("train.txt's format has been converted. ")

file\_in = open("../data/result/入院记录现病史/valid/valid.txt" , "r")

file\_out = open("../data/result/入院记录现病史/ultimate\_data/valid.txt" , "a") #打开一个文件用于追加

for line in file\_in.readlines(): #依次读取每行

# line = line.strip() #去掉每行头尾空白

list2 = line.split() #以空格隔开

# print(list[1])

file\_out.write(list2[0] + " " + "AA" + " " + "BB" + " " + list2[-1] + "\n")

file\_in.close()

file\_out.close()

print("valid.txt's format has been converted.")

file\_in = open("../data/result/入院记录现病史/test/test.txt" , "r")

file\_out = open("../data/result/入院记录现病史/ultimate\_data//test.txt" , "a") #打开一个文件用于追加

for line in file\_in.readlines(): #依次读取每行

# line = line.strip() #去掉每行头尾空白

list3 = line.split() #以空格隔开

file\_out.write(list3[0] + " " + "AA" + " " + "BB" + " " + list3[-1] + "\n")

file\_in.close()

file\_out.close()

print("test.txt format has been converted.")

#Tag.sh 这是利用crf工具+训练的model去标注文本

#!/bin/bash

#for i in {401..600}

#do

# crf\_test -m ../data/result/入院记录现病史/crf\_model\_1-400train ../data/result/入院记录现病史/valid/入院记录现病史.valid-$i.txt > ../data/result/入院记录现病史/valid\_result/入院记录现病史.valid\_result-$i.txt

#done

for i in {1..400}

do

crf\_test -m ../data/result/入院记录现病史/train/crf\_model ../data/result/入院记录现病史/test2/入院记录现病史.test-$i.txt > ../data/result/入院记录现病史/test2\_tag\_result/入院记录现病史.test\_result-$i.txt

done

echo "finished"

#template 特征模板

#Unigram

U00:%x[-2,0]

U01:%x[-1,0]

U02:%x[0,0]

U03:%x[1,0]

U04:%x[2,0]

U10:%x[-1,0]/%x[0,0]

U11:%x[0,0]/%x[1,0]

# Bigram

B

#NeuroNER/src/main.py 主方法

'''

To run:

CUDA\_VISIBLE\_DEVICES="" python3.5 main.py &

CUDA\_VISIBLE\_DEVICES=1 python3.5 main.py &

CUDA\_VISIBLE\_DEVICES=2 python3.5 main.py &

CUDA\_VISIBLE\_DEVICES=3 python3.5 main.py &

'''

from \_\_future\_\_ import print\_function

import os

import argparse

from argparse import RawTextHelpFormatter

import sys

from neuroner import NeuroNER

import warnings

warnings.filterwarnings('ignore')

def load\_parameters(parameters\_filepath, arguments={}, verbose=True):

'''

Load parameters from the ini file if specified, take into account any command line argument, and ensure that each parameter is cast to the correct type.

Command line arguments take precedence over parameters specified in the parameter file.

'''

parameters = {'pretrained\_model\_folder':'../trained\_models/conll\_2003\_en',

'dataset\_text\_folder':'../data/conll2003/en',

'character\_embedding\_dimension':25,

'character\_lstm\_hidden\_state\_dimension':25,

'check\_for\_digits\_replaced\_with\_zeros':True,

'check\_for\_lowercase':True,

'debug':False,

'dropout\_rate':0.5,

'experiment\_name':'experiment',

'freeze\_token\_embeddings':False,

'gradient\_clipping\_value':5.0,

'learning\_rate':0.005,

'load\_only\_pretrained\_token\_embeddings':False,

'main\_evaluation\_mode':'conll',

'maximum\_number\_of\_epochs':100,

'number\_of\_cpu\_threads':8,

'number\_of\_gpus':0,

'optimizer':'sgd',

'output\_folder':'../output',

'patience':10,

'plot\_format':'pdf',

'reload\_character\_embeddings':True,

'reload\_character\_lstm':True,

'reload\_crf':True,

'reload\_feedforward':True,

'reload\_token\_embeddings':True,

'reload\_token\_lstm':True,

'remap\_unknown\_tokens\_to\_unk':True,

'spacylanguage':'en',

'tagging\_format':'bioes',

'token\_embedding\_dimension':100,

'token\_lstm\_hidden\_state\_dimension':100,

'token\_pretrained\_embedding\_filepath':'../data/word\_vectors/glove.6B.100d.txt',

'tokenizer':'spacy',

'train\_model':True,

'use\_character\_lstm':True,

'use\_crf':True,

'use\_pretrained\_model':False,

'verbose':False}

# If a parameter file is specified, load it

if len(parameters\_filepath) > 0:

conf\_parameters = configparser.ConfigParser()

conf\_parameters.read(parameters\_filepath, encoding="UTF-8")

nested\_parameters = utils.convert\_configparser\_to\_dictionary(conf\_parameters)

for k,v in nested\_parameters.items():

parameters.update(v)

# Ensure that any arguments the specified in the command line overwrite parameters specified in the parameter file

for k,v in arguments.items():

if arguments[k] != arguments['argument\_default\_value']:

parameters[k] = v

for k,v in parameters.items():

v = str(v)

# If the value is a list delimited with a comma, choose one element at random.

if ',' in v:

v = random.choice(v.split(','))

parameters[k] = v

# Ensure that each parameter is cast to the correct type

if k in ['character\_embedding\_dimension','character\_lstm\_hidden\_state\_dimension','token\_embedding\_dimension',

'token\_lstm\_hidden\_state\_dimension','patience','maximum\_number\_of\_epochs','maximum\_training\_time','number\_of\_cpu\_threads','number\_of\_gpus']:

parameters[k] = int(v)

elif k in ['dropout\_rate', 'learning\_rate', 'gradient\_clipping\_value']:

parameters[k] = float(v)

elif k in ['remap\_unknown\_tokens\_to\_unk', 'use\_character\_lstm', 'use\_crf', 'train\_model', 'use\_pretrained\_model', 'debug', 'verbose',

'reload\_character\_embeddings', 'reload\_character\_lstm', 'reload\_token\_embeddings', 'reload\_token\_lstm', 'reload\_feedforward', 'reload\_crf',

'check\_for\_lowercase', 'check\_for\_digits\_replaced\_with\_zeros', 'freeze\_token\_embeddings', 'load\_only\_pretrained\_token\_embeddings']:

parameters[k] = distutils.util.strtobool(v)

# If loading pretrained model, set the model hyperparameters according to the pretraining parameters

if parameters['use\_pretrained\_model']:

pretraining\_parameters = load\_parameters(parameters\_filepath=os.path.join(parameters['pretrained\_model\_folder'], 'parameters.ini'), verbose=False)[0]

for name in ['use\_character\_lstm', 'character\_embedding\_dimension', 'character\_lstm\_hidden\_state\_dimension', 'token\_embedding\_dimension', 'token\_lstm\_hidden\_state\_dimension', 'use\_crf']:

if parameters[name] != pretraining\_parameters[name]:

print('WARNING: parameter {0} was overwritten from {1} to {2} to be consistent with the pretrained model'.format(name, parameters[name], pretraining\_parameters[name]))

parameters[name] = pretraining\_parameters[name]

if verbose: pprint(parameters)

# TODO: update conf\_parameters to reflect the overriding

return parameters, conf\_parameters

def get\_valid\_dataset\_filepaths(parameters):

dataset\_filepaths = {}

dataset\_brat\_folders = {}

for dataset\_type in ['train', 'valid', 'test', 'deploy']:

dataset\_filepaths[dataset\_type] = os.path.join(parameters['dataset\_text\_folder'], '{0}.txt'.format(dataset\_type))

dataset\_brat\_folders[dataset\_type] = os.path.join(parameters['dataset\_text\_folder'], dataset\_type)

dataset\_compatible\_with\_brat\_filepath = os.path.join(parameters['dataset\_text\_folder'], '{0}\_compatible\_with\_brat.txt'.format(dataset\_type))

# Conll file exists

if os.path.isfile(dataset\_filepaths[dataset\_type]) and os.path.getsize(dataset\_filepaths[dataset\_type]) > 0:

# Brat text files exist

if os.path.exists(dataset\_brat\_folders[dataset\_type]) and len(glob.glob(os.path.join(dataset\_brat\_folders[dataset\_type], '\*.txt'))) > 0:

# Check compatibility between conll and brat files

brat\_to\_conll.check\_brat\_annotation\_and\_text\_compatibility(dataset\_brat\_folders[dataset\_type])

if os.path.exists(dataset\_compatible\_with\_brat\_filepath):

dataset\_filepaths[dataset\_type] = dataset\_compatible\_with\_brat\_filepath

conll\_to\_brat.check\_compatibility\_between\_conll\_and\_brat\_text(dataset\_filepaths[dataset\_type], dataset\_brat\_folders[dataset\_type])

# Brat text files do not exist

else:

# Populate brat text and annotation files based on conll file

conll\_to\_brat.conll\_to\_brat(dataset\_filepaths[dataset\_type], dataset\_compatible\_with\_brat\_filepath, dataset\_brat\_folders[dataset\_type], dataset\_brat\_folders[dataset\_type])

dataset\_filepaths[dataset\_type] = dataset\_compatible\_with\_brat\_filepath

# Conll file does not exist

else:

# Brat text files exist

if os.path.exists(dataset\_brat\_folders[dataset\_type]) and len(glob.glob(os.path.join(dataset\_brat\_folders[dataset\_type], '\*.txt'))) > 0:

dataset\_filepath\_for\_tokenizer = os.path.join(parameters['dataset\_text\_folder'], '{0}\_{1}.txt'.format(dataset\_type, parameters['tokenizer']))

if os.path.exists(dataset\_filepath\_for\_tokenizer):

conll\_to\_brat.check\_compatibility\_between\_conll\_and\_brat\_text(dataset\_filepath\_for\_tokenizer, dataset\_brat\_folders[dataset\_type])

else:

# Populate conll file based on brat files

brat\_to\_conll.brat\_to\_conll(dataset\_brat\_folders[dataset\_type], dataset\_filepath\_for\_tokenizer, parameters['tokenizer'], parameters['spacylanguage'])

dataset\_filepaths[dataset\_type] = dataset\_filepath\_for\_tokenizer

# Brat text files do not exist

else:

del dataset\_filepaths[dataset\_type]

del dataset\_brat\_folders[dataset\_type]

continue

if parameters['tagging\_format'] == 'bioes':

# Generate conll file with BIOES format

bioes\_filepath = os.path.join(parameters['dataset\_text\_folder'], '{0}\_bioes.txt'.format(utils.get\_basename\_without\_extension(dataset\_filepaths[dataset\_type])))

utils\_nlp.convert\_conll\_from\_bio\_to\_bioes(dataset\_filepaths[dataset\_type], bioes\_filepath)

dataset\_filepaths[dataset\_type] = bioes\_filepath

return dataset\_filepaths, dataset\_brat\_folders

def check\_parameter\_compatiblity(parameters, dataset\_filepaths):

# Check mode of operation

if parameters['train\_model']:

if 'train' not in dataset\_filepaths or 'valid' not in dataset\_filepaths:

raise IOError("If train\_model is set to True, both train and valid set must exist in the specified dataset folder: {0}".format(parameters['dataset\_text\_folder']))

elif parameters['use\_pretrained\_model']:

if 'train' in dataset\_filepaths and 'valid' in dataset\_filepaths:

print("WARNING: train and valid set exist in the specified dataset folder, but train\_model is set to FALSE: {0}".format(parameters['dataset\_text\_folder']))

if 'test' not in dataset\_filepaths and 'deploy' not in dataset\_filepaths:

raise IOError("For prediction mode, either test set and deploy set must exist in the specified dataset folder: {0}".format(parameters['dataset\_text\_folder']))

else: #if not parameters['train\_model'] and not parameters['use\_pretrained\_model']:

raise ValueError('At least one of train\_model and use\_pretrained\_model must be set to True.')

if parameters['use\_pretrained\_model']:

if all([not parameters[s] for s in ['reload\_character\_embeddings', 'reload\_character\_lstm', 'reload\_token\_embeddings', 'reload\_token\_lstm', 'reload\_feedforward', 'reload\_crf']]):

raise ValueError('If use\_pretrained\_model is set to True, at least one of reload\_character\_embeddings, reload\_character\_lstm, reload\_token\_embeddings, reload\_token\_lstm, reload\_feedforward, reload\_crf must be set to True.')

if parameters['gradient\_clipping\_value'] < 0:

parameters['gradient\_clipping\_value'] = abs(parameters['gradient\_clipping\_value'])

def parse\_arguments(arguments=None):

''' Parse the NeuroNER arguments

arguments:

arguments the arguments, optionally given as argument

'''

parser = argparse.ArgumentParser(description='''NeuroNER CLI''', formatter\_class=RawTextHelpFormatter)

parser.add\_argument('--parameters\_filepath', required=False, default=os.path.join('.','parameters.ini'), help='The parameters file')

argument\_default\_value = 'argument\_default\_dummy\_value\_please\_ignore\_d41d8cd98f00b204e9800998ecf8427e'

parser.add\_argument('--character\_embedding\_dimension', required=False, default=argument\_default\_value, help='')

parser.add\_argument('--character\_lstm\_hidden\_state\_dimension', required=False, default=argument\_default\_value, help='')

parser.add\_argument('--check\_for\_digits\_replaced\_with\_zeros', required=False, default=argument\_default\_value, help='')

parser.add\_argument('--check\_for\_lowercase', required=False, default=argument\_default\_value, help='')

parser.add\_argument('--dataset\_text\_folder', required=False, default=argument\_default\_value, help='')

parser.add\_argument('--debug', required=False, default=argument\_default\_value, help='')

parser.add\_argument('--dropout\_rate', required=False, default=argument\_default\_value, help='')

parser.add\_argument('--experiment\_name', required=False, default=argument\_default\_value, help='')

parser.add\_argument('--freeze\_token\_embeddings', required=False, default=argument\_default\_value, help='')

parser.add\_argument('--gradient\_clipping\_value', required=False, default=argument\_default\_value, help='')

parser.add\_argument('--learning\_rate', required=False, default=argument\_default\_value, help='')

parser.add\_argument('--load\_only\_pretrained\_token\_embeddings', required=False, default=argument\_default\_value, help='')

parser.add\_argument('--load\_all\_pretrained\_token\_embeddings', required=False, default=argument\_default\_value, help='')

parser.add\_argument('--main\_evaluation\_mode', required=False, default=argument\_default\_value, help='')

parser.add\_argument('--maximum\_number\_of\_epochs', required=False, default=argument\_default\_value, help='')

parser.add\_argument('--number\_of\_cpu\_threads', required=False, default=argument\_default\_value, help='')

parser.add\_argument('--number\_of\_gpus', required=False, default=argument\_default\_value, help='')

parser.add\_argument('--optimizer', required=False, default=argument\_default\_value, help='')

parser.add\_argument('--output\_folder', required=False, default=argument\_default\_value, help='')

parser.add\_argument('--patience', required=False, default=argument\_default\_value, help='')

parser.add\_argument('--plot\_format', required=False, default=argument\_default\_value, help='')

parser.add\_argument('--pretrained\_model\_folder', required=False, default=argument\_default\_value, help='')

parser.add\_argument('--reload\_character\_embeddings', required=False, default=argument\_default\_value, help='')

parser.add\_argument('--reload\_character\_lstm', required=False, default=argument\_default\_value, help='')

parser.add\_argument('--reload\_crf', required=False, default=argument\_default\_value, help='')

parser.add\_argument('--reload\_feedforward', required=False, default=argument\_default\_value, help='')

parser.add\_argument('--reload\_token\_embeddings', required=False, default=argument\_default\_value, help='')

parser.add\_argument('--reload\_token\_lstm', required=False, default=argument\_default\_value, help='')

parser.add\_argument('--remap\_unknown\_tokens\_to\_unk', required=False, default=argument\_default\_value, help='')

parser.add\_argument('--spacylanguage', required=False, default=argument\_default\_value, help='')

parser.add\_argument('--tagging\_format', required=False, default=argument\_default\_value, help='')

parser.add\_argument('--token\_embedding\_dimension', required=False, default=argument\_default\_value, help='')

parser.add\_argument('--token\_lstm\_hidden\_state\_dimension', required=False, default=argument\_default\_value, help='')

parser.add\_argument('--token\_pretrained\_embedding\_filepath', required=False, default=argument\_default\_value, help='')

parser.add\_argument('--tokenizer', required=False, default=argument\_default\_value, help='')

parser.add\_argument('--train\_model', required=False, default=argument\_default\_value, help='')

parser.add\_argument('--use\_character\_lstm', required=False, default=argument\_default\_value, help='')

parser.add\_argument('--use\_crf', required=False, default=argument\_default\_value, help='')

parser.add\_argument('--use\_pretrained\_model', required=False, default=argument\_default\_value, help='')

parser.add\_argument('--verbose', required=False, default=argument\_default\_value, help='')

try:

arguments = parser.parse\_args(args=arguments)

except:

parser.print\_help()

sys.exit(0)

arguments = vars(arguments) # http://stackoverflow.com/questions/16878315/what-is-the-right-way-to-treat-python-argparse-namespace-as-a-dictionary

arguments['argument\_default\_value'] = argument\_default\_value

return arguments

def main(argv=sys.argv):

''' NeuroNER main method

Args:

parameters\_filepath the path to the parameters file

output\_folder the path to the output folder

'''

# Parse arguments

arguments = parse\_arguments(argv[1:])

nn = NeuroNER(\*\*arguments)

nn.fit()

nn.close()

if \_\_name\_\_ == "\_\_main\_\_":

main()