# 概述

命名实体识别是NLP里的一项很基础的任务，就是指从文本中识别出命名性指称项，为关系抽取等任务做铺垫。狭义上，是识别出人命、地名和组织机构名这三类命名实体（时间、货币名称等构成规律明显的实体类型可以用正则表达式等方式识别）。当然，在特定的领域中，会相应地定义领域内的各种实体类型。

汉语作为象形文字，相比于英文等拼音文字来说，针对中文的NER任务来说往往要更有挑战性，下面列举几点：

(1) 中文文本里不像英文那样有空格作为词语的界限标志，而且“词”在中文里本来就是一个很模糊的概念，中文也不具备英文中的字母大小写等形态指示

(2) 中文的用字灵活多变，有些词语在脱离上下文语境的情况下无法判断是否是命名实体，而且就算是命名实体，当其处在不同的上下文语境下也可能是不同的实体类型

(3) 命名实体存在嵌套现象，如“北京大学第三医院”这一组织机构名中还嵌套着同样可以作为组织机构名的“北京大学”，而且这种现象在组织机构名中尤其严重

(4) 中文里广泛存在简化表达现象，如“北医三院”、“国科大”，乃至简化表达构成的命名实体，如“国科大桥”。

## 基于规则的方法

利用手工编写的规则，将文本与规则进行匹配来识别出命名实体。例如，对于中文来说，“说”、“老师”等词语可作为人名的下文，“大学”、“医院”等词语可作为组织机构名的结尾，还可以利用到词性、句法信息。在构建规则的过程中往往需要大量的语言学知识，不同语言的识别规则不尽相同，而且需要谨慎处理规则之间的冲突问题；此外，构建规则的过程费时费力、可移植性不好。

## 基于特征模板的方法

统计机器学习方法将 NER 视作序列标注任务，利用大规模语料来学习出标注模型，从而对句子的各个位置进行标注。常用的应用到 NER 任务中的模型包括生成式模型HMM、判别式模型CRF等。比较流行的方法是特征模板 + CRF(条件随机场)的方案：特征模板通常是人工定义的一些二值特征函数，试图挖掘命名实体内部以及上下文的构成特点。对于句子中的给定位置来说，提特征的位置是一个窗口，即上下文位置。而且，不同的特征模板之间可以进行组合来形成一个新的特征模板。CRF的优点在于其为一个位置进行标注的过程中可以利用到此前已经标注的信息，利用Viterbi解码来得到最优序列。对句子中的各个位置提取特征时，满足条件的特征取值为1，不满足条件的特征取值为0；然后把特征给CRF，training阶段建模标签的转移，进而在inference阶段为测试句子的各个位置做标注。

## 基于神经网络的方法

近年来，随着硬件能力的发展以及词的分布式表示（word embedding）的出现，神经网络成为可以有效处理许多NLP任务的模型。这类方法对于序列标注任务（如CWS、POS、NER）的处理方式是类似的，将token从离散one-hot表示映射到低维空间中成为稠密的embedding，随后将句子的embedding序列输入到RNN中，用神经网络自动提取特征，Softmax来预测每个token的标签。这种方法使得模型的训练成为一个端到端的整体过程，而非传统的pipeline，不依赖特征工程，是一种数据驱动的方法；但网络变种多、对参数设置依赖大，模型可解释性差。此外，这种方法的一个缺点是对每个token打标签的过程中是独立的分类，不能直接利用上文已经预测的标签（只能靠隐状态传递上文信息），进而导致预测出的标签序列可能是非法的，例如标签B-PER后面是不可能紧跟着I-LOC的，但Softmax不会利用到这个信息。

学界提出了 LSTM-CRF 模型做序列标注。在LSTM层后接入CRF层来做句子级别的标签预测，使得标注过程不再是对各个token独立分类。

# 算法

## 基于CRF的命名实体识别方法

基于CRF的命名实体识别就是把命名实体识别过程看作一个序列标注问题。其基本思路是（以汉语为例）：将给定的文本首先进行分词处理，然后对人名、简单地名和简单的组织机构名进行识别，最后识别复合地名和复合组织机构名。

基于CRF的命名实体识别方法属于有监督的学习方法，因此，需要利用已标注的大规模语料对CRF模型的参数进行训练。北京大学计算语言学研究所标注的现代汉语多级加工语料库被众多研究者用于汉语命名实体识别的模型训练。

在训练阶段，首先需要将分词语料的标记符号转化成用于命名实体序列标注的标记，如用PNB表示人名的起始用字，PNI表示名字的内部用字。类似地，用LOCB表示地名的起始用字，LOCI表示地名的内部用字；ORGB表示组织机构的起始用字，ORGI表示组织机构的内部用字。用OUT统一表示该字或词不属于某个实体。

条件随机场(Conditional Random Fields, 以下简称CRF)是给定一组输入序列条件下另一组输出序列的条件概率分布模型，在自然语言处理中得到了广泛应用。

https://www.cnblogs.com/pinard/p/7048333.html

**从随机场到马尔科夫随机场。**首先，我们来看看什么是随机场。“随机场”的名字取的很玄乎，其实理解起来不难。随机场是由若干个位置组成的整体，当给每一个位置中按照某种分布随机赋予一个值之后，其全体就叫做随机场。还是举词性标注的例子：假如我们有一个十个词形成的句子需要做词性标注。这十个词每个词的词性可以在我们已知的词性集合（名词，动词...)中去选择。当我们为每个词选择完词性后，这就形成了一个随机场。

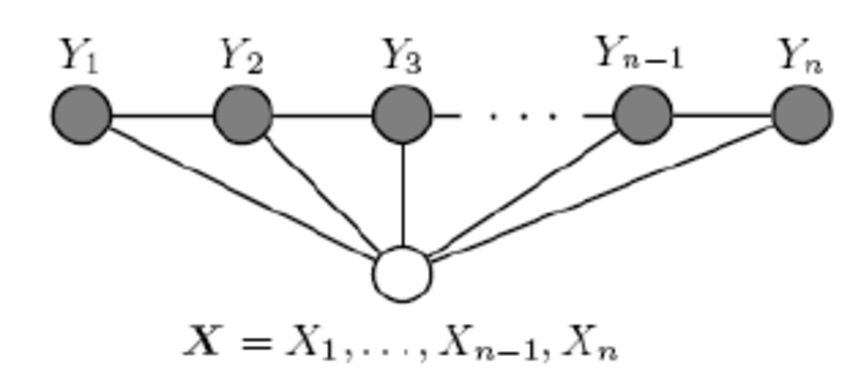
了解了随机场，我们再来看看马尔科夫随机场。马尔科夫随机场是随机场的特例，它假设随机场中某一个位置的赋值仅仅与和它相邻的位置的赋值有关，和与其不相邻的位置的赋值无关。继续举十个词的句子词性标注的例子：　如果我们假设所有词的词性只和它相邻的词的词性有关时，这个随机场就特化成一个马尔科夫随机场。比如第三个词的词性除了与自己本身的位置有关外，只与第二个词和第四个词的词性有关。

**从马尔科夫随机场到条件随机场。**理解了马尔科夫随机场，再理解CRF就容易了。CRF是马尔科夫随机场的特例，它假设马尔科夫随机场中只有X和Y两种变量，X一般是给定的，而Y一般是在给定X的条件下我们的输出。这样马尔科夫随机场就特化成了条件随机场。在我们十个词的句子词性标注的例子中，X是词，Y是词性。因此，如果我们假设它是一个马尔科夫随机场，那么它也就是一个CRF。

对于CRF，我们给出准确的数学语言描述：设X与Y是随机变量，P(Y|X)是给定X时Y的条件概率分布，若随机变量Y构成的是一个马尔科夫随机场，则称条件概率分布P(Y|X)是条件随机场。

**从条件随机场到线性链条件随机场。**注意在CRF的定义中，我们并没有要求X和Y有相同的结构。而实现中，我们一般都假设X和Y有相同的结构，即:X=(X1,X2,...Xn),Y=(Y1,Y2,...Yn)

　　　　我们一般考虑如下图所示的结构：X和Y有相同的结构的CRF就构成了线性链条件随机场(Linear chain Conditional Random Fields,以下简称 linear-CRF)。



我们再来看看 linear-CRF的数学定义：

　设X=(X1,X2,...Xn),Y=(Y1,Y2,...Yn)均为线性链表示的随机变量序列，在给定随机变量序列X的情况下，随机变量Y的条件概率分布P(Y|X)构成条件随机场，即满足马尔科夫性：

P(Yi|X,Y1,Y2,...Yn)=P(Yi|X,Yi−1,Yi+1)则称P(Y|X)为线性链条件随机场。

**线性链条件随机场的参数化形式。**对于前面讲到的linear-CRF，我们如何将其转化为可以学习的机器学习模型呢？这是通过特征函数和其权重系数来定义的。什么是特征函数呢？

在linear-CRF中，特征函数分为两类，第一类是定义在Y节点上的节点特征函数，这类特征函数只和当前节点有关，记为：

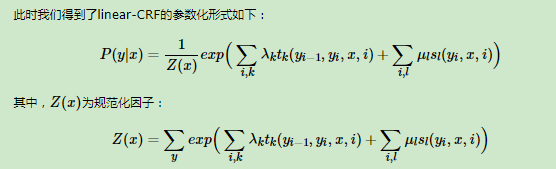


其中L是定义在该节点的节点特征函数的总个数，i是当前节点在序列的位置。

第二类是定义在Y上下文的局部特征函数，这类特征函数只和当前节点和上一个节点有关，记为：



其中K是定义在该节点的局部特征函数的总个数，i是当前节点在序列的位置。无论是节点特征函数还是局部特征函数，它们的取值只能是0或者1。即满足特征条件或者不满足特征条件。同时，我们可以为每个特征函数赋予一个权值，用以表达我们对这个特征函数的信任度。假设tk的权重系数是λk,sl的权重系数是μl,则linear-CRF由我们所有的tk,λk,sl,μl共同决定。



回到特征函数本身，每个特征函数定义了一个linear-CRF的规则，则其系数定义了这个规则的可信度。所有的规则和其可信度一起构成了我们的linear-CRF的最终的条件概率分布。

大量的实验表明，在人名、地名、组织机构名三类实体中，组织机构名识别的性能最低。一般情况下，英语和汉语人名识别的F1值都可以达到90％左右，而组织机构名识别的F1值一般都在85％左右，这也反映出组织机构名是最难识别的一种命名实体。当然，对于不同领域和不同类型的文本，测试性能会有较大的差异。

## 基于字的BiLSTM-CRF模型

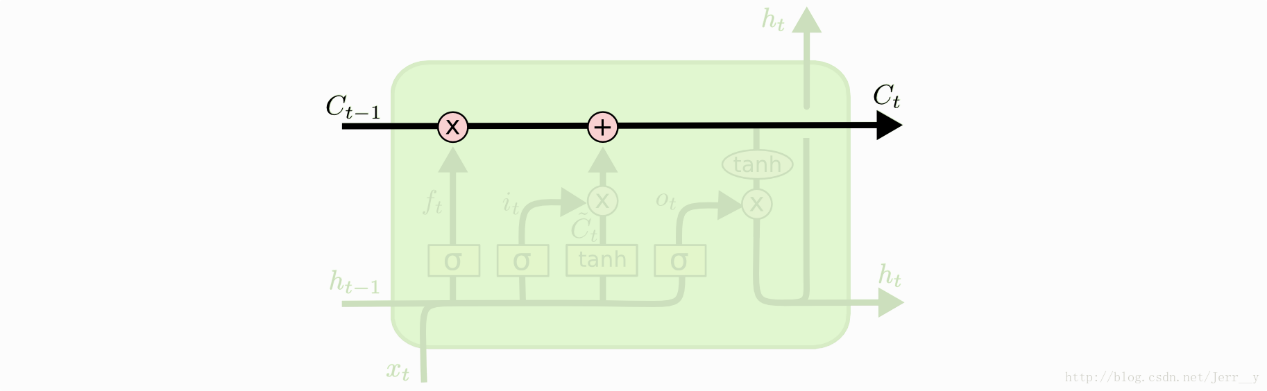
长短时记忆网络(Long Short Term Memory Network, LSTM)，是一种改进之后的循环神经网络，可以解决RNN无法处理长距离的依赖的问题，目前比较流行。

长短时记忆网络的思路：原始 RNN 的隐藏层只有一个状态，即h，它对于短期的输入非常敏感。再增加一个状态，即c，让它来保存长期的状态，称为单元状态(cell state)。



LSTMs 最关键的地方在于 cell（整个绿色的框就是一个 cell） 的状态 和 结构图上面的那条横穿的水平线。

cell 状态的传输就像一条传送带，向量从整个 cell 中穿过，只是做了少量的线性操作。这种结构能够很轻松地实现信息从整个 cell 中穿过而不做改变。（译者注：这样我们就可以实现了长时期的记忆保留了）



若只有上面的那条水平线是没办法实现添加或者删除信息的。而是通过一种叫做 门（gates） 的结构来实现的。

门 可以实现选择性地让信息通过，主要是通过一个 sigmoid 的神经层 和一个逐点相乘的操作来实现的。

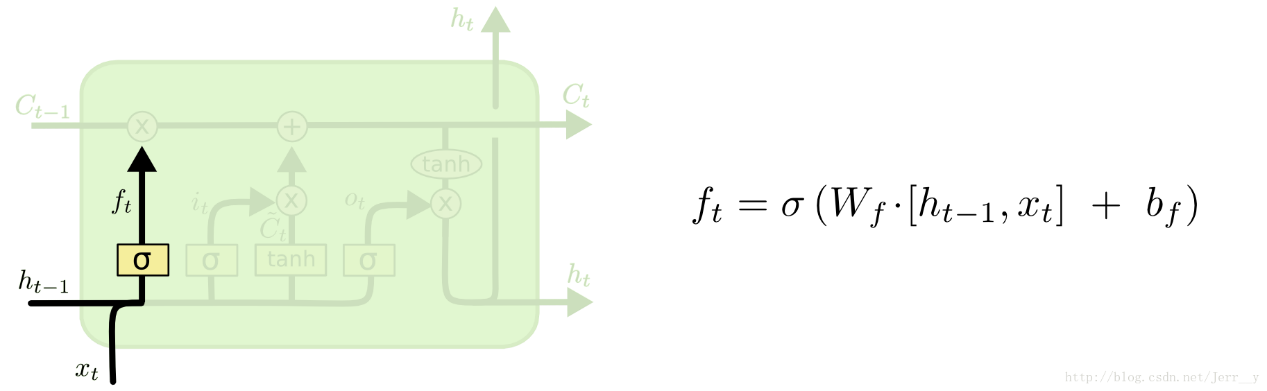
sigmoid 层输出（是一个向量）的每个元素都是一个在 0 和 1 之间的实数，表示让对应信息通过的权重（或者占比）。比如， 0 表示“不让任何信息通过”， 1 表示“让所有信息通过”。

每个 LSTM 有三个这样的门结构，来实现保护和控制信息。（译者注：分别是 “forget gate layer”, 遗忘门； “input gate layer”，传入门； “output gate layer”, 输出门）

**遗忘门**

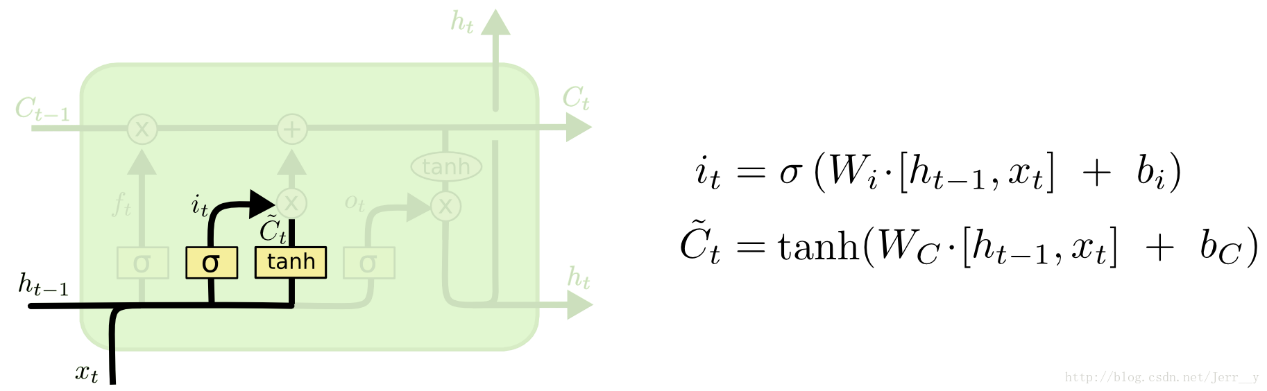
首先是 LSTM 要决定让那些信息继续通过这个 cell，这是通过一个叫做“forget gate layer ”的sigmoid 神经层来实现的。它的输入是$ h\_{t-1} 和和 x\_t $，输出是一个数值都在 0，1 之间的向量（向量长度和 cell 的状态 $ C\_{t-1} $ 一样），表示让 $C\_{t-1} $ 的各部分信息通过的比重。 0 表示“不让任何信息通过”， 1 表示“让所有信息通过”。

回到我们上面提到的语言模型中，我们要根据所有的上文信息来预测下一个词。这种情况下，每个 cell 的状态中都应该包含了当前主语的性别信息（保留信息），这样接下来我们才能够正确地使用代词。但是当我们又开始描述一个新的主语时，就应该把上文中的主语性别给忘了才对(忘记信息)。



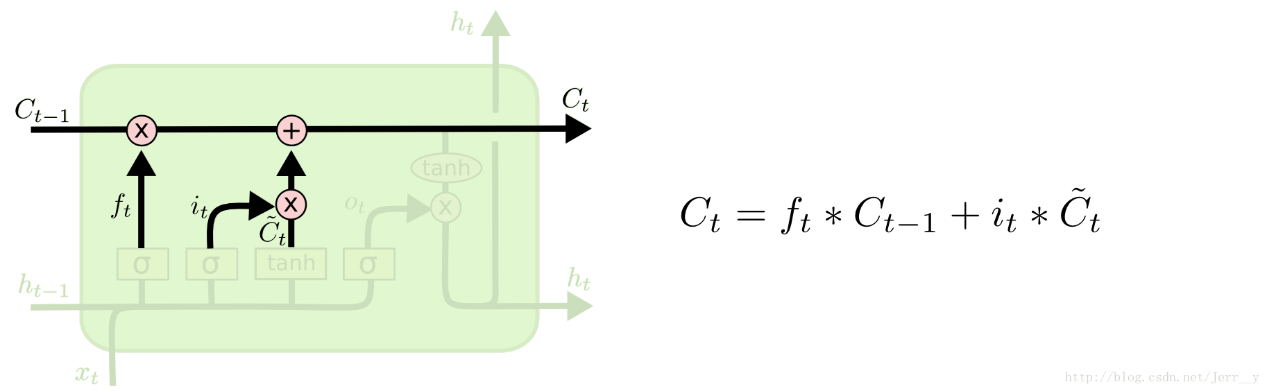
**传入门**

下一步是决定让多少新的信息加入到 cell 状态 中来。实现这个需要包括两个 步骤：首先，一个叫做“input gate layer ”的 sigmoid 层决定哪些信息需要更新；一个 tanh 层生成一个向量，也就是备选的用来更新的内容，Ct˜ 。在下一步，我们把这两部分联合起来，对 cell 的状态进行一个更新。



我们的语言模型的例子中，我们想把新的主语性别信息添加到 cell 状态中，来替换掉老的状态信息。

有了上述的结构，我们就能够更新 cell 状态了， 即把$ C\_{t-1} $更新为 $C\_{t} $。 从结构图中应该能一目了然， 首先我们把旧的状态 $C\_{t-1} 和和 f\_t 相乘，把一些不想保留的信息忘掉。然后加上相乘，把一些不想保留的信息忘掉。然后加上 i\_t \* \tilde{C\_{t}} $。这部分信息就是我们要添加的新内容。



**开始之前**

我们假设我们的数据集中有两类实体——人名和地名，与之相对应在我们的训练数据集中，有五类标签： B-Person， I- Person，B-Organization，I-Organization， O

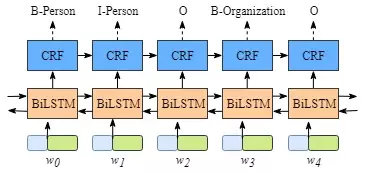
假设句子x由五个字符w1,w2,w3,w4,w5组成，其中【w1,w2】为人名类实体，【w3】为地名类实体，其他字符标签为“O”。

**BiLSTM-CRF模型**

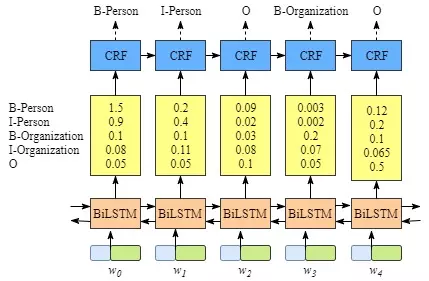
以下将给出模型的结构：

第一，句子x中的每一个单元都代表着由字嵌入或词嵌入构成的向量。其中，字嵌入是随机初始化的，词嵌入是通过数据训练得到的。所有的嵌入在训练过程中都会调整到最优。

第二，这些字或词嵌入为BiLSTM-CRF模型的输入，输出的是句子x中每个单元的标签。



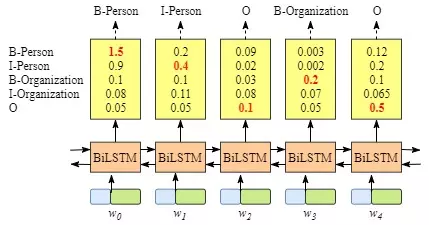
尽管一般不需要详细了解BiLSTM层的原理，但是为了更容易知道CRF层的运行原理，我们需要知道BiLSTM的输出层。



如上图所示，BiLSTM层的输出为每一个标签的预测分值，例如，对于单元w0,BiLSTM层输出的是1.5 (B-Person), 0.9 (I-Person), 0.1 (B-Organization), 0.08 (I-Organization) and 0.05 (O). 这些分值将作为CRF的输入。

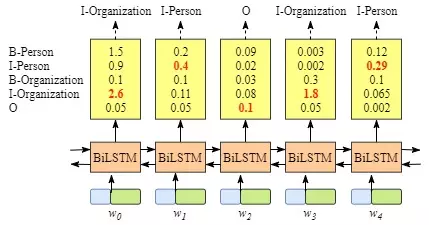
**如果没有CRF层会怎样**

你也许已经发现了，即使没有CRF层，我们也可以训练一个BiLSTM命名实体识别模型，如图3.所示：



由于BiLSTM的输出为单元的每一个标签分值，我们可以挑选分值最高的一个作为该单元的标签。例如，对于单元w0,“B-Person”有最高分值—— 1.5，因此我们可以挑选“B-Person”作为w0的预测标签。同理，我们可以得到w1——“I-Person”，w2—— “O” ，w3——“B-Organization”，w4——“O”。

虽然我们可以得到句子x中每个单元的正确标签，但是我们不能保证标签每次都是预测正确的。例如，图4.中的例子，标签序列是“I-Organization I-Person” and “B-Organization I-Person”，很显然这是错误的。



**CRF层能从训练数据中获得约束性的规则**

CRF层可以为最后预测的标签添加一些约束来保证预测的标签是合法的。在训练数据训练过程中，这些约束可以通过CRF层自动学习到。

这些约束可以是：

I：句子中第一个词总是以标签“B-“ 或 “O”开始，而不是“I-”

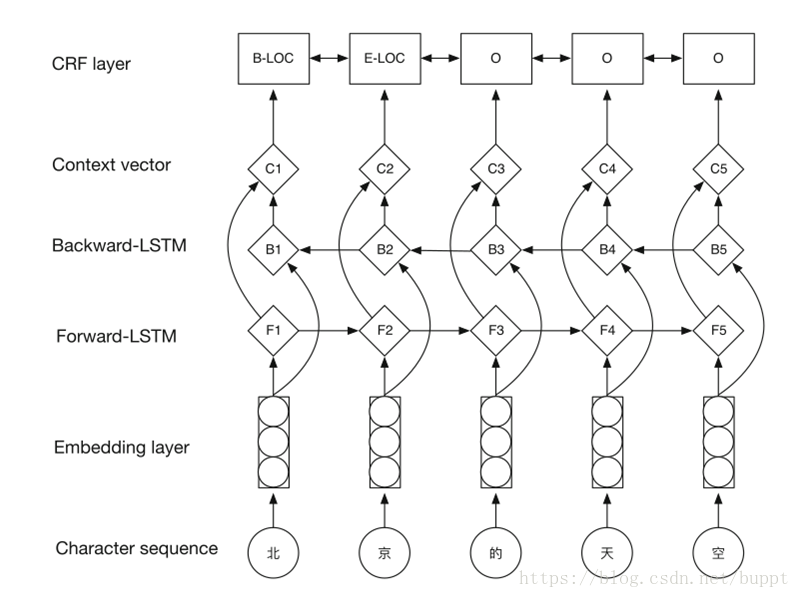
II：标签“B-label1 I-label2 I-label3 I-…”,label1, label2, label3应该属于同一类实体。例如，“B-Person I-Person” 是合法的序列, 但是“B-Person I-Organization” 是非法标签序列.

III：标签序列“O I-label” is 非法的.实体标签的首个标签应该是 “B-“ ，而非 “I-“, 换句话说,有效的标签序列应该是“O B-label”。

有了这些约束，标签序列预测中非法序列出现的概率将会大大降低。

**Bilstm+crf中的crf详解**

<https://blog.csdn.net/buppt/article/details/82227030>



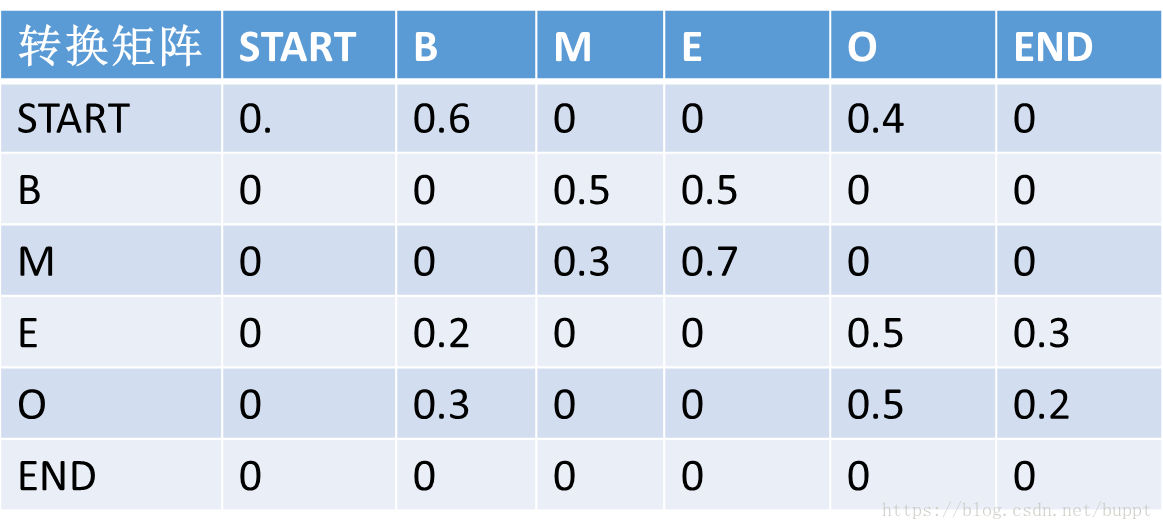
我们已知BiLSTM的输出就是每个字标注的概率。假设lstm输出概率如下所示。这里为了方便，只写了 BMEO 4种标注结果。更多的话也是相同的。



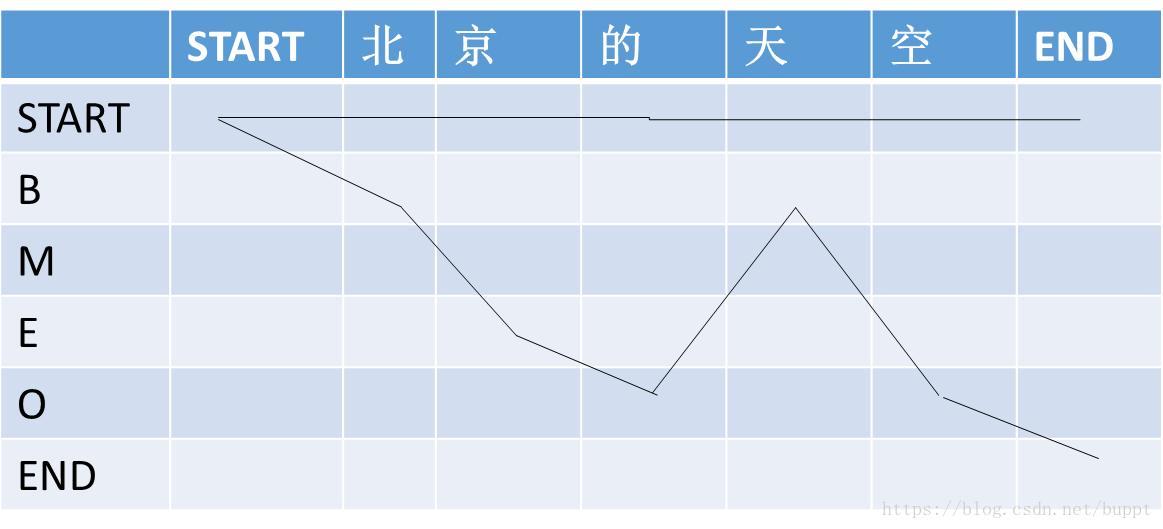
而crf首先在每句话的前面增加一个<start>字，在每句话的结尾增加一个<end>字。



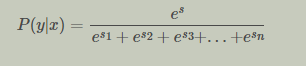
然后定义了一个转移矩阵，转移矩阵中的数值代表前面一个字标注结果到下一个字的标注结果的概率。比如下面矩阵中的第一行，代表的含义就是前一个字标注为start，下一个字标注为B 的概率是0.6，标注为O的概率就是0.4。这个矩阵是随机初始化的，里面的数值也是通过梯度下降自动更新的。



然后又定义了“路径”这个概念，一句话的每一种标注结果就代表一个路径。下图就代表两条路径。



每条路径的分数



s = 初试分数 + 转换分数

初试分数 = 路径上lstm输出分数和

转换分数 = 路径上转换矩阵分数和

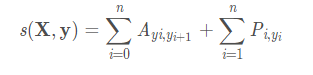
我们要找的就是分数最大的那一条路径，就可以得到这句话每个字的标注结果，然后就可以通过BME规则把实体抽取出来了。

该方法中的CRF和普通的CRF存在不同的地方，区别就在于s。在之前的介绍中，s由状态特征函数和转移特征函数组成，并有各自的权重。而bilstm+crf中的crf其s的组成要简单很多。

**损失函数：https://blog.csdn.net/qq\_41837900/article/details/100201109**

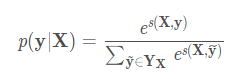
简单的说对于每一个输入X=(x1,x2,…,xn)，我们得预测结果y=(y1,y2,…,yn)

对此对这个预测结果定义 s(X,y) s(\mathbf{X}, \mathbf{y})s(X,y) 为该预测值的打分：



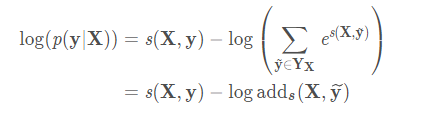
其中Pi 为第i个位置softmax输出为yi的概率， 为从yi\_̲到yi+1的转移概率。这个得分函数S就很好地弥补了传统BiLSTM的不足，因为我们当一个预测序列得分很高时，并不是各个位置都是softmax输出最大概率值对应的label，还要考虑前面转移概率相加最大，即还要符合输出规则（B后面不能再跟B），比如假设BiLSTM输出的最有可能序列为BBIBIOOO，那么因为我们的转移概率矩阵中B->B的概率很小甚至为负，那么根据s得分，这种序列不会得到最高的分数，即就不是我们想要的序列。

softmax对所有可能的标签序列产生一个序列y˜的概率,即对于训练样本X，求出所有可能的标注序列y˜的得分S(X,y˜)



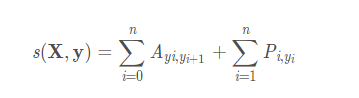
p(y∣X)为出现正确系列的概率，我们要做的就是最大化这个对数概率。

y是正确标注序列，S(X,y) S(X,{y})S(X,y)为出现正确序列的得分。



那么我们的目标就是最大化上式（即真实标记应该对应最大概率值），因为叫损失函数，所以我们也可以对上式取负然后最小化之，这样我们就可以使用梯度下降等优化方法来求解参数。在这个过程中，我们要最大化真实标记序列的概率，也就训练了转移概率矩阵A和BiLSTM中的参数。在预测的时候，根据训练好的参数求出所有可能的y序列对应的S得分，将最大的S得分的y输出。

crf\_log\_likelihood主要调用了crf\_unary\_score（计算一元分数）与crf\_binary\_score（计算二元分数），并将这两个函数的返回值相加返回。这就是：



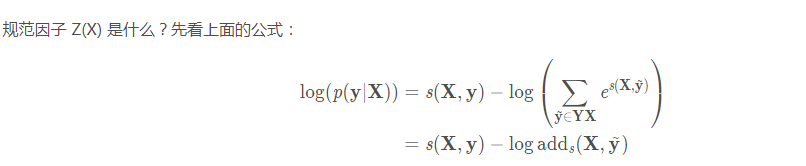
的实现，也就是

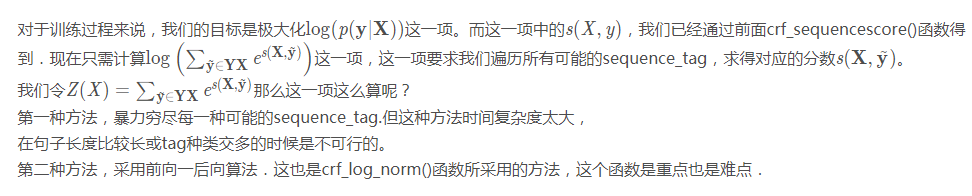
什么叫一元分数？就是在一个句子中，第i个单词对应第j个tag所得的分数用Pij 表示，然后将句子中每个单词所得的分数相加就是一元分数。

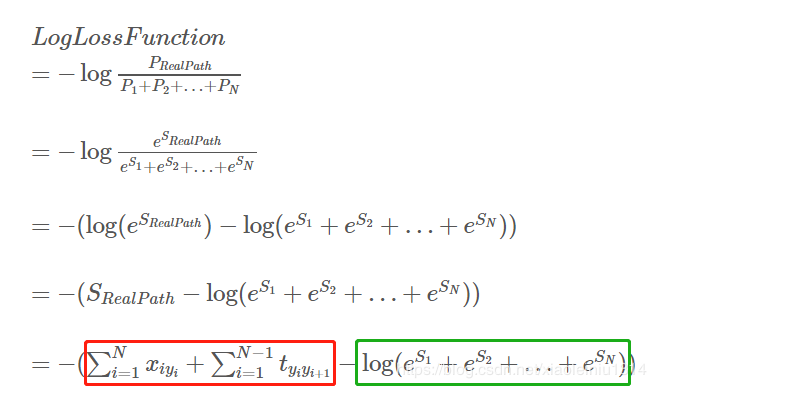
调crf\_binary\_score()计算二元分数

什么叫二元分数？就是从第i个tag转移到第j个tag所得的分数，用Ai,j 表示，然后将句子中每个位置的状态转移所得的分数相加就是二元分数。

**调用crf\_log\_norm计算规范因子Z(X)**

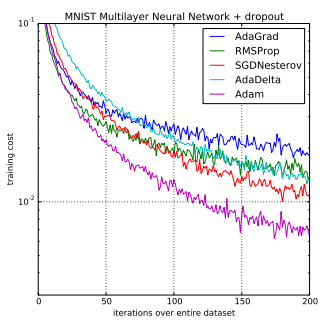






## 优化算法

“利用大型模型和数据集，我们证明了Adam可以有效地解决实际的深度学习问题。”



### Adam

Adam优化算法是一种对随机梯度下降法的扩展，最近在计算机视觉和自然语言处理中广泛应用于深度学习应用。

Adam与经典的随机梯度下降法是不同的。随机梯度下降保持一个单一的学习速率(称为alpha)，用于所有的权重更新，并且在训练过程中学习速率不会改变。每一个网络权重(参数)都保持一个学习速率，并随着学习的展开而单独地进行调整。该方法从梯度的第一次和第二次矩的预算来计算不同参数的自适应学习速率。

Adam时将随机梯度下降法两种扩展的优势结合在一起。

具体地说:

自适应梯度算法(AdaGrad)维护一个参数的学习速率，可以提高在稀疏梯度问题上的性能(例如，自然语言和计算机视觉问题)。

均方根传播(RMSProp)也维护每个参数的学习速率，根据最近的权重梯度的平均值(例如变化的速度)来调整。这意味着该算法在线上和非平稳问题上表现良好(如:噪声)。

Adam意识到AdaGrad和RMSProp的好处。与在RMSProp中基于平均第一个时刻(平均值)的参数学习速率不同，Adam也使用了梯度的第二个时刻的平均值(非中心方差)。

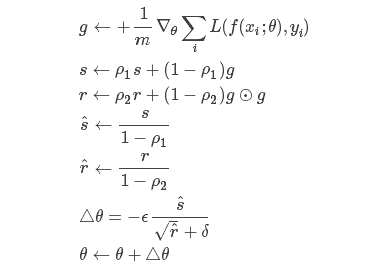
具体地说，该算法计算了梯度和平方梯度的指数移动平均值，并且参数beta1和beta2控制了这些移动平均的衰减率。移动平均值和beta1和beta2的初始值接近1.0(推荐值)，这导致了估计时间的偏差为0。这种偏差是通过第一次计算偏差估计然后再计算比可用偏差校正估计来克服的。

自适应矩估计（daptive moment estimation，Adam），是Kingma等在2015年提出的一种新的优化算法，本质上是带有动量项的RMSprop，其结合了Momentum和RMSprop算法的思想。它利用梯度的一阶矩估计 和 二阶矩估计 动态调整每个参数的学习率。

具体实现每步迭代过程:

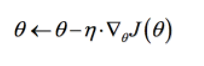
从训练集中的随机抽取一批容量为m的样本{x1,…,xm},以及相关的输出yi

计算梯度和误差,更新r和s,再根据r和s以及梯度计算参数更新量 ：



### SGD

梯度下降算法（Gradient Descent Optimization）是神经网络模型训练最常用的优化算法。梯度下降算法背后的原理：目标函数J(θ)关于参数θ的梯度将是目标函数上升最快的方向，对于最小化优化问题，只需要将参数沿着梯度相反的方向前进一个步长(学习速率)，就可以实现目标函数的下降。参数更新公式如下：



其中 是参数θ的梯度。

根据计算目标函数J(θ)采用数据量的大小，梯度下降算法又可以分为批量梯度下降算法（Batch Gradient Descent），随机梯度下降算法（Stochastic GradientDescent）和小批量梯度下降算法（Mini-batch Gradient Descent）。

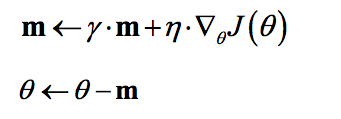
批量梯度下降算法，J(θ)是在整个训练集上计算的，如果数据集比较大，可能会面临内存不足问题，而且其收敛速度一般比较慢。

随机梯度下降算法，J(θ)是针对训练集中的一个训练样本计算的，又称为在线学习，即得到了一个样本，就可以执行一次参数更新。所以其收敛速度会快一些，但是有可能出现目标函数值震荡现象，因为高频率的参数更新导致了高方差。

小批量梯度下降算法，是折中方案，J(θ)选取训练集中一个小批量样本计算，这样可以保证训练过程更稳定，而且采用批量训练方法也可以利用矩阵计算的优势。这是目前最常用的梯度下降算法。

### momentum

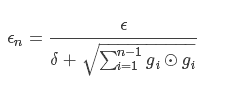
SGD方法的一个缺点是，其更新方向完全依赖于当前的batch，因而其更新十分不稳定，每次迭代计算的梯度含有比较大的噪音。解决这一问题的一个简单的做法便是引入momentum，momentum即动量，是BorisPolyak在1964年提出的，其基于物体运动时的惯性：将一个小球从山顶滚下，其初始速率很慢，但在加速度作用下速率很快增加，并最终由于阻力的存在达到一个稳定速率，即更新的时候在一定程度上保留之前更新的方向，同时利用 当前batch的梯度 微调最终的更新方向。这样一来，可以在一定程度上增加稳定性，从而学习地更快，并且还有一定摆脱局部最优的能力。其更新方程如下：



可以看到，参数更新时不仅考虑当前梯度值，而且加上了一个动量项γm，但多了一个超参γ，通常γ设置为0.5，直到初始学习稳定，然后增加到0.9或更高。相比原始梯度下降算法，动量梯度下降算法有助于加速收敛。当梯度与动量方向一致时，动量项会增加，而相反时，动量项减少，因此动量梯度下降算法可以减少训练的震荡过程。

### AdaGrad

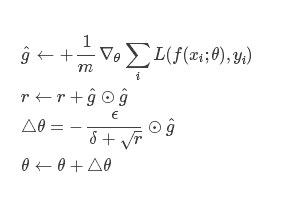
AdaGrad是Duchi在2011年提出的一种学习速率自适应的梯度下降算法。在训练迭代过程，其学习速率是逐渐衰减的，经常更新的参数其学习速率衰减更快，这是一种自适应算法。其更新过程如下：



每步迭代过程:

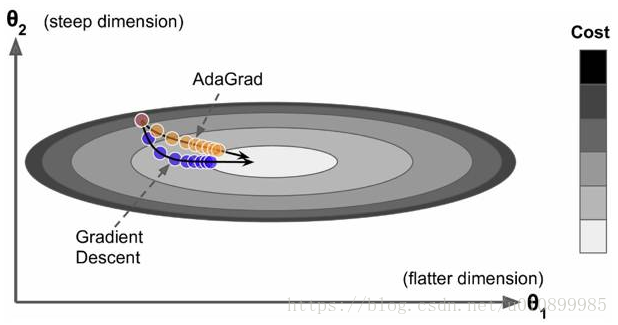
从训练集中的随机抽取一批容量为m的样本{x1,…,xm},以及相关的输出yi

计算梯度和误差,更新r,再根据r和梯度计算参数更新量:



其中，全局学习速率 ϵ, 初始参数 θ，梯度平方的累计量r初始化为0)， δ（通常为10^−7）是为了防止分母的为 0。

由于梯度平方的累计量r逐渐增加的，那么学习速率是衰减的。考虑下图所示的情况，目标函数在两个方向的坡度不一样，如果是原始的梯度下降算法，在接近坡底时收敛速度比较慢。而当采用AdaGrad，这种情况可以被改善。由于比较陡的方向梯度比较大，其学习速率将衰减得更快，这有利于参数沿着更接近坡底的方向移动，从而加速收敛。对于每个参数，随着其更新的总距离增多，其学习速率也随之变慢。

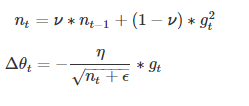


缺点: 任然要设置一个变量ϵ ,经验表明，在普通算法中也许效果不错，但在深度学习中，深度过深时会造成训练提前结束。

### Adadelta

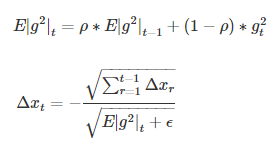
Adadelta是对Adagrad的扩展，最初方案依然是对学习率进行自适应约束，但是进行了计算上的简化。

Adagrad会累加之前所有的梯度平方，而Adadelta只累加固定大小的项，并且也不直接存储这些项，仅仅是近似计算对应的平均值。即：



其中，η是学习率，gt 是梯度

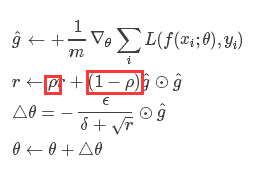
在此处Adadelta其实还是依赖于全局学习率的，但是作者做了一定处理，经过近似牛顿迭代法之后：



其中，E代表求期望。此时，可以看出Adadelta已经不用依赖于全局学习率了。

### RMSProp

RMSprop是对Adagrad算法的改进。其实思路很简单，类似Momentum思想，引入一个衰减系数，让梯度平方的累计量r 每回合都衰减一定比例：



其中，衰减系数ρ

优点：

- 相比于AdaGrad,这种方法有效减少了出现梯度爆炸情况，因此避免了学习速率过快衰减的问题。

- 适合处理非平稳目标，对于RNN效果很好

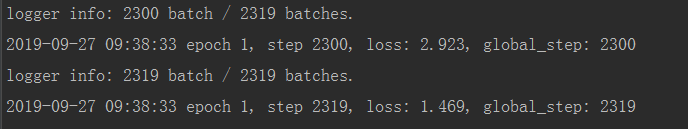
缺点：

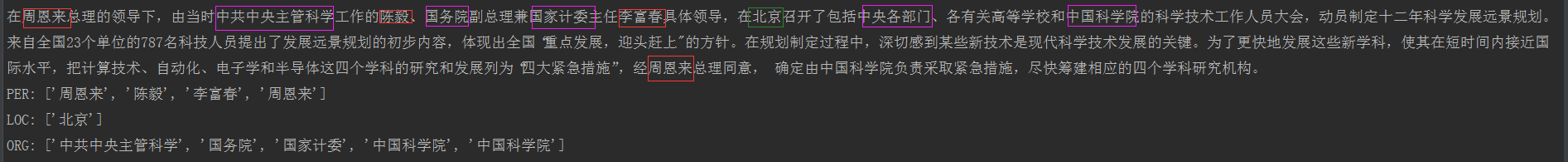
- 又引入了新的超参—衰减系数ρ

- 依然依赖于全局学习速率，

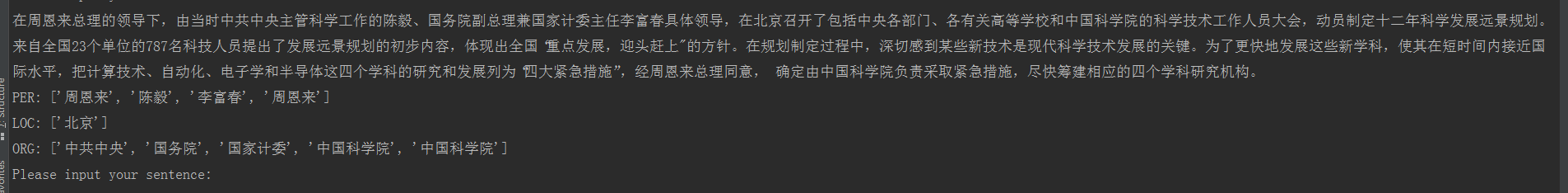
总结：RMSprop算是Adagrad的一种发展，和Adadelta的变体，效果趋于二者之间。

我自己训练的结果epoch=1， 差不多训练了1个小时（1569548311）





自己训练的结果epoch=40



别人训练的结果epoch=40

