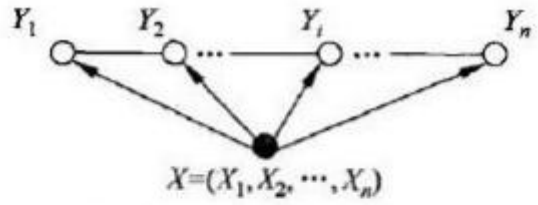
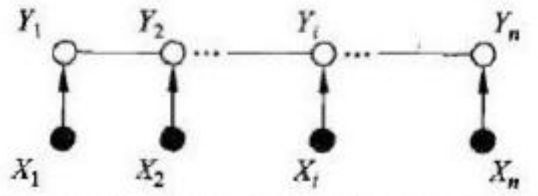
条件随机场

条件随机场（Conditional Random Fields），是在给定一组输入随机变量条件下另外一组输出随机变量的条件概率分布模型，它是一种判别式的概率无向图模型，既然是判别式，拿就是对条件概率分布建模。

综上所述，设有线性链结构的随机变量序列 X=(X1,X2,...,Xn),Y=(Y1,Y2,...,YN)，在给定观察序列 X 的条件下，随机变量序列 Y 的条件概率分布为 P(Y|X)，若其满足马尔科夫特性，即

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



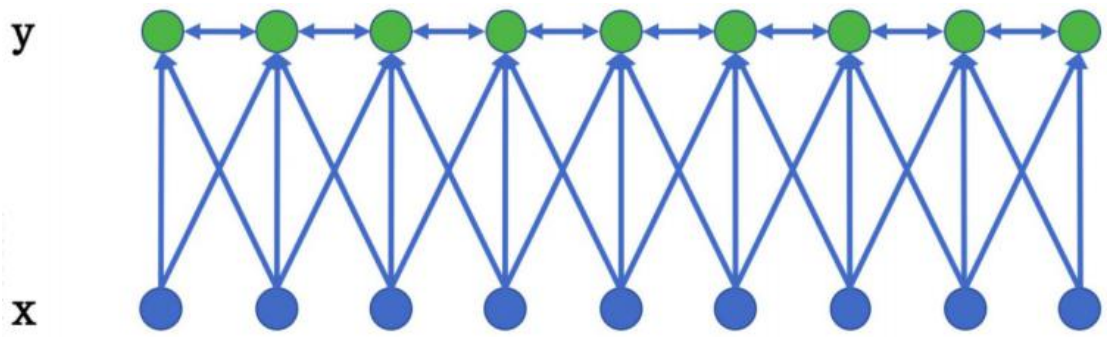


条件随机场（ConditionalRandom Field，CRF）是经典 NER 的主流模型。

它的目标函数不仅考虑输入的状态特征函数，而且还包含了标签转移特征函数。

在训练时可以使用 SGD 学习模型参数。在已知模型时，给输入序列求预测输出序列即求使目标函数最大化的最优序列，是一个动态规划问题，可以使用 Viterbi 算法解码来得到最优标签序列。

CRF 的优点在于其为一个位置进行标注的过程中可以利用丰富的内部及上下文特征信息。



介绍 CRF

假设你有许多小明同学一天内不同时段的照片，从小明提裤子起床到脱裤子睡觉各个时间段都有（小明是照片控！）。现在的任务是对这些照片进行分类。比如有的照片是吃饭，那就给它打上吃饭的标签；有的照片是跑步时拍的，那就打上跑步的标签；有的照片是开会时拍的，那就打上开会的标签。问题来了，你准备怎么干？

一个简单直观的办法就是，不管这些照片之间的时间顺序，想办法训练出一个多元分类器。就是用一些打好标签的照片作为训练数据，训练出一个模型，直接根据照片的特征来分类。例如，如果照片是早上 6:00 拍的，且画面是黑暗的，那就给它打上睡觉的标签;如果照片上有车，那就给它打上开车的标签。

这样可行吗？

乍一看可以！但实际上，由于我们忽略了这些照片之间的时间顺序这一重要信息，我们的分类器会有缺陷的。举个例子，假如有一张小明闭着嘴的照片，怎么分类？显然难以直接判断，需要参考闭嘴之前的照片，如果之前的照片显示小明在吃饭，那这个闭嘴的照片很可能是小明在咀嚼食物准备下咽，可以给它打上吃饭的标签；如果之前的照片显示小明在唱歌，那这个闭嘴的照片很可能是小明唱歌瞬间的抓拍，可以给它打上唱歌的标签。

所以，为了让我们的分类器能够有更好的表现，在为一张照片分类时，我们必须将与它相邻的照片的标签信息考虑进来。这——就是条件随机场(CRF)大显身手的地方！

POS 标注问题

非常简单的，就是给一个句子中的每个单词注明词性。比如这句话：“Bob drank coffee at Starbucks”，注明每个单词的词性后是这样的：“Bob (名词) drank(动词) coffee(名词) at(介词) Starbucks(名词)”。

下面，就用条件随机场来解决这个问题。

以上面的话为例，有 5 个单词，我们将：**(名词，动词，名词，介词，名词)**作为一个标注序列，称为 l，可选的标注序列有很多种，比如 l 还可以是这样：**（名词，动词，动词，介词，名词）**，我们要在这么多的可选标注序列中，挑选出一个**最靠谱**的作为我们对这句话的标注。

怎么判断一个标注序列靠谱不靠谱呢？

就我们上面展示的两个标注序列来说，第二个显然不如第一个靠谱，因为它把第二、第三个单词都标注成了动词，动词后面接动词，这在一个句子中通常是说不通的。

假如我们给每一个标注序列打分，打分越高代表这个标注序列越靠谱，我们至少可以说，凡是标注中出现了**动词后面还是动词**的标注序列，要给它**减分！！**

上面所说的**动词后面还是动词**就是一个特征函数，我们可以定义一个特征函数集合，用这个特征函数集合来为一个标注序列打分，并据此选出最靠谱的标注序列。也就是说，

每一个特征函数都可以用来为一个标注序列评分，把集合中所有特征函数对同一个标注序列的评分综合起来，就是这个标注序列最终的评分值。

CRF 中的特征函数们

现在，我们正式地定义一下什么是 CRF 中的特征函数，所谓特征函数，就是这样的函数，它接受四个参数：

- 句子 s（就是我们要标注词性的句子）
- i，用来表示句子 s 中第 i 个单词
- l_i，表示要评分的标注序列给第 i 个单词标注的词性
- l_i-1，表示要评分的标注序列给第 i-1 个单词标注的词性

它的输出值是 0 或者 1,0 表示要评分的标注序列不符合这个特征，1 表示要评分的标注序列符合这个特征。

Note:这里，我们的特征函数仅仅依靠当前单词的标签和它前面的单词的标签对标注序列进行评判，这样建立的 CRF 也叫作线性链 CRF，这是 CRF 中的一种简单情况。为简单起见，本文中我们仅考虑线性链 CRF。

特征到概率

定义好一组特征函数后，我们要给每个特征函数 f_j 赋予一个权重λ_j。现在，只要有一个句子 s，有一个标注序列 l，我们就可以利用前面定义的特征函数集来对 l 评分。

$$score(l|s) = \sum_{j=1}^{m} \sum_{i=1}^{n} \lambda_j f_j(s, i, l_i, l_{i-1})$$

上式中有两个相加，外面的相加用来相加每一个特征函数 f_j，里面的相加用来相加句子中每个位置的单词的的特征值。

对这个分数进行**指数化和标准化**，我们就可以得到标注序列 l 的概率值**p(l|s)**，如下所示：

$$p(l|s) = \frac{exp[score(l|s)]}{\sum_{l'} exp[score(l'|s)]} = \frac{exp[\sum_{j=1}^{m} \sum_{i=1}^{n} \lambda_j f_j(s,i,l_i,l_{i-1})]}{\sum_{l'} exp[\sum_{j=1}^{m} \sum_{i=1}^{n} \lambda_j f_j(s,i,l'_i,l'_{i-1})]}$$

特征函数的例子

前面我们已经举过特征函数的例子，下面我们再看几个具体的例子，帮助增强大家的感性认识。

$$f_1(s, i, l_i, l_{i-1}) = 1$$

当 l_i 是“副词”并且第 i 个单词以“ly”结尾时，我们就让 f1 = 1，其他情况 f1 为 0。不难想到，f1 特征函数的权重λ1 应当是正的。而且λ1 越大，表示我们越倾向于采用那些把以“ly”结尾的单词标注为“副词”的标注序列

$$f_2(s,i,l_i,l_{i-1})=1$$

如果 i=1，l_i=动词，并且句子 s 是以“？”结尾时，f2=1，其他情况 f2=0。同样，λ2 应当是正的，并且λ2 越大，表示我们越倾向于采用那些把问句的第一个单词标注为“动词”的标注序列。

$$f_3(s,i,l_i,l_{i-1})=1$$

当 l_i-1 是介词，l_i 是名词时，f3 = 1，其他情况 f3=0。λ3 也应当是正的，并且λ3 越大，说明我们越认为介词后面应当跟一个名词。

$$f_4(s,i,l_i,l_{i-1})=1$$

如果 l_i 和 l_i-1 都是介词，那么 f4 等于 1，其他情况 f4=0。这里，我们应当可以想到λ4 是负的，并且λ4 的绝对值越大，表示我们越不认可介词后面还是介词的标注序列。

好了，一个条件随机场就这样建立起来了，让我们总结一下：为了建一个条件随机场，我们首先要定义一个特征函数集，每个特征函数都以整个句子 s，当前位置 i，位置 i 和 i-1 的标签为输入。然后为每一个特征函数赋予一个权重，然后针对每一个标注序列 l，对所有的特征函数加权求和，必要的话，可以把求和的值转化为一个概率值。

对比逻辑回归

CRF 概率的形式或许看起来眼熟

$$p(l|s)=\frac{exp[\sum_{j=1}^{m}\sum_{i=1}^{n}f_j(s,i,l_i,l_{i-1})]}{\sum_{l'}exp[\sum_{j=1}^{m}\sum_{i=1}^{n}f_j(s,i,l'_i,l'_{i-1})]}$$

那是因为 CRF 确实基本上是逻辑回归的序列版本，鉴于逻辑回归是一个对数线性模型用于分类，CRF 是一个对数线性模型用于序列打标签。

对比 HMM

回顾 HMM 是另一个模型用于 POS 标注和整体上可用于序列打标签问题。鉴于 CRF 抛出成捆的特征函数去获得标签分数，HMM 采用生成式方法去打标签，定义：

$$p(l,s)=p(l_1)\prod_i p(l_i|l_{i-1})p(w_i|l_i)$$

这里，

- $p(l_i|l_{i-1})$是转移概率，例如介词后面跟动词的概率
- $p(w_i|l_i)$是发射概率，例如名词发射出词“爸爸”的概率

那么 HMM 如何和 CRF 进行比较呢？CRF 更强大，它可以做 HMM 可以完成的一切，

并且还能做 HMM 不能做的。一种方法去理解这个如下：

注意 HMM 概率的对数形式

$$\log p(l,s) = \log p(l_0) + \sum_i \log p(l_i|l_{i-1}) + \sum_i \log p(w_i|l_i)$$

这就是 CRF 的对数线性形式，如果我们考虑这些对数概率为相对应的二元的转化和发射指标特征的权重。

即，我们可以构建一个 CRF 等同于任意 HMM...

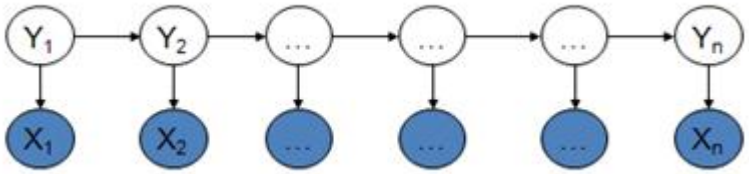
- 对于每个 HMM 的转移概率 $p(l_i = y|l_{i-1} = x)$，定义一组形式上为 $f_{x,y}(s,i,l_i,l_{i-1}) = 1 \text{ if } l_i = y \text{ and } l_{i-1} = x$ CRF 的转移特征。给定每个特征一个权重 $w_{x,y} = \log p(l_i = y|l_{i-1} = x)$
- 类似的，对于每个 HMM 的发射概率 $p(w_i = z|l_i = x)$，定义一组形式上为 $g_{x,y}(s,i,l_i,l_{i-1}) = 1 \text{ if } w_i = z \text{ and } l_i = x$ CRF 发射特征。给定每个特征一个权重 $w_{x,z} = \log p(w_i = z|l_i = x)$

因此，由 CRF 计算的分数 p(l|s)使用这些特征函数是精确地成比例的对应于 HMM 计算的分数，所以每一个 HMM 是等同于一些 CRF。

可是 CRFs 可以建模更丰富的标签分布，这里有两个原因：

1. CRFs 可以定义更大的特征集合。然后 HMMs 必然是局部在本质上（因为它们被限制到二元的转换和发射特征函数 ,强迫每个单词仅依赖于当前的标签和每个标签仅依赖于之前的标签），CRFs 可以使用更多全局的特征。例如，一个特征在我们的 POS 标签的概率增长，一个句子首个单词标注为动词如果句子最后包含一个问号符号。
2. CRFs 可以有任意的权重。鉴于 HMM 的概率必须满足某些限制，例如 $0 <= p(w_i|l_i) <= 1, \sum_w p(w_i = w|l_1) = 1$，CRF 的权重是不限制的，例如 $\log p(w_i|l_i)$ 可以是任意它要的。

HMM



HMM 模型中存在两个假设：

1，输出观察值之间严格独立

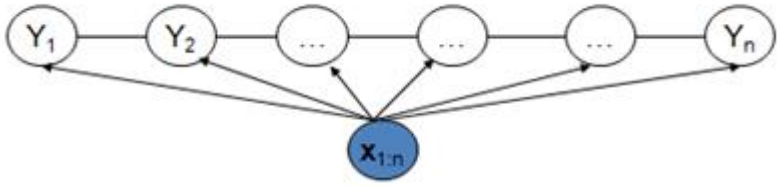
2，状态的转移过程中当前状态只与前一状态有关（一阶马尔可夫模型）

标注偏置问题

有 3 个状态 label a, label b, label c

在训练语料库中，a 转移到 b 的概率，大于 a 转移到 c 的概率，HMM 是通过统计是不是统计出来 a 到 b 还有 a 到 c 的转移概率。造成 HMM 在进行测试时，始终只能出现 a 到 b 的状态！

CRF



CRF 能够采用丰富的特征，其无向图结构，能够避免这个问题

权重学习

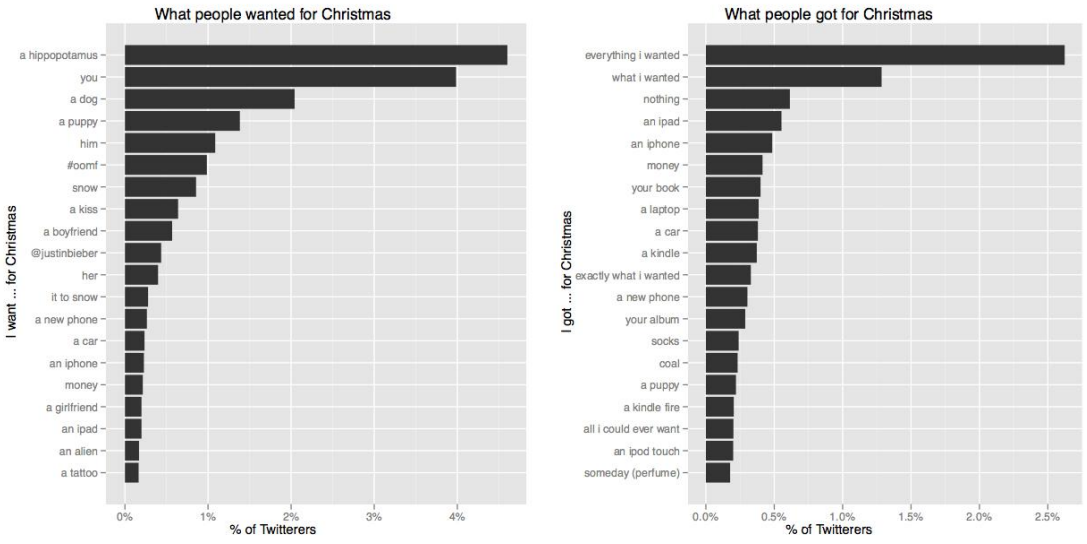
让我们回到如何学习 CRF 的特征权重的问题，一种方法，就是去使用梯度下降。

假设我们有一捆训练样本，句子和对应的 part-of-speech 标签。随机初始化我们 CRF 模型的权重，调整这些速记初始化的参数，对于每一个训练样本。

有趣的例子

好了，POS 标注有点无聊，这里存在大量 POS 标注工具。什么时候或许你可以使用 CRF 在现实生活中？

假设你想要挖掘 Twitter 上在圣诞节人们收到礼物的种类：

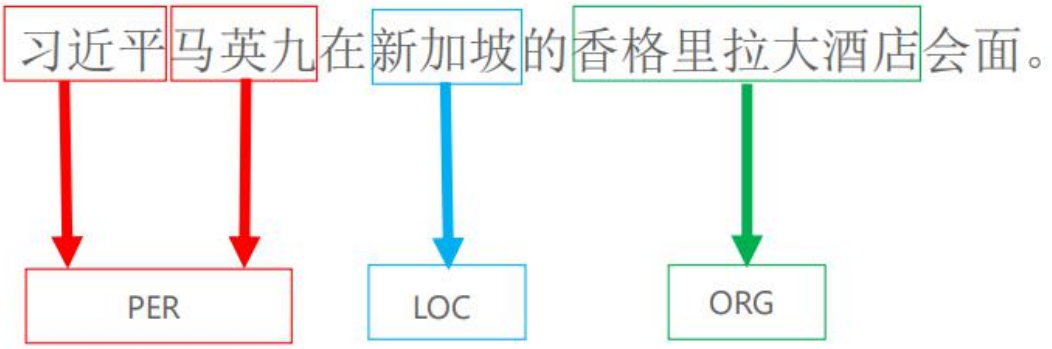


你可以怎么指出哪些词对应礼物？

去收集像上图一样的数据，简单地查看短语形式“I want XXX for Christmas” and “I got XXX for Christmas”。可是，一个更复杂地 CRF 变种可以使用一个 GIFT 像 POS 标注一样地标签(增加一些标签 GIFT-GIVER 和 GIFT-RECEIVER，在从谁那里谁得到什么，去获得更多信息)并且像对待 POS 标注问题一样。特征可以基于事物像“这个词是一个 GIFT 如果前面的词是一个 GIFT-RECEIVER 并且词前面是‘gave’”或者“这个词是一个 GIFT 如果下两个词是‘for Christmas’”。

命名实体识别

◆现实世界中存在大量的半结构化和非结构化数据。
◆构建知识图谱的关键是如何获取领域知识。
◆实体识别是从半结构化数据或非结构化数据中获取知识的重要方法。



主要方法

早期方法

■基于规则和词典的方法

基于统计学习的方法

■ HMM/CRF
■混合方法
➢ 统计学习方法之间或内部层叠融合。
➢ 规则、词典和机器学习方法之间的融合。
➢ 将各类模型、算法结合起来，将前一级模型的结果作为下一级的训练数据

深度学习的方法

■NN/CNN-CRF
■RNN-CRF/LSTM-CRF
最新进展

■注意力机制

■迁移学习

标注策略

IOB 比如识别人名：PER B：begin O：out I：internal

BIES 或 BMES M：middle S：single

我 O

叫 O

张 B-PER

三 I-PER

， O

我

喜

欢

标注：人工标注成本比较高

A. CRF 层之于 BiLSTM

概述

接下来要讲的内容包括：

✓ 介绍 - CRF 层之于 BiLSTM 作命名实体识别 NER 的整体思路

✓ 一个具体的例子 - 通过一个例子一步步的解释 CRF 层如何工作的

先验知识

你仅需要知道什么是命名实体识别。如果你不知道神经网络，CRF 或者其它相关知识，请不要担心，我会尽可能直观的解释这些。

1.介绍

对于命名实体识别任务，基于神经网络是非常流行和常见的。例如，使用 BiLSTM-CRF 结合词或字嵌入的网络模型来作命名实体识别。下面将会用这个网络模型来解释 CRF 层是如何工作的。

如果你不知道 BiLSTM 和 CRF 的细节，就记住它们是两个不同的层在 NER 模型中。

1.1 在我们开始之前

我们假设我们有一个数据集有两类实体类型，Person 和 Organization。因此事实上在我们的数据集中，我们有 5 个实体标签：

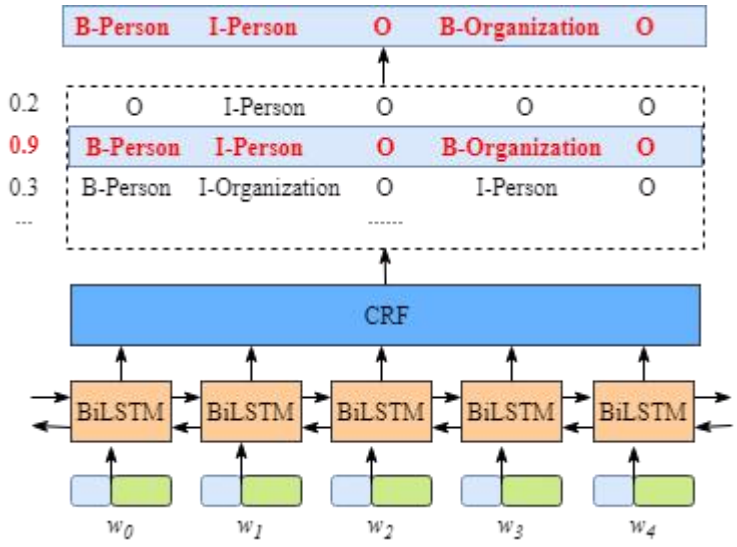
- B-Person
- I- Person
- B-Organization
- I-Organization
- O

进而，x 是一个句子包含 5 个词，w0,w1,w2,w3,w4。更多地，在句子 x, [w0,w1] 是一个 Person entity, [w3] 是一个 Organization entity 和其它的是“O”。

1.2 BiLSTM-CRF 模型

对于这个模型我将会给出一个主要的介绍。

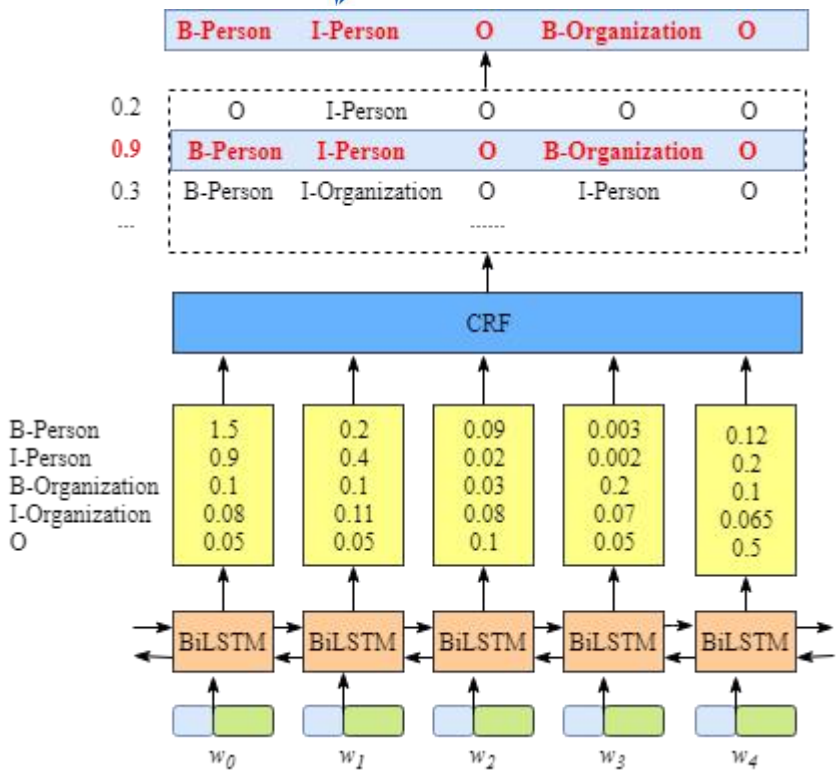
正如下图所示：



首先，在句子每个词，x 被表达为一个向量，向量由包含词的字嵌入和词嵌入组成。字的嵌入是随机初始化的。词嵌入通常是来自于预训练的词嵌入模型文件。在整个训练过程中所有的嵌入将会被细粒度的调优。

其次，BiLSTM-CRF 模型的输入是那些嵌入向量，输出是对于句子 x 中词的预测标签。

虽然，没有必要去知道 BiLSTM 层的细节，为了去更容易的理解 CRF 层，我们不得不知道 BiLSTM 层的输出的含义是什么。

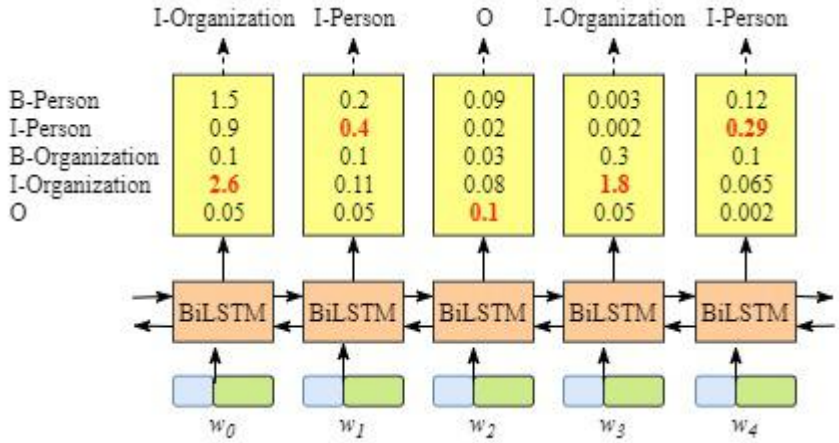


上图展示了 BiLSTM 层的输出是每个标签的分数。例如，对于 w0，BiLSTM 节点的输出是 1.5 (B-Person), 0.9 (I-Person), 0.1 (B-Organization), 0.08 (I-Organization)和 0.05 (O)。这些分数将会被作为 CRF 层的输入。

然后，由 BiLSTM 块预测的所有分数将会被带入 CRF 层。在 CRF 层中，有最高预测分数的标签序列将会被选为最优的答案。

1.3 如果我们不加 CRF 层

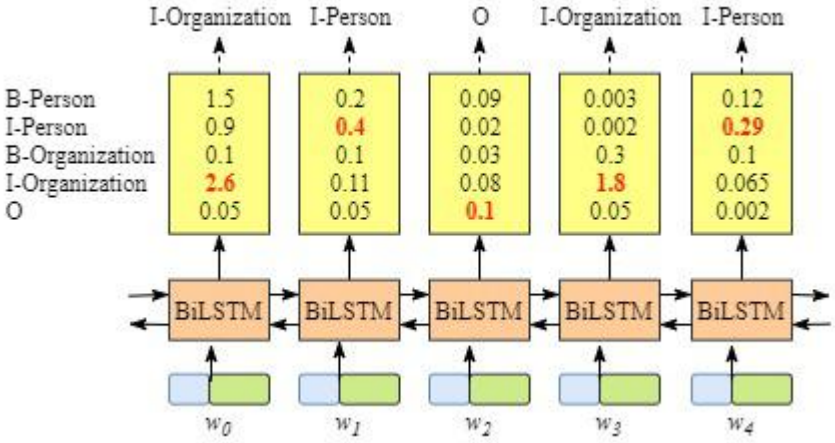
你们或许已经发现，即使没有 CRF 层，换句话说，我们可以训练一个 BiLSTM 模型作 NER 就如下图所示：



因为对于每一个单词 BiLSTM 模型的输出是各种标签的分数。我们可以对于每个单词选择有最高分数的标签。

例如，对于 w0，“B-Person”有最高的分数(1.5)，因此我们可以选择“B-Person”作为它的最好的预测标签。用同样的方式，我们可以选择“I-Person”给 w1，“O”给 w2，“B-Organization”给 w3 和“O”给 w4。

虽然我们可以得到句子 x 正确的标签序列在这个例子中，但是并不是总是这样。再试一下下面图中的例子。



很明显，“I-Organization I-Person”和“B-Organization I-Person”这些输出是无效的。

1.4 CRF 层可以从训练数据学习限制

CRF 层可以加一些限制给最后的预测标签去确保它们是有效的。这些限制通过训练过程可以被 CRF 层从训练集数据中自动学到。

这些限制可以是：

- 句子开头首个词的标签应该是“B-“ 或“O”，而不是“I-“。
- “B-label1 I-label2 I-label3 I-...”，在这个模式中，label1, label2, label3 ... 应该是同样的命名实体标签．例如，“B-Person I-Person”是有效的，但“B-Person I-Organization”是无效的。
- “O I-label”是无效的。一个命名实体的第一个标签应该以“B-“ 开头，而不是“I-“，换句话说，有效的模式应该是“O B-label”。
- ...

有了这些有用的限制，无效的预测标签序列的数量会急剧的下降。

接下来

在接下来的一节，我将会分析 CRF 的损失函数，去解释如何并且为什么 CRF 层可以从训练数据中学习上面提到的那些限制。

B. CRF 层之于 BiLSTM

回顾

在前面一节中，我们知道了 CRF 层可以从训练数据中学习到一些限制，为了确保最后的预测实体标签训练是有效的。

这些限制可以是：

- 句子开头首个词的标签应该是“B-“ 或“O”，而不是“I-“。
- “B-label1 I-label2 I-label3 I-...”，在这个模式中, label1, label2, label3 ... 应该是同样的命名实体标签. 例如，“B-Person I-Person” 是有效的，但 “B-Person I-Organization” 是无效的。
- “O I-label” 是无效的。一个命名实体的第一个标签应该以 “B-“ 开头 ,而不是 “I-“, 换句话说，有效的模式应该是 “O B-label” 。
- ...

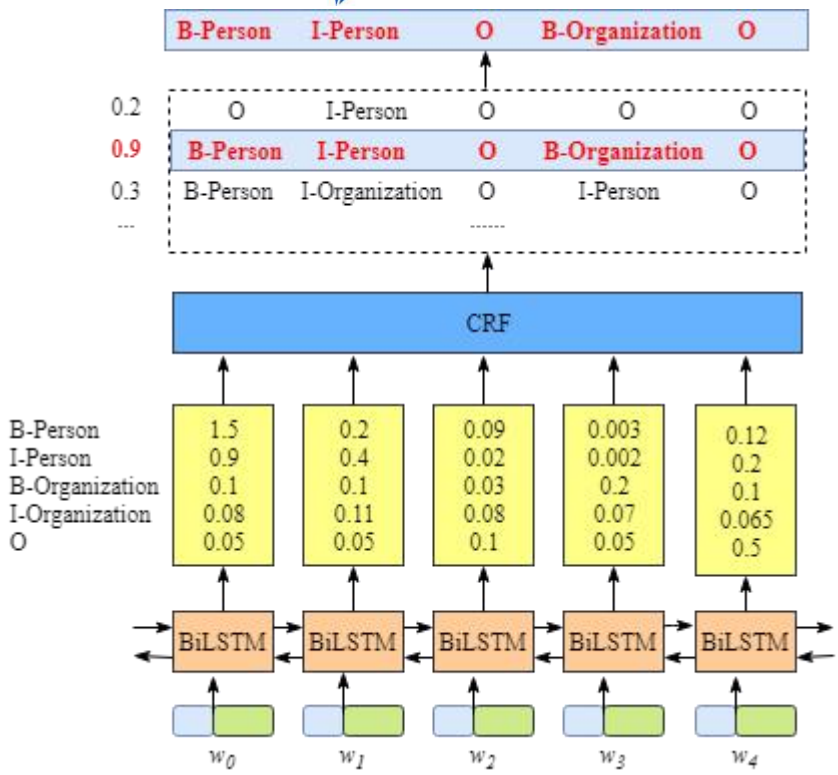
在学习了这一小节后，你将会知道为什么 CRF 层可以学习到这些限制。

2. CRF 层

在 CRF 层的损失函数中，我们有两种类型的分数。这两分数是 CRF 层的关键。

2.1 Emission score

第一个是 emission score。这些 emission scores 来自于 BiLSTM 层。正如图所示，例如, w0 是标签 B-Person 的分数为 1.5。



为了方便，我们会给每一个标签一个索引号正如下表所示，

Label	Index
B-Person	0
I-Person	1
B-Organization	2
I-Organization	3
O	4

我们使用 xiyj 去表达一个 emission score。i 是词的索引同时 yj 是标签的索引。例如，根据图,xi=1,yj=2= $x_{\text{w1,B-Organization}}$ =0.1 这意味着 w1 作为 B-Organization 的分数是 0.1。

2.2 Transition score

我们使用 tyiyj 去表达一个 transition score。例如, $t_{\text{B-Person,I-Person}}$ =0.9 意味着标签转移 B−Person→I−Person 的分数是 0.9。因此，我们有一个转移分数矩阵存储了所有的标签和标签之间的分数。

为了去使得转移分数矩阵更加的鲁棒，我们将多添加两个标签 START 和 END。START 意味着句子的开始，不是第一个词。END 意味着句子的结束。

这里有一个转移分数矩阵的例子包含了额外添加的 START 和 END 标签。

	START	B-Person	I-Person	B-Organization	I-Organization	O	END
START	0	0.8	0.007	0.7	0.0008	0.9	0.08
B-Person	0	0.6	0.9	0.2	0.0006	0.6	0.009
I-Person	-1	0.5	0.53	0.55	0.0003	0.85	0.008
B-Organization	0.9	0.5	0.0003	0.25	0.8	0.77	0.006
I-Organization	-0.9	0.45	0.007	0.7	0.65	0.76	0.2
O	0	0.65	0.0007	0.7	0.0008	0.9	0.08
END	0	0	0	0	0	0	0

正如上面所示，我们可以发现这个转移矩阵已经学到了一些有用的限制。

- ✓ 在句子中第一个单词的标签应该由“B-“或“O”开头,不是“I-“(从 START 到 I-Person 或者 I-Organization 转移的分数是非常低的)
- ✓ “B-label1 I-label2 I-label3 I-...”，在这个模式中, label1, label2, label3 ...应该是同一个命名实体标签。例如，“B-Person I-Person”是有效的，但“B-Person I-Organization”是无效的。(例如，从 B-Organization 到 I-Person 的分数只有 0.0003，这跟其它相比是非常低的分数)
- ✓ “O I-label”是无效的。一个命名实体的第一个标签应该由“B-“开头而不是“I-“,换句话说,有效的模式应该是“O B-label”(再有,例如，$t_{\text{O,I-Person}}$ 的分数是非常小的)
- ✓ ...

你或许想问一个关于矩阵的问题。哪里或者如何得到这个转移矩阵？

事实上，这个矩阵是 BiLSTM-CRF 模型的参数。在你训练这个模型之前，你可以随机初始化矩阵中的所有转移分数。在模型的训练过程中所有随机的分数将会被自动地更新。换句话说, CRF 层可以由它自己学习那些限制。我们没有必要去手动建立这个矩阵。这些分数随着训练迭代的增加会逐渐变得越来越合理。

接下来

2.3 CRF loss function

介绍 CRF 的损失函数，它由真正路径的分数和所有可能路径的总分数构成。

2.4 Real path score

如何去计算一个句子真实标签序列的分数。

2.5 The score of all the possible paths

如何去计算所有可能路径的总分数，用一个一步步的例子来讲解。

C. CRF 层之于 BiLSTM

2.3 CRF loss function

CRF 的损失函数是由真实路径分数和所有可能路径的总分数构成。在那些所有可能路径中，这个真实路径应该具有最高得分。

例如，在我们的数据集中如果我们有那些标签，如下所示：

Label	Index
B-Person	0
I-Person	1
B-Organization	2
I-Organization	3
O	4
START	5
END	6

我们也有一个含有 5 个单词的句子。这些可能的路径将会是：

1) START B-Person B-Person B-Person B-Person B-Person END

2) START B-Person I-Person B-Person B-Person B-Person END

...

10) START B-Person I-Person O B-Organization O END

...

N) O O O O O O O

假设每一个可能的路径有一个分数 Pi，并且这里总共有 N 种可能的路径，这些路径的总分数是

$$P_{total} = P_1 + P_2 + \ldots + P_N = e^{S_1} + e^{S_2} + \ldots + e^{S_N}$$

e 是一个数学上的常量 e. (在 2.4 部分中，我们将会解释如何计算 Si，你也可以把它看成是一个路径的分数)

如果我们说 10th 路径是真实的标签路径，换句话说, the 10th path 是由训练集提供的标签。分数 P10 它就应该是在所有可能路径中有最大比例的。

下面给出的式子同时也是损失函数，在训练过程中，我们 BiLSTM-CRF 模型的参数值将会被不断的更新，为了保障真实路径的分数所占比例不断的增加。

$$LossFunction = \frac{PRealPath}{P_1 + P_2 + \ldots + P_N}$$

现在，问题是：

1) 如何去定义一个路径的分数?

2) 如何去计算所有路径的总分数?

3) 当我们计算总分数的时候，我们是否需要去罗列所有可能的路径？(这个问题的答案是NO.)

在接下来的小结，我们将会看到如何解决这些问题。

接下来

2.4 Real path score

如何去计算一个句子真实标签的分数

2.5 The score of all the possible paths

如何去计算所有可能路径的总分数，用一个一步步的例子来讲解。

D. CRF 层之于 BiLSTM

在 2.3 小结中，我们假设每一个可能的路径有一个分数 Pi 并且总共有 N 个可能的路径，所有路径的总分数是

$$P_{total} = P_1 + P_2 + \ldots + P_N = e^{S_1} + e^{S_2} + \ldots + e^{S_N}$$

e 是数学常量 e.

很明显，在所有可能的路径中肯定有一个真实的路径。例如，在 1.2 小节中，真实路径是“START B-Person I-Person O B-Organization O END”。其它是不正确的路径，例如“START B-Person B-Organization O I-Person I-Person B-Person”.

e^{Si} 是 i^{th} 路径的分数。

在训练过程中，CRF 损失函数仅需要两个分数：真正路径的分数和所有可能路径的总分数。真正路径分数在所有可能路径分数中的比例将会被逐渐地增加。

真正路径分数的计算，e^{Si} 是分成直接的。

这里我们关注于 Si 的计算。

拿我们之前的采样的真实路径“START B-Person I-Person O B-Organization O END”，例如：

- 我们有一个句子含有 5 个单词，w1,w2,w3,w4,w5
- 我们添加两个额外的单词，它们指明一个句子的开始和结束，w0,w6
- Si 由两部分组成 Si=EmissionScore+TransitionScore (The emission and transition score 已经在 2.1 和 2.2 小节中解释过)

Emission Score:

EmissionScore= $x_{0,\text{START}} + x_{1,\text{B-Person}} + x_{2,\text{I-Person}} + x_{3,\text{O}} + x_{4,\text{B-Organization}} + x_{5,\text{O}} + x_{6,\text{END}}$

- ✓ $x_{\text{index,label}}$ 是分数如果第 index 个词被标签为 label

✓ 这些分数 $x_{0,\text{START}}$、$x_{1,\text{B-Person}}$、$x_{2,\text{I-Person}}$、$x_{3,\text{O}}$、$x_{4,\text{B-Organization}}$、 $x_{5,\text{O}}$ 来自于前面的 BiLSTM 输出

✓ 像 $x_{0,\text{START}}$、 $x_{6,\text{END}}$ 我们可以就设置它们为 0

Transition Score:

TransitionScore= $t_{\text{START->B-Person}}$ + $t_{\text{B-Person->I-Person}}$ + $t_{\text{I-Person->O}}$ + $t_{\text{O->B-Organization}}$ + $t_{\text{B-Organization->O}}$ + $t_{\text{O->END}}$

✓ $t_{\text{label1->label2}}$ 是从 label1 到 label2 的转移分数

✓ 这些分数来自于 CRF 层。换句话说，这些转移分数是真正的 CRF 层的参数

来总结一下，现在我们可以计算路径 Si 和路径的分数 e^{Si}. 下一步是如何去计算所有路径的总分数。

接下来

2.5 The total score of all the possible paths

如何去计算所有可能路径的总分数，用一个一步步的例子来讲解。

这部分将会是一个最重要的并且有一点难度的部分。但是不用担心。这节中会给一个例子来尽量简单的解释细节。

E. CRF 层之于 BiLSTM

2.5 The total score of all the paths

在上一部分，我们学习了如何去计算一个真实标签路径的分数 e^{Si}。到目前为止，我们还有一个问题需要去解决，就是如何获得所有路径的总分数：

$$P_{total} = P_1 + P_2 + \ldots + P_N = e^{S1} + e^{S2} + \ldots + e^{SN}$$

最简单的方法去测量总分数是这样：枚举所有可能路径，并加和它们的分数。是的，你可以以这种方式计算总分数。可是，它是非常低效的。训练时间将无法忍受。

在你们探索接下来的内容之前，我建议你们准备一张白纸和一支笔，并且跟上例子中罗列的每一步。我保证这讲帮助你去更好的理解算法细节。进而，你应该知道如何去实现它用你们喜欢的编程语言来作。

第 1 步：回想 CRF 损失函数

在 2.3 小节中，我们定义 CRF 损失函数如下:

$$LossFunction = \frac{P_{RealPath}}{P_1 + P_2 + \ldots + P_N}$$

现在我们改变这个损失函数为一个对数损失函数：

$$LogLossFunction = \log \frac{P_{RealPath}}{P_1+P_2+\ldots+P_N}$$

因为当我们训练一个模型，通常我们的目标是去最小化我们的损失函数，我们增加一个负号：

$$\begin{aligned} & LogLossFunction \\ & = -\log \frac{P_{RealPath}}{P_1+P_2+\ldots+P_N} \\ & = -\log \frac{e^{S_{RealPath}}}{e^{S_1}+e^{S_2}+\ldots+e^{S_N}} \\ & = -(\log(e^{S_{RealPath}}) - \log(e^{S_1}+e^{S_2}+\ldots+e^{S_N})) \\ & = -(S_{RealPath} - \log(e^{S_1}+e^{S_2}+\ldots+e^{S_N})) \\ & = -(\sum_{i=1}^{N} x_{iyi} + \sum_{i=1}^{N-1} t_{yiyi+1} - \log(e^{S_1}+e^{S_2}+\ldots+e^{S_N})) \end{aligned}$$

在之前的小节，我们已经知道如何去计算真实的路径分数，现在我们需要去找到一个有效的解决方法去计算

$$\log(e^{S_1}+e^{S_2}+\ldots+e^{S_N})$$

第 2 步：回想 Emission and Transition Score

为了简化，我们假设我们正在利用一个长度仅为 3 的句子来训练我们的模型

x=[w0,w1,w2]

进而，在我们的数据集中，我们有两个标签：

LabelSet={l1,l2}

我们同时有 Emission scores 由 Bi-LSTM 层输出获得就如在 2.1 小节中描述的：

	l_1	l_2
w_0	x_{01}	x_{02}
w_1	x_{11}	x_{12}
w_2	x_{21}	x_{22}

xij 代表 wi 被标记为 lj 的分数。

进而，Transition scores 由 CRF 层而来就如小节 2.2 描述的:

	l_1	l_2
l_1	t_{11}	t_{12}
l_2	t_{21}	t_{22}

tij 是 labeli 到 labelj 的标签转移分数。

第 3 步：开始战斗! (纸和笔准备好)

记住：我们的目标是 $\log(e^{S_1} + e^{S_2} + \ldots + e^{S_N})$

这个过程是一个分数的累加。思想类似于动态规划(如果你不知道动态规划是什么，你也可以继续往下。我将会一步步来解释这个例子。但是我强烈推荐你去学习动态规划算法)。简言之, w0 的所有路径的总分数被计算。之后，我们利用这个总分数去计算 w0→w1。最后，我们利用上一个的总分数去计算 w0→w1→w2。这最后的总分数就是我们想要的。

在接下来的步骤，你将会看到两个变量：obs 和 previous。previous 存储之前步骤的最后结果。obs 指明当前单词的信息。

w0:

obs=[x01,x02]
previous=None

如果我们的句子仅有一个单词 w0，我们没有之前步骤的结果，因此 previous 是空。另外，我们仅能观测第一个单词的发射分数是 obs=[x01,x02]。x01 和 x02 是上面提到的发射分数。

你也许思考，什么是 w0 的所有路径的总分数?答案非常简单它是：

$$TotalScore(w_0) = \log(e^{x01} + e^{x02})$$

w0→w1:

obs=[x11,x12]
previous=[x01,x02]
(小心并且紧跟上)

1) 扩展 previous 为:

$$previous = \begin{pmatrix} x_{01} & x_{01} \\ x_{02} & x_{02} \end{pmatrix}$$

2) 扩展 obs 为:

$$obs = \begin{pmatrix} x_{11} & x_{12} \\ x_{11} & x_{12} \end{pmatrix}$$

你也许想知道，为什么我们需要去扩展 previous 和 obs 为矩阵。因为矩阵可以使得总分数的计算高效。你将在接下来的步骤中看到这个。

3) 加和 previous, obs 和 transition scores:

$$scores = \begin{pmatrix} x_{01} & x_{01} \\ x_{02} & x_{02} \end{pmatrix} + \begin{pmatrix} x_{11} & x_{12} \\ x_{11} & x_{12} \end{pmatrix} + \begin{pmatrix} t_{11} & t_{12} \\ t_{21} & t_{22} \end{pmatrix}$$

那么：

$$scores = \begin{pmatrix} x_{01} + x_{11} + t_{11} & x_{01} + x_{12} + t_{12} \\ x_{02} + x_{11} + t_{21} & x_{02} + x_{12} + t_{22} \end{pmatrix}$$

为了下一次迭代改变 previous 的值：

$$previous = [\log(e^{x_{01}+x_{11}+t_{11}} + e^{x_{02}+x_{11}+t_{21}}), \log(e^{x_{01}+x_{12}+t_{12}} + e^{x_{02}+x_{12}+t_{22}})]$$

事实上，第二次迭代已经完成。以防万一，有些人想知道从 w0 到 w1 如何去计算所有路径的总分数 (label1 →label1,label1 →label2,label2 →label1,label2 →label2) ,你可以如下计算：

我们使用新的 previous 中的元素：

$$\begin{aligned} &TotalScore(w_0 \to w_1) \\ &= \log(e^{previous[0]} + e^{previous[1]}) \\ &= \log(e^{\log(e^{x_{01}+x_{11}+t_{11}} + e^{x_{02}+x_{11}+t_{21}})} + e^{\log(e^{x_{01}+x_{12}+t_{12}} + e^{x_{02}+x_{12}+t_{22}})}) \\ &= \log(e^{x_{01}+x_{11}+t_{11}} + e^{x_{02}+x_{11}+t_{21}} + e^{x_{01}+x_{12}+t_{12}} + e^{x_{02}+x_{12}+t_{22}}) \end{aligned}$$

你发现一些了吗？是的，这就是我们的目标：

$$\log(e^{S_1} + e^{S_2} + \ldots + e^{S_N})$$

在等式中，我们可以看到：

- $S_1 = x_{01} + x_{11} + t_{11}$ $(label_1 \to label_1)$
- $S_2 = x_{02} + x_{11} + t_{21}$ $(label_2 \to label_1)$
- $S_3 = x_{01} + x_{12} + t_{12}$ $(label_1 \to label_2)$
- $S_4 = x_{02} + x_{12} + t_{22}$ $(label_2 \to label_2)$

w0→w1→w2:

学到这里基本上就完成了。事实上，在这次迭代，我们将会作和上一次迭代一样的步骤。

$$obs = [x_{21}, x_{22}]$$

$$previous = [\log(e^{x_{01}+x_{11}+t_{11}} + e^{x_{02}+x_{11}+t_{21}}), \log(e^{x_{01}+x_{12}+t_{12}} + e^{x_{02}+x_{12}+t_{22}})]$$

1) 扩展 previous 为:

$$previous = \begin{pmatrix} \log(e^{x01+x11+t11} + e^{x02+x11+t21}) & \log(e^{x01+x11+t11} + e^{x02+x11+t21}) \\ \log(e^{x01+x12+t12} + e^{x02+x12+t22}) & \log(e^{x01+x12+t12} + e^{x02+x12+t22}) \end{pmatrix}$$

2) 扩展 obs 为:

$$obs = \begin{pmatrix} x_{21} & x_{22} \\ x_{21} & x_{22} \end{pmatrix}$$

3) 加和 previous, obs 和 transition scores:

$$scores = \begin{pmatrix} \log(e^{x01+x11+t11} + e^{x02+x11+t21}) & \log(e^{x01+x11+t11} + e^{x02+x11+t21}) \\ \log(e^{x01+x12+t12} + e^{x02+x12+t22}) & \log(e^{x01+x12+t12} + e^{x02+x12+t22}) \end{pmatrix} + \begin{pmatrix} x_{21} & x_{22} \\ x_{21} & x_{22} \end{pmatrix} + \begin{pmatrix} t_{11} & t_{12} \\ t_{21} & t_{22} \end{pmatrix}$$

那么：

$$scores = \begin{pmatrix} \log(e^{x01+x11+t11} + e^{x02+x11+t21}) + x_{21} + t_{11} & \log(e^{x01+x11+t11} + e^{x02+x11+t21}) + x_{22} + t_{12} \\ \log(e^{x01+x12+t12} + e^{x02+x12+t22}) + x_{21} + t_{21} & \log(e^{x01+x12+t12} + e^{x02+x12+t22}) + x_{22} + t_{22} \end{pmatrix}$$

为了下一次迭代改变 previous 的值：

$$previous = [$$

$$\log(e^{\log(e^{x01+x11+t11}+e^{x02+x11+t21})+x21+t11} + e^{\log(e^{x01+x12+t12}+e^{x02+x12+t22})+x21+t21}),$$

$$\log(e^{\log(e^{x01+x11+t11}+e^{x02+x11+t21})+x22+t12} + e^{\log(e^{x01+x12+t12}+e^{x02+x12+t22})+x22+t22})$$

$$]$$

$$= [\log((e^{x01+x11+t11} + e^{x02+x11+t21})e^{x21+t11} + (e^{x01+x12+t12} + e^{x02+x12+t22})e^{x21+t21}),$$
$$\log((e^{x01+x11+t11} + e^{x02+x11+t21})e^{x22+t12} + (e^{x01+x12+t12} + e^{x02+x12+t22})e^{x22+t22})]$$

我们使用新的 previous 中的元素：

$$TotalScore(w_0 \to w_1 \to w_2)$$
$$= \log(e^{previous[0]} + e^{previous[1]})$$
$$= \log(e^{\log((e^{x01+x11+t11}+e^{x02+x11+t21})e^{x21+t11}+(e^{x01+x12+t12}+e^{x02+x12+t22})e^{x21+t21})}$$
$$+e^{\log((e^{x01+x11+t11}+e^{x02+x11+t21})e^{x22+t12}+(e^{x01+x12+t12}+e^{x02+x12+t22})e^{x22+t22})})$$

$$= \log(e^{x01+x11+t11+x21+t11} + e^{x02+x11+t21+x21+t11}$$
$$+e^{x01+x12+t12+x21+t21} + e^{x02+x12+t22+x21+t21}$$
$$+e^{x01+x11+t11+x22+t12} + e^{x02+x11+t21+x22+t12}$$
$$+e^{x01+x12+t12+x22+t22} + e^{x02+x12+t22+x22+t22})$$

恭喜!

我们实现了目标，$\log(e^{S_1} + e^{S_2} + \ldots + e^{S_N})$

在我们的句子中有 3 个单词和在我们的标签集中有 2 个标签，因此，这里应该总共有 8 个可能的路径。

在你去休息享用咖啡和甜品之前，请允许我说一下，尽管你发现过程十分复杂，但是算法的实现将会非常非常简单。计算机的一个优势就是作重复的工作。

现在你可以实现 CRF 损失函数并开始训练你自己的模型。

接下来

2.6 Infer the labels for a new sentence

我们已经学习了 CRF 损失函数的细节，下一步是当我们应用我们模型到测试集上的时候，如何对于一个新的句子推理计算标签。

F. CRF 层之于 BiLSTM

2.6 Infer the labels for a new sentence

在之前的小节中，我们学习了 BiLSTM-CRF 模型的结构和 CRF 损失函数的细节。你可以实现你自己的 BiLSTM-CRF 模型用各种开源框架(Keras, PyTorch, TensorFlow etc.). 一个最重要的事情是你的模型反向传播的时候这些框架可以自动计算梯度 ,因此你不需要训练模型而自己实现反向传播(i.e. 计算梯度和更新参数). 而且, 有些框架已经实现 CRF 层, 所以用你的模型整合一个 CRF 层将会非常简单，仅仅添加一行代码。

在这一节 ,我们将探索如何去推理测试集中一个句子的标签当我们的模型已经有了的时候。

第 1 步：从 BiLSTM-CRF 模型得到 Emission and transition scores

我们有一个句子包含三个单词: x=[w0,w1,w2]

进而，我们已经从 BiLSTM 模型得到了发射分数，已经从 CRF 层得到转移分数，如下：

	l_1	l_2
w_0	x_{01}	x_{02}
w_1	x_{11}	x_{12}
w_2	x_{21}	x_{22}

xij 指明 wi 被标记为 lj 的分数

	l_1	l_2
l_1	t_{11}	t_{12}
l_2	t_{21}	t_{22}

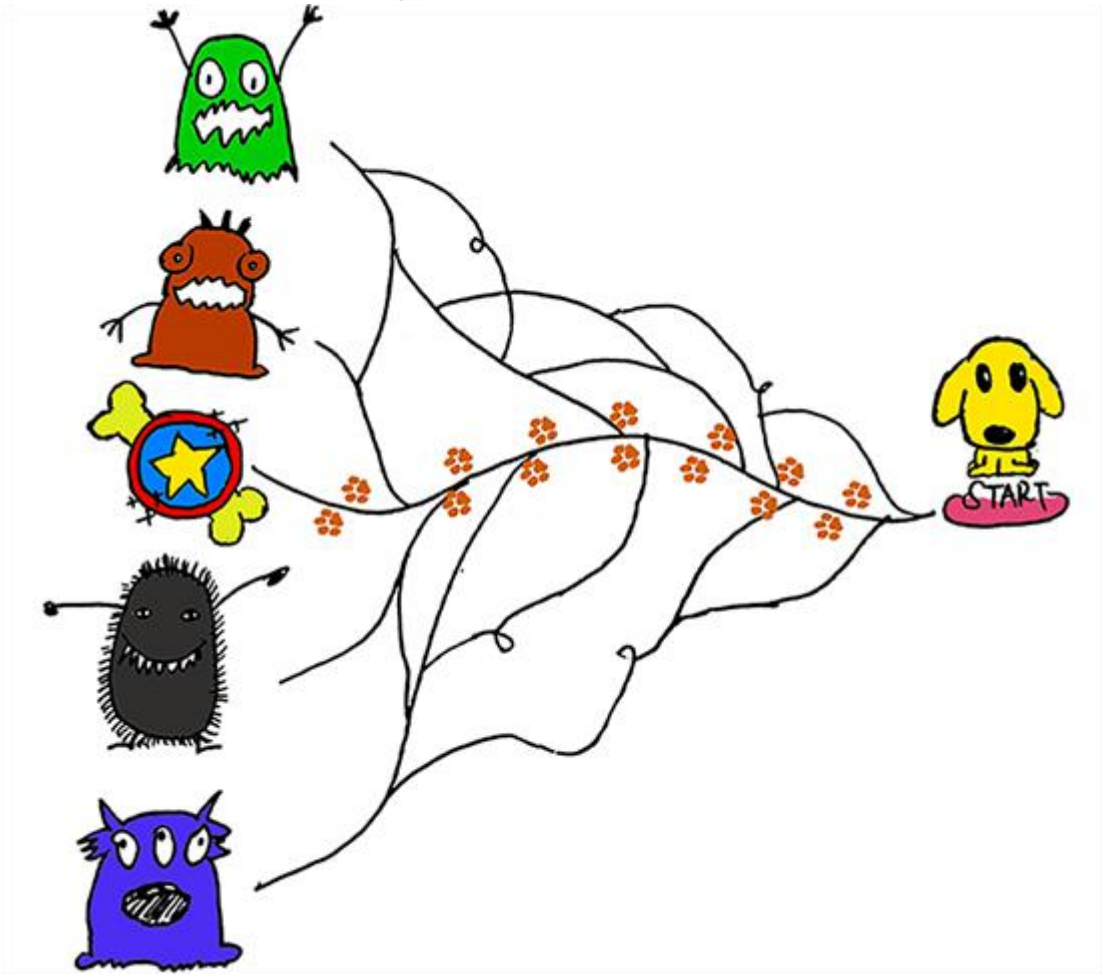
tij 是 label i 到 label j 的标签转移分数

第 2 步：开始推理

如果你熟悉维特比算法，这个小节将会对你来说很简单。但是如果你不会，请不用担心。和之前的小节类似，我会一步步解释算法。我们将运行从左到右对句子推理算法，如下：
w0
w0 → w1
w0 → w1 → w2

你将会看到两个变量：obs 和 previous。Previous 保存之前步骤的最后结果。Obs 指明当前词的信息。

alpha0 是最好分数的历史记录，alpha1 是对应索引的历史纪录。 这两个变量的细节将会在它们出现的解释。请看下面的图：你可以把这两个变量看成是狗探索森林时沿着路做的标记，这些标记将会有助于它找到回家的路。



狗需要找到最优路径去得到它喜爱的骨头玩具，并且沿着来时的路返回家。

w0:

obs=[x01,x02]

previous=None

现在，我们正在观测第一个单词 w0。到目前位置对于 w0 最好的标签显而易见。

例如，如果 obs=[x01=0.2,x02=0.8]，很明显最好的标签对于 w0 应该是 I2

因为这里仅有一个单词并且没有标签之间的转移，转移分数没有用到。

w0→w1:

obs=[x11,x12]

previous=[x01,x02]

(小心并且紧跟上)

1) 扩展 previous 为:

$$previous = \begin{pmatrix} previous[0] & previous[0] \\ previous[1] & previous[1] \end{pmatrix} = \begin{pmatrix} x_{01} & x_{01} \\ x_{02} & x_{02} \end{pmatrix}$$

2) 扩展 obs 为:

$$obs = \begin{pmatrix} obs[0] & obs[1] \\ obs[0] & obs[1] \end{pmatrix} = \begin{pmatrix} x_{11} & x_{12} \\ x_{11} & x_{12} \end{pmatrix}$$

3) 加和 previous, obs 和 transition scores:

$$scores = \begin{pmatrix} x_{01} & x_{01} \\ x_{02} & x_{02} \end{pmatrix} + \begin{pmatrix} x_{11} & x_{12} \\ x_{11} & x_{12} \end{pmatrix} + \begin{pmatrix} t_{11} & t_{12} \\ t_{21} & t_{22} \end{pmatrix}$$

那么：

$$scores = \begin{pmatrix} x_{01} + x_{11} + t_{11} & x_{01} + x_{12} + t_{12} \\ x_{02} + x_{11} + t_{21} & x_{02} + x_{12} + t_{22} \end{pmatrix}$$

你或许发现这里和之前小节去计算所有路径的总分数没什么区别。请保持耐心并小心，你将会很快看到不同。

为了下一次迭代改变 previous 的值：

$$previous = [\max(scores[00], scores[10]), \max(scores[01], scores[11])]$$

例如，如果我们的分数是：

$$scores = \begin{pmatrix} x_{01} + x_{11} + t_{11} & x_{01} + x_{12} + t_{12} \\ x_{02} + x_{11} + t_{21} & x_{02} + x_{12} + t_{22} \end{pmatrix} = \begin{pmatrix} 0.2 & 0.3 \\ 0.5 & 0.4 \end{pmatrix}$$

为了下一次迭代，我们的 previous 将会是：

$$previous = [\max(scores[00], scores[10]), \max(scores[01], scores[11])] = [0.5, 0.4]$$

Previous 的含义是什么？Previous 列表存储着当前词的每一个标签的最大分数。

[例子开始]

例如:

我们知道在我们的语料库中，我们总共有 2 个标签，

label1(l1)和 label2(l2)。这两个标签的索引分别是 0 和 1 (因为在编程中，索引值由 0 开始而不是由 1 开始)

previous[0]是有着第 0th 个标签 l1 结束的路径的最大分数。相似的,

previous[1]是有着标签 l2 结束的路径的最大分数。在每次迭代中，变量 previous 存储路径由每一个标签结束的路径的最大分数。换言之，在每次迭代中，我们仅保留每个标签的最优路径信息(previous=[max(scores[00],scores[10]),max(scores[01],scores[11])])。更少的分数的路径信息将会被丢弃。

[例子结束]

回到我们主要的任务：

同时，我们也由两个变量去存储历史信息（分数和索引），alpha0 和 alpha1。对于这次迭代，我们将添加最好的分数到 alpha0。为了你方便，每个标签的最大分数加了下划线。

$$scores = \begin{pmatrix} x_{01} + x_{11} + t_{11} & x_{01} + x_{12} + t_{12} \\ \underline{x_{02} + x_{11} + t_{21}} & \underline{x_{02} + x_{12} + t_{22}} \end{pmatrix} = \begin{pmatrix} 0.2 & 0.3 \\ \underline{0.5} & \underline{0.4} \end{pmatrix}$$

$$alpha_0 = [(scores[10], scores[11])] = [(0.5, 0.4)]$$

另外，对应的列索引被保存在了 alpha1：

$$alpha_1 = [(ColumnIndex(scores[10]), ColumnIndex(scores[11]))] = [(1, 1)]$$

解释，l1 的索引是 0，l2 的索引是 1。所以(1,1)=(l2,l2)意味着，对于当前词 wi 和标签 li：

$(1, 1)$
$= (l_2, l_2)$
=(we can get the maximum score 0.5 when the path is $\underline{l^{(i-1)} = l_2} \rightarrow \underline{l^{(i)} = l_1}$),
we can get the maximum score 0.4 when the path is $\underline{l^{(i-1)} = l_2} \rightarrow \underline{l^{(i)} = l_2}$)

$l^{(i-1)}$ is the label of the previous word w_{i-1}.

w0→w1→w2:

obs=[x21,x22]
previous=[0.5,0.4]

1) 扩展 previous 为:

$$previous = \begin{pmatrix} previous[0] & previous[0] \\ previous[1] & previous[1] \end{pmatrix} = \begin{pmatrix} 0.5 & 0.5 \\ 0.4 & 0.4 \end{pmatrix}$$

2) 扩展 obs 为:

$$obs = \begin{pmatrix} obs[0] & obs[1] \\ obs[0] & obs[1] \end{pmatrix} = \begin{pmatrix} x_{21} & x_{22} \\ x_{21} & x_{22} \end{pmatrix}$$

3) 加和 previous, obs 和 transition scores:

$$scores = \begin{pmatrix} 0.5 & 0.5 \\ 0.4 & 0.4 \end{pmatrix} + \begin{pmatrix} x_{21} & x_{22} \\ x_{21} & x_{22} \end{pmatrix} + \begin{pmatrix} t_{11} & t_{12} \\ t_{21} & t_{22} \end{pmatrix}$$

那么：

$$scores = \begin{pmatrix} 0.5 + x_{11} + t_{11} & 0.5 + x_{12} + t_{12} \\ 0.4 + x_{11} + t_{21} & 0.4 + x_{12} + t_{22} \end{pmatrix}$$

为了下一次迭代改变 previous 的值：

$$previous = [\max(scores[00], scores[10]), \max(scores[01], scores[11])]$$

我们得到这次迭代的分数：

$$scores = \begin{pmatrix} 0.6 & \underline{0.9} \\ \underline{0.8} & 0.7 \end{pmatrix}$$

因此，我们得到最近的 previous：

previous=[0.8,0.9]

事实上，在 previous[0]和 previous[1]之间更大的值是最优预测路径的分数。

同时，每个标签的最大分数和索引将会被添加进 alpha0 和 alpha1:

$$alpha_0 = [(0.5, 0.4), \underline{(scores[10], scores[01])}]$$

$$= [(0.5, 0.4), \underline{(0.8, 0.9)}]$$

$$alpha_1 = [(1, 1), \underline{(1, 0)}]$$

第 3 步：找出具有最高分值的最优路径

这是最后一步！你将要完成了！在这步中，alpha0 和 alpha1 将会被使用去找分数最高的路径。我们将从最后一个到开始第一个，检查在这两个列表中的元素。

w1→w2:

首先，检测 alpha0 和 alpha1 的最后一个元素:(0.8,0.9)和(1,0)。0.9 意味着标签是 l2,我们可以得到最高路径分数 0.9。我们也知道路径 l2 的索引是 1，因此检查(1,0)[1]=0 中的值。索引 “0” 意味着前一个标签是 l1(l1 的索引是 0)。所以我们得到最优路径是 w1 → w2:是 l1 → l2

w0→w1:

其次，我们继续反向移动并且得到 alpha1 的元素:(1,1)。从上一个段落我们知道 w1 的标签是 l1(索引是 0)，因为我们可以检查(1,1)[0]=1。所以我们可以得到最优(w0 → w1:): l2 → l1

恭喜！对于我们的例子我们得到的最优路径是 l2 → l1 → l2