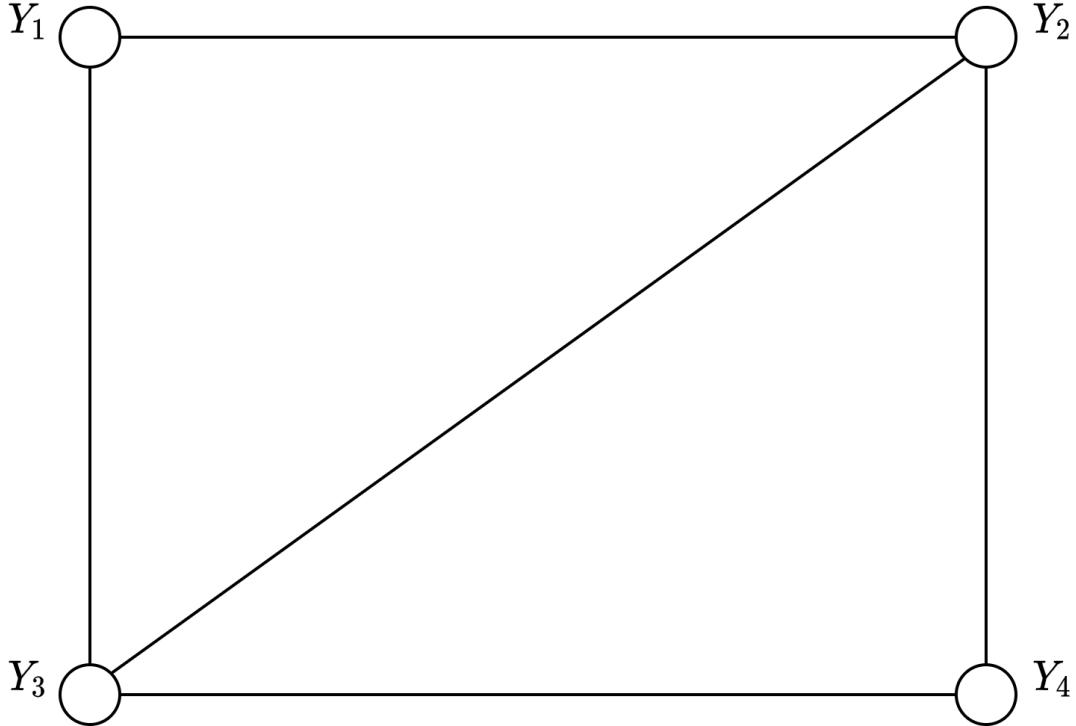


## 一、CRF简介：

### 1.1. CRF基础原理：

- 无向图的团和最大团：

- 无向图： $G = (V, E)$ ,  $V$ 表示节点集合,  $E$ 表示边的集合
- 完全图：每个节点之间均有连接的无向图
- 完全子图：给定无向图 $G = (V, E)$ , 如果节点结合 $U \subseteq V$ , 且任意节点 $u, v \subseteq U$ 有 $(u, v) \subseteq E$ , 则称 $U$ 是 $G$ 的完全子图
- 团：最大完全子图，理解为孩子图每个节点互相均有连接，且孩子图的原图也包含这些节点和边
- 最大团：在一个无向图中找出一个点数最多的完全子图



- 推理：（依据上图）

- ① 已知 $Y_1$ 和 $Y_4$ 不相连，所以对于 $Y_2$ 和 $Y_3$ 的条件下，他们相互独立： $P(Y_1, Y_4 | Y_2, Y_3) = P(Y_1 | Y_2, Y_3) * P(Y_4 | Y_2, Y_3)$
- ② 又条件概率定理可知 $P(A|B)P(B) = P(AB)$ , 所以 $P(Y_1, Y_2, Y_3, Y_4) = P(Y_1, Y_4 | Y_2, Y_3) * P(Y_2, Y_3)$
- ③  $P(Y_1, Y_4 | Y_2, Y_3) * P(Y_2, Y_3) = P(Y_1, Y_4 | Y_2, Y_3) * P(Y_2, Y_3) * P(Y_2, Y_3) / P(Y_2, Y_3)$   
 $P(Y_1, Y_4 | Y_2, Y_3) * P(Y_2, Y_3) * P(Y_2, Y_3) / P(Y_2, Y_3)$
- ④ 由①可知，  
 $= P(Y_1 | Y_2, Y_3) * P(Y_4 | Y_2, Y_3) * [P(Y_2, Y_3) * P(Y_2, Y_3) / P(Y_2, Y_3)]$   
 $= [P(Y_1 | Y_2, Y_3) * P(Y_2, Y_3)] * [P(Y_4 | Y_2, Y_3) * P(Y_2, Y_3)] / P(Y_2, Y_3)$
- ⑤ 同②条件概率定理可知，  
 $[P(Y_1 | Y_2, Y_3) * P(Y_2, Y_3)] * [P(Y_4 | Y_2, Y_3) * P(Y_2, Y_3)] / P(Y_2, Y_3)$   
 $= P(Y_1, Y_2, Y_3) * P(Y_4, Y_2, Y_3) / P(Y_2, Y_3)$
- ⑥ 所以  $P(Y_1, Y_2, Y_3) * P(Y_4, Y_2, Y_3) / P(Y_2, Y_3) = P(Y_1, Y_2, Y_3, Y_4)$
- ⑦ 根据上图和最大团的定义可知， $P(Y_1, Y_2, Y_3)$ 和 $P(Y_4, Y_2, Y_3)$ 是 $P(Y_1, Y_2, Y_3, Y_4)$ 的最大团
- ⑧ 因为 $P(B) = \sum P(A|B)$ 即发生的条件下，所有A事件发生概率之和，所以 $P(Y_2, Y_3) = \sum P(Y_1, Y_4 | Y_2, Y_3)$
- ⑨ 由①可知 $Y_1$ 和 $Y_2$ 相互独立，所以 $\sum P(Y_1, Y_4 | Y_2, Y_3) = \sum P(Y_1 | Y_2, Y_3) * P(Y_4 | Y_2, Y_3)$
- ⑩ 可以容易看出 $Y_2$ 和 $Y_3$ 发生时，所有 $Y_1$ 事件概率之和， $\sum P(Y_1 | Y_2, Y_3) = \sum P(Y_1 Y_2 Y_3)$ ,  $\sum P(Y_4 | Y_2, Y_3)$ 也同理，所以  
 $\sum P(Y_1 | Y_2, Y_3) * P(Y_4 | Y_2, Y_3) = \sum P(Y_1 Y_2 Y_3) * P(Y_4 Y_2 Y_3)$
- 可证： $P(Y_1, Y_2, Y_3, Y_4) = P(Y_1, Y_2, Y_3) * P(Y_4, Y_2, Y_3) / \sum P(Y_1 Y_2 Y_3) * P(Y_4 Y_2 Y_3)$
- 推广：

- 如果想要求无向图的联合概率，就需要用无向图中所有最大团概率乘积，除以最大团所有可能的概率的乘积之和
- $P(Y_1, Y_2, Y_3, Y_4) = \frac{P(Y_1, Y_2, Y_3) * P(Y_4, Y_2, Y_3)}{\sum P(Y_1 Y_2 Y_3) * P(Y_4 Y_2 Y_3)} = \frac{\Phi_1 \Phi_2}{\sum \Phi_1 \Phi_2}$ ,  $\Phi$ 正比于概率 $P(Y_1, Y_2, Y_3, Y_4)$ ，不一定和概率相等

- 案例：

- 条件：假设 $Y_1, Y_2, Y_3, Y_4 \in \{1, 2\}$ 
  - $P(Y_1 = 2, Y_2 = 1, Y_3 = 1) = 0.2$
  - $P(Y_4 = 1, Y_2 = 1, Y_3 = 1) = 0.15$
  - $P(Y_2 = 1, Y_3 = 1) = 0.5$

$$P(Y_1 = 2, Y_2 = 1, Y_3 = 1, Y_4 = 1)$$

- 结果:  $= P(Y_1 = 2, Y_2 = 1, Y_3 = 1) * P(Y_4 = 1, Y_2 = 1, Y_3 = 1) / P(Y_2 = 1, Y_3 = 1)$   
 $= 0.2 * 0.12 / 0.5$

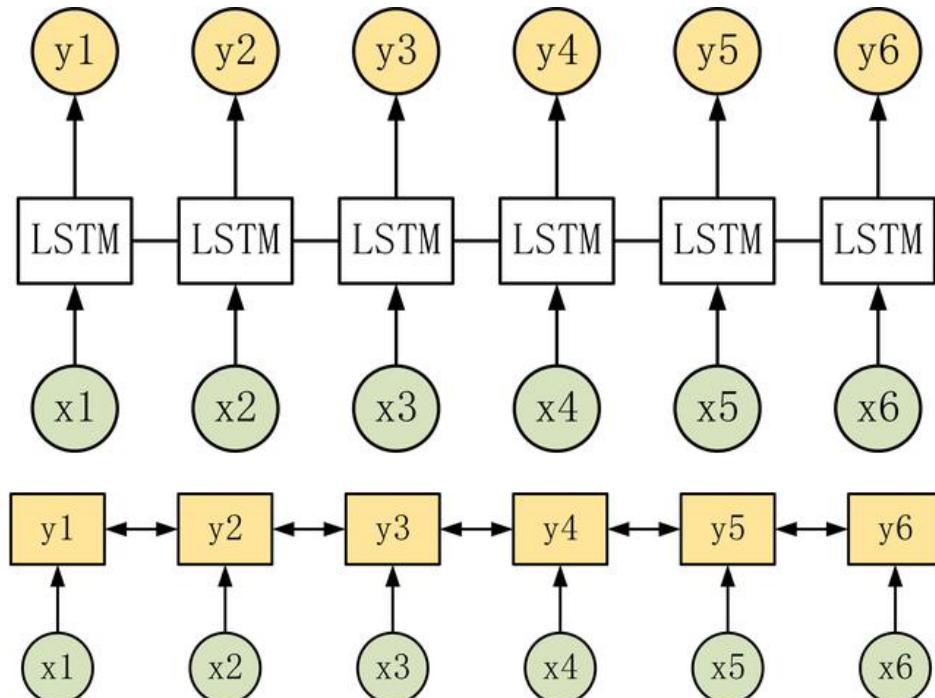
- 概念梳理:

- 随机场: 随机场是一种图模型, 包含结点的集合和边的集合, 结点表示一个随机变量, 而边表示随机变量之间的依赖关系。如果按照某一种分布随机给图中每一个结点赋予一个值, 则称为随机场。
- 马尔科夫随机场: 马尔科夫性质指某一个时刻  $t$  的输出值只和  $t - 1$  时刻的输出有关系, 和更早的输出没有关系。马尔科夫随机场则是一种特殊的随机场, 其假设每一个结点的取值只和相邻的结点有关系, 和不相邻结点无关。
- 条件随机场 CRF: CRF 是一种特殊的马尔科夫随机场, CRF 假设模型中只有  $\mathbf{X}$  (观测值) 和  $\mathbf{Y}$  (状态值)。在 CRF 中每一个状态值  $y_i$  只和其相邻的状态值有关, 而观测值  $x$  不具有马尔科夫性质。注意观测序列  $\mathbf{X}$  是作为一个整体影响  $\mathbf{Y}$  计算。
- 线性链条件随机场 Linear-chain CRF: 线性链条件随机场指序列  $\mathbf{Y}$  和  $\mathbf{X}$  都是线性链的条件随机场。

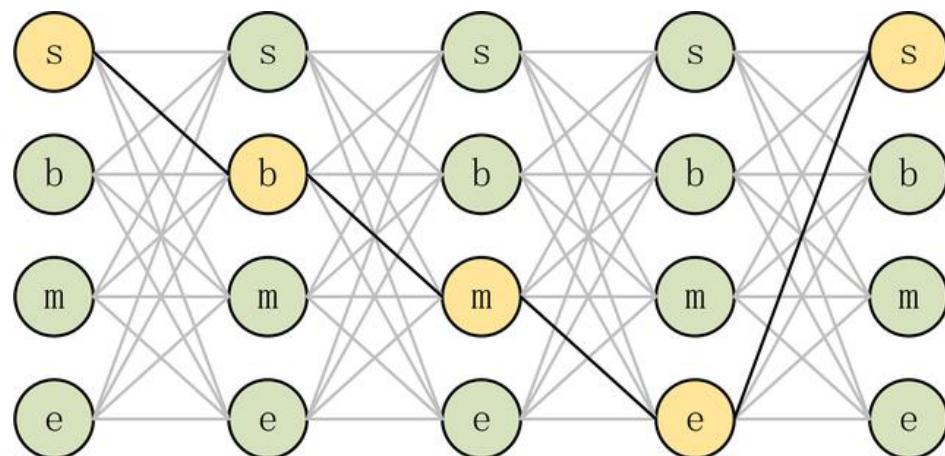
- 和其他序列方法对比:

- 对比LSTM:

- LSTM 和 CRF 都可以进行序列标注, 用 LSTM 进行序列标注的时候就是对于每一个时刻的输入进行分类; 但是 LSTM 不能考虑输出的每一个元素的相关性。(例如在分词时,  $s$  表示单个字的词, 则  $s$  之后是不能接  $m$  或者  $e$  的, 而 LSTM 逐元素分类是不能考虑这种相关性的)
- 但是 CRF 可以考虑输出元素的前后关联性(CRF 可以通过特征函数学习状态间的关联)



- CRF 会计算出一个输出序列的分数, 并用所有可能序列的分数之和进行归一化, 如下图所示,  $sbmes$  就是其中一个序列路径, CRF 会找出概率最大的路径作为预测序列。



- 对比HMM:

- HMM是生成模型, CRF是判别模型
- HMM是概率有向图, CRF是概率无向图
- HMM求解过程可能是局部最优, CRF可以全局最优
- CRF概率归一化较合理, HMM则会导致label bias 问题

- 隐形马尔科夫 (HMM) :

- HMM针对问题:

- 一类可观察数据( $v$ )、一类不可观察数据( $q$ )
- 基于序列的问题(时间、状态序列)

#### ■ HMM假设：

- 齐次马尔科夫链假设：任意时刻的隐藏状态( $q_i$ )只依赖于它前一个隐藏状态( $q_{i-1}$ )
- 观察独立性假设：任意时刻的观察状态( $v_i$ )只仅仅依赖于当前时刻的隐藏状态( $q_i$ )
- 任务案例：输入单词序列，输出词性序列
  - HMM假设了两类特征，一是当前词词性与上一个词词性的关系，以及当前词语和当前词性的关系，分别对应着转移概率和发射概率。HMM的学习过程就是在训练集中学习这两个概率矩阵

- 更多细节详见附件：《HMM与CRF.pdf》和《李宏毅crf.pdf》

#### 1.2. 代码实现：

- 更好的参考：<https://www.kaggle.com/xuyouqian/ner-bilstm-crf>

```
import numpy as np

class CRF(object):
    """ 实现条件随机场预测问题的维特比算法 """
    def __init__(self, V, VW, E, EW):
        """
        :param V: 是定义在节点上的特征函数，称为状态特征
        :param VW: 是V对应的权值
        :param E: 是定义在边上的特征函数，称为转移特征
        :param EW: 是E对应的权值
        """

        self.V = V # 点分布表
        self.VW = VW # 点权值表
        self.E = E # 边分布表
        self.EW = EW # 边权值表
        self.D = [] # Delta表，最大非规范化概率的局部状态路径概率（回溯时，从前置最优路径到该位置的取值）
        self.P = [] # Psi表，当前状态和最优前导状态的索引表s（回溯最优路径的索引）
        self.BP = [] # BestPath，最优路径
        return

    def viterbi(self):
        """
        条件随机场预测问题的维特比算法，此算法一定要结合CRF参数化形式对应的状态路径图来理解，更容易理解。
        """

        self.D = np.full(shape=(np.shape(self.V)), fill_value=.0)
        self.P = np.full(shape=(np.shape(self.V)), fill_value=.0)
        # 遍历长度为3的序列
        for i in range(np.shape(self.V)[0]):
            # 初始化
            if 0 == i:
                # 当前节点和权值的乘积
                self.D[i] = np.multiply(self.V[i], self.VW[i])
                # 因为初始化，从start到v，回溯索引是0
                self.P[i] = np.array([0, 0])
                print('self.V[%d]=' % i, self.V[i], 'self.VW[%d]=' % i, self.VW[i])
                print('self.D[%d]=' % i, self.D[i], 'self.P:', self.P)
                pass
            # 递推求解布局最优状态路径
            else:
                for y in range(np.shape(self.V)[1]): # delta[i][y=1,2...]
                    for l in range(np.shape(self.V)[1]): # V[i-1][l=1,2...]
                        delta = 0.0
                        delta += self.D[i - 1, l] # 前导状态的最优状态路径的概率
                        delta += self.E[i - 1][l, y] * self.EW[i - 1][l, y] # 前导状态到当前状态的转移概率
                        delta += self.V[i, y] * self.VW[i, y] # 当前状态的概率
                        print('(x%d,y%d)-->(x%d,y=%d):%.2f + %.2f + %.2f=' % (i - 1, l, i, y, \
                            self.D[i - 1, l], \
                            self.E[i - 1][l, y] * self.EW[i - 1][l, y], \
                            self.V[i, y] * self.VW[i, y]),
                        - 1)[
                            l, y], \
                            self.V[i, y] * self.VW[i, y]),
                        delta)
                # 如果还没有记录，或者是有各大的值，就重新给该位置赋值（更优路径）
                if 0 == l or delta > self.D[i, y]:
                    self.D[i, y] = delta
                    self.P[i, y] = l
                print('self.D[x%d,y=%d]=%.2f\n' % (i, y, self.D[i, y]))
            print('self.D:\n', self.D)
            print('self.P:\n', self.P)

        # 返回，得到所有的最优前导状态
        N = np.shape(self.V)[0]
        self.BP = np.full(shape=(N,), fill_value=0.0)
        # 从尾到扱回溯
```

```

t_range = -1 * np.array(sorted(-1 * np.arange(N)))
# 回溯找到最优
for t in t_range:
    if N - 1 == t: # 最后一个选取最优状态 (得分最高)
        self.BP[t] = np.argmax(self.D[-1])
    else: # 回溯得到最优前导状态
        self.BP[t] = self.P[t + 1, int(self.BP[t + 1])]

# 最优状态路径表现在存储的是状态的下标, 我们执行存储值+1转换成示例中的状态值
# 也可以不用转换, 只要你能理解, self.BP中存储的0是状态1就可以~~~~
self.BP += 1
print('最优状态路径为: ', self.BP)
return self.BP

def CRF_manual():
    # 已知节点的特征函数以及权值, 还有连接边的特征函数和权值
    V = np.array([[1, 1], # X1:S(Y1=1), S(Y1=2)
                  [1, 1], # X2:S(Y2=1), S(Y2=2)
                  [1, 1]]) # X3:S(Y3=1), S(Y3=1)
    VW = np.array([[1.0, 0.5], # X1:SW(Y1=1), SW(Y1=2)
                  [0.8, 0.5], # X2:SW(Y2=1), SW(Y2=2)
                  [0.8, 0.5]]) # X3:SW(Y3=1), SW(Y3=1)
    E = np.array([[[1, 1], # Edge:Y1=1-->(Y2=1, Y2=2)
                  [1, 0]], # Edge:Y1=2-->(Y2=1, Y2=2)
                  [[0, 1], # Edge:Y2=1-->(Y3=1, Y3=2)
                  [1, 1]]]) # Edge:Y2=2-->(Y3=1, Y3=2)
    EW = np.array([[[[0.6, 1], # Edgew:Y1=1-->(Y2=1, Y2=2)
                  [1, 0.0]], # Edgew:Y1=2-->(Y2=1, Y2=2)
                  [[0.0, 1], # Edgew:Y2=1-->(Y3=1, Y3=2)
                  [1, 0.2]]]]) # Edgew:Y2=2-->(Y3=1, Y3=2)
    crf = CRF(V, VW, E, EW)
    ret = crf.Viterbi()
    return

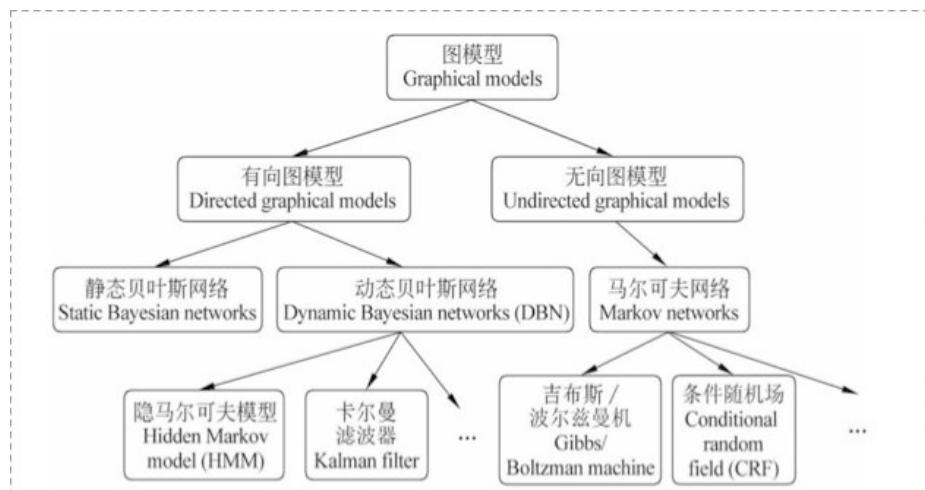
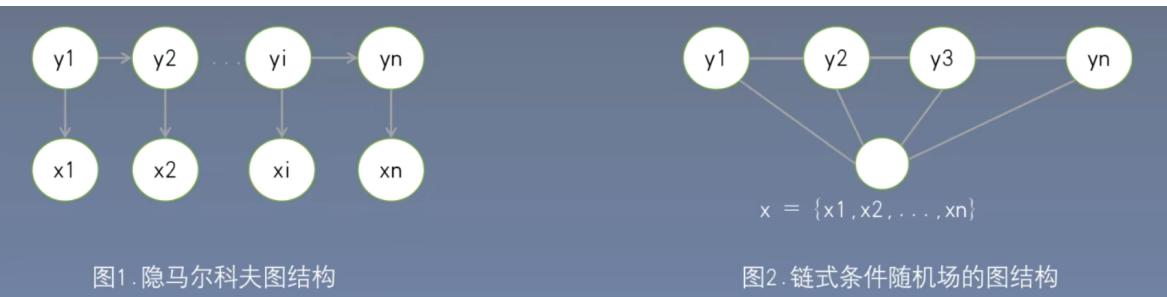
if __name__ == '__main__':
    CRF_manual()

```

## 二、BiLSTM + CRF原理:

### 2.1. 论文算法模型总览:

- **概率图模型**: 用图表达变量相关关系的概率模型, 通常用一个节点表示一个或一组随机变量, 用节点间的边来表示变量间的概率关系, 分为有向无环图 (贝叶斯网) + 无向图模型 (马尔可夫网)



- **HMM模型**: 命名实体标注中“BIO”的实体标签，是一个不可观测的隐状态，而HMM模型描述的就是这些隐状态（实体标记）生成可观测状态（可读文本）的过程

- **三个概率矩阵**: 由初始隐状态概率，转移概率和发射概率来模拟概率分布（详见pinard & 李宏毅）：

$$p(X, Y) = \text{Initial Probability} * \text{Transition Probability} * \text{Observation Probability} = p(x_1) * \prod_{t=1}^{T-1} p(x_{t+1}|x_t) * \prod_{t=1}^T p(y_t|x_t)$$

▪ **初始隐状态概率**:  $\pi = P(i_1 = q_i), q \in Q_{hidden} = \{q_0, q_1, \dots, q_{N-1}\}$

▪ **转移概率**:  $A_{i,j} = P(i_{t+1}|i_t = q_t), q \in Q_{hidden}$

▪ **发射概率**:  $B_{j,k} = P(o_t = v_k|i_t = q_j), q \in Q_{hidden}, v \in V_{obs} = \{v_0, v_1, \dots, v_{M-1}\}$

- **三个基本问题**: (详见hmm代码实现 + pinard)

▪ **概率计算问题**: 一致模型和观测序列，计算在该模型下观测序列出现的概率（前向/后向算法评估）

▪ **学习问题**: 已知观测序列，估计模型的参数（已知隐藏状态用最大似然，否则用鲍姆-韦尔奇算法）

▪ **预测问题**: 一致模型参数和观测序列，求最后可能的对应状态序列（维特比算法解码）

设我们的可观测状态序列是由所有汉字组成的集合，我们用 $V_{Observation}$ 来表示：

$$V_{obs.} = \{v_1, v_2, \dots, v_M\}$$

上式中， $v$ 表示字典中单个字，假设我们已知的字数为 $M$ 。

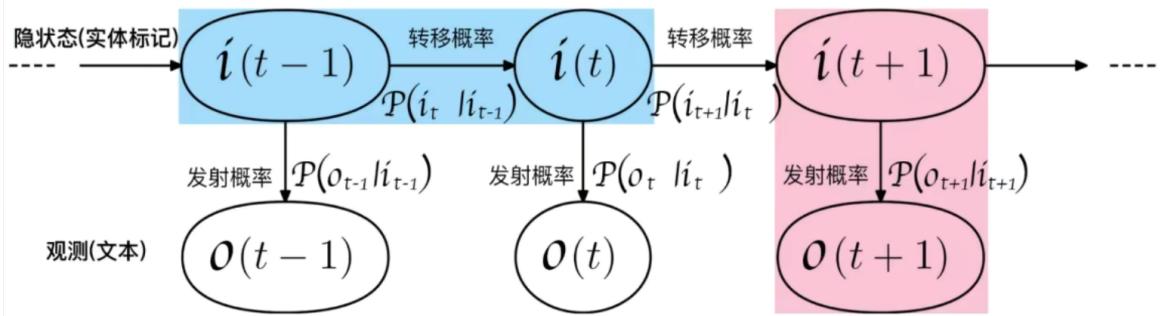
设所有可能的隐藏状态集合为 $Q_{hidden}$ 一共有 $N$ 种隐藏状态，例如我们现在的命名实体识别数据里面只有7种标签：

$$Q_{hidden} = \{q_1, q_2, \dots, q_N\}$$

设我们有观测到的一串自然语言序列文本 $O$ ，一共有 $T$ 个字，又有这段观测到的文本所对应的实体标记，也就是隐状态 $I$ :

$$I = \{i_1, i_2, \dots, i_T\} \text{(隐状态)} \quad O = \{o_1, o_2, \dots, o_T\} \text{(观测)}$$

注意上式中，我们常称 $t$ 为时刻，如上式中一共有 $T$ 个时刻( $T$ 个汉字)。



- **CRF模型**: 设两组随机变量 $X = (X_1, \dots, X_n)$ ,  $Y = (Y_1, \dots, Y_n)$ , 那么线性链条件随机场的定义为

$P(Y_i|X, Y_1, \dots, Y_{i-1}, Y_{i+1}, \dots, Y_n) = P(Y_i|X, Y_{i-1}, Y_{i+1})$ ,  $i = 1, \dots, n$ , 其中 $i$ 取1或 $n$ 时，只考虑单边；即已知 $X$ 和对应的 $Y_i$ 左边和右边的概率分布求 $Y_i$ ，CRF里设定此时 $Y_i$ 的值只和它左边 $Y_{i-1}$ 和右边 $Y_{i+1}$ 的概率分布，以及整体 $X$ 序列有关。

- **HMM**: HMM有向性，只和前一个 $Y_{i-1}$ 有关，详见下图1

- **MEMM**: 不同于HMM的方向， $Y$ 是由 $X$ 来决定的方向发生了转变 (MEMM)，详见下图2

- **CRF**: 不同于HMM的有向性， $X$ 和 $Y$ 互相影响，既和前一个 $Y_{i-1}$ 有关也和后一个 $Y_{i+1}$ 有关，且每个 $Y_i$ 和整个 $X$ 序列有关，详见下图3

▪ 从HMM预估参数的生成模型，变为由 $X$ 序列推断 $Y$ 的判别模型

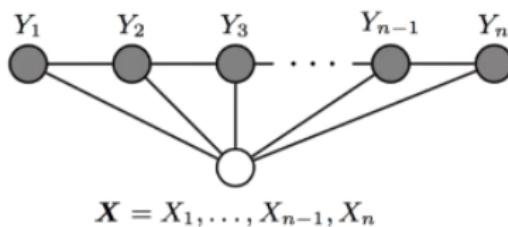


Figure 1: Graphical structure of a chain-structured CRFs for sequences. The variables corresponding to unshaded nodes are *not* generated by the model.

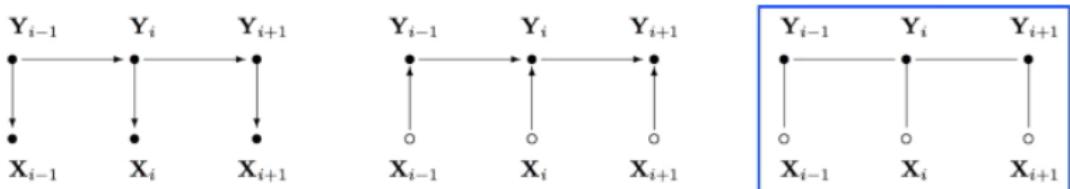


Figure 2. Graphical structures of simple HMMs (left), MEMMs (center), and the chain-structured case of CRFs (right) for sequences. An open circle indicates that the variable is not generated by the model.

- **特征函数**: 取值0或1，当满足规定好的特征条件时取值为1，否则为0

- **转移特征函数**:  $t_k(y_{i-1}, y_i, x, i)$ 定义在边上的特征函数，依赖于当前位置 $i$ 和前一个位置 $i - 1$ ，对应权值为 $\lambda_k$

- $t_3(y_{i-1}, y_i, x, i)$ : if  $(y_{i-1} = B - PER, y_i == I - LOC)$  1 else 0, 权重大, 表示倾向于认为B-LOC后面跟着I-LOC
- $t_4(y_{i-1}, y_i, x, i)$ : if  $(y_{i-1} = B - PER, y_i == I - PER)$  1 else 0, 权重小, 就表明倾向于认为B-LOC后面不会跟着I-PER
- 状态特征函数:  $s_l(y_i, x, i)$  定义在节点上的特征函数, 依赖于当前位置*i*, 对应权值为 $\mu_l$ 
  - $s_1(y_i, x, i)$ : if  $(y_i = B - PER, x_i == '张')$  1 else 0, 权重大, 表示倾向于将当前为“张”标注为人名的开始
  - $s_2(y_i, x, i)$ : if  $(y_i = I - DATE, x_{i+1} == '日')$  1 else 0, 正权重, 倾向于将“日”前面的字标注为日期的中间, 例如“12日3点”
- 参数化形式: 给定一个线性链条件随机链  $P(Y|X)$ , 当观测序列为  $x = x_1, x_2, \dots$  时, 状态序列为  $y = y_1, y_2, \dots$  的概率为  $P(Y = y|x)$ ,  $Z(x)$  为规范化因子, 将其转为概率
  - CRF可以类比为序列版本的逻辑回归, 是序列标注问题的对数线性模型 (将特征转化成概率)

$$P(Y = y|x) = \frac{1}{Z(x)} \exp\left(\sum_k \lambda_k \sum_i t_k(y_{i-1}, y_i, x, i) + \sum_l \mu_l \sum_i s_l(y_i, x, i)\right)$$

$$Z(x) = \sum_y \exp\left(\sum_k \lambda_k \sum_i t_k(y_{i-1}, y_i, x, i) + \sum_l \mu_l \sum_i s_l(y_i, x, i)\right)$$

- 对比HMM: 可以用CRF来表达HMM, 但HMM表达能力首先, 不能反过来表达CRF (详见下图)

- 不同于HMM, CRF可以以来全局的  $X$  的特征, 更加全面
- CRF可以由任意的权重值
- HMM概率值必须满足特定的约束 (概率和为1)

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

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

- 为每个HMM的转换概率  $p(l_i = y | l_{i-1} = x)$ , 定义一组转换形式为  $f_{x,y}(s, i, l_i, l_{i-1}) = 1$  if  $l_i = y$  且  $l_{i-1} = x$  的CRF转换特征。给定每个特征权重值为  $w_{x,y} = \log p(l_i = y | l_{i-1} = x)$
- 
- 类似的, 为每个HMM的发射概率  $p(w_i = z | l_i = x)$ , 定义一组CRF发射特征形如  $g_{x,y}(s, i, l_i, l_{i-1})$  if  $w_i = z$  且  $l_i = x$ 。给定每个特征的权重值为  $w_{x,z} = \log p(w_i = z | l_i = x)$ 。

- 对比LSTM:

- LSTM也可以吸纳全局的, 特征提取能力很强
- CRF的标签之间有相互以来, LSTM标签之间没有依赖

- BiLSTM简介: 结合CRF的双向标签约束 + 以及LSTM的强大的特征提取能力
  - 首先采用双向的LSTM, 可以实现对前后全局的依赖
  - 另外学习CRF, 构建标签之间关联性

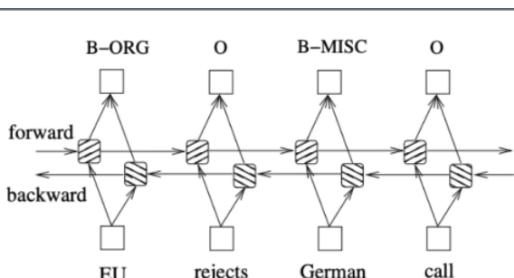


Figure 4: A bidirectional LSTM network.

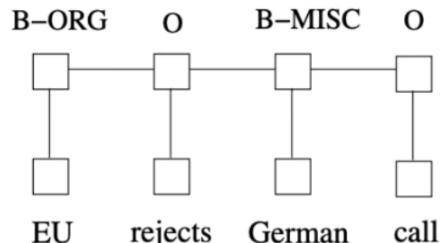


Figure 5: A CRF network.

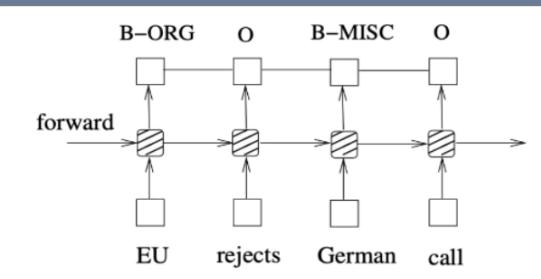


Figure 6: An LSTM-CRF model.

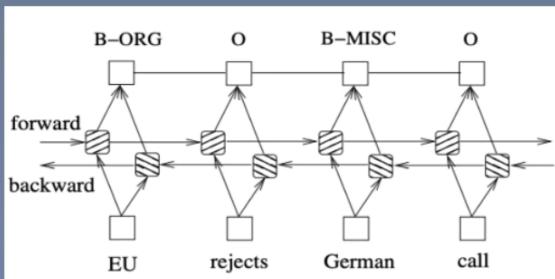
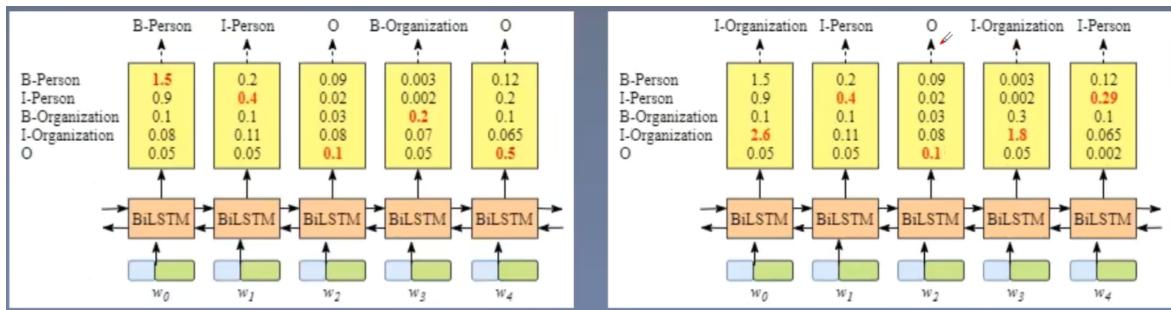


Figure 7: A BI-LSTM-CRF model.

## 2.2. 模型结构:

- 模型构成:
    - 句子中转化为字词向量序列，字词向量可以在事先训练好或随机初始化，在模型训练时可以再训练
    - 经过BiLSTM特征提取，输出的是每个单词对应的预测标签
    - 经过CRF层约束，输出最有标签序列
- 
- |       | B-Person | I-Person | O    | B-Organization | I-Organization |
|-------|----------|----------|------|----------------|----------------|
| $w_0$ | 1.5      | 0.2      | 0.09 | 0.003          | 0.12           |
| $w_1$ | 0.9      | 0.4      | 0.02 | 0.002          | 0.2            |
| $w_2$ | 0.1      | 0.1      | 0.03 | 0.2            | 0.1            |
| $w_3$ | 0.08     | 0.11     | 0.08 | 0.07           | 0.065          |
| $w_4$ | 0.05     | 0.05     | 0.1  | 0.05           | 0.5            |
- Word Embedding:
    - 方法流程:
      - 将一个含有 $n$ 个词的句子记作:  $x = (x_1, x_2, \dots, x_n)$
      - 利用预训练的embedding矩阵将每个字映射为低维稠密的向量
    - 重点理解:
      - 预训练词向量为何可以提高模型泛化能力: 假设同一类型的字词在空间中距离较为接近的空间表示，帮助最终抽取的表达更贴近真实语料
      - 在词向量后加dropout的作用: 训练时相当于只用部分词向量做计算，在推测时全部用上，相当于是一种集成算法
  - BiLSTM提取文本特征:
    - 方法流程:
      - 将一个句子各个字的Embedding序列作为双向LSTM各个时间步的输入
      - 将正反向输出的隐状态进行拼接，得到完整的隐状态序列
    - 重点理解: 详见RNN网络介绍
      - 为什么NER选择的神经网络结构从RNN演变到LSTM,再到BiLSTM
      - 如何从公式层面上理解RNN提取序列化数据特征的弊端
      - 如何从公式层面上理解LSTM的强大之处
      - 为什么LSTM要演变成最终的BiLSTM: 将双向LSTM结果隐藏层输出拼接提取出全文的特征
  - 得到P矩阵:
    - 方法流程:
      - 将完整的隐状态序列接入线性层，从 $n$ 维映射到 $k$ 维，其中 $k$ 是标注集的标签数
      - 从而得到自动提取的句子特征，记作矩阵  $P = (p_1, p_2, \dots, p_n)$ ，注意该矩阵是非归一化矩阵
      - 其中 $p_i$ 表示该单词对应各个类别的分数
  - 为什么要加CRF:
    - 在正常情况下LSTM可以抽出该序列可能的对应标签，但是由于没有标签之间的约束，是的可能会出现非常规的答案，比如第一个词不是B做开头的（例如图二）
    - 引入CRF可以引入标签之间的约束，来修正错误的结果



- CRF层引入：

- 引入原因：NER是一类特殊的任务，因为表征标签的可解释序列“语法”强加了几个硬约束，可能的约束有：
  - 判定“B-label1, I-label2, I-label3...”为错误
  - 判定“O I-label”是错误的
  - 命名实体的开头应该是“B-”而不是“I-”。
- 转移分数：来自CRF层可以学到的转移矩阵（对发射分数进行修正）。转移矩阵是BiLSTM-CRF模型的一个参数。可随机初始化转移矩阵的分数，然后在训练中更新。

	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

- 最终结果的计算：

- CRF考虑前后标记依赖约束，综合使用标记状态转换概率作为评分：式子意为对整个序列x,整个序列标注的打分等于各个位置的打分之和，打分为2部分： $score(x, y) = \sum_{i=1}^n P_{i,y_i} + \sum_{i=1}^{n+1} A_{y_{i-1}, y_i}$

- 前者由LSTM输出的 $p_i$ 决定
- 后者由CRF转移矩阵A决定，其中 $A_{y_{i-1}, y_i}$ 表示从第 $y_{i-1}$ 个标签到第 $y_i$ 个标签的转移得分。

- 路径分数计算：发射分数 + 转移分数

**Emission Score:**

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

- $x_{index,label}$  is the score if the  $index^{th}$  word is labelled by  $label$

- These scores  $x_{1,B-Person} x_{2,I-Person} x_{3,O} x_{4,Organization} x_{5,O}$  are from the previous BiLSTM output.

- As for the  $x_{0,START}$  and  $x_{6,END}$ , we can just set them zeros.

**Transition Score:**

$$TransitionScore =$$

$$t_{START \rightarrow B-Person} + t_{B-Person \rightarrow I-Person} + \\ t_{I-Person \rightarrow O} + t_{0 \rightarrow B-Organization} + t_{B-Organization \rightarrow O} + t_{O \rightarrow END}$$

- $t_{label1 \rightarrow label2}$  is the transition score from  $label1$  to  $label2$
- These scores come from the CRF Layer. In other words, these transition scores are actually the parameters of CRF Layer.

### 2.3. 损失函数和维特比解码：

- 概率计算问题：已知模型参数( $\pi, A, B$ )和观测序列 $O = (o_1, o_2, \dots, o_T)$ ，计算观测序列出现的概率
  - 直接计算法：穷举，计算量过大
  - 前向算法：基于状态序列的路径结构递归计算，局部计算前向概率，利用路径结构将前向概率递推至全局
- 动态规划：
  - 动态规划通过把原问题分解为相对简单的子问题的方式求解复杂问题的方法
  - 动态规划常常适用于有叠子问题和最优子结构性质的问题，动态规划方法所耗时间往往远少于朴素解法
- 作用：
  - 损失函数：快速计算真实路径得分与所有路径得分的比值，CRF Loss
  - 预测或者说是解码：得到最优的标注序列，输出最终结果

- **维特比算法**: 定义: 一种用以选择最优路径的动态规划算法, 从开始状态后每走一步, 记录到达该状态所有路径的最大概率值, 最后以最大值为基准继续向后推进。最后再从结尾回溯最大概率, 也就是最有可能的最优路径
  - 穷举法:  $N^T$
  - 维特比:  $TN^2$
- **CRF损失函数**: 真实路径分数 / 所有路径分数 (真实路径的分数应该是所有路径中分数最高的)

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

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

Now We change the loss function into a log loss function:

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

Because when we are training a model, usually our goal is to **minimize** our loss function, we add a negative sign:

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

- **当前节点得分**: 通过上图的公式推导,  $S_{RealPath}$  得分可以较为容易得出, 这里需要再用一次类似于维特比的算办法, 计算每个节点记录之前所有节点到当前节点的路径总和, 最后一步即可得到所有路径的总和 ( $\log(e^{S_1} + e^{S_2} + \dots + e^{S_N})$ )
  - previous矩阵: 表示第0步的状态矩阵 (1或2两种取值)
  - obs矩阵: 表示从第1步的状态矩阵 (1或2两种取值)
  - 转移矩阵t: 表示从第0步的任意状态转移到第1步任意状态的转移矩阵
  - 得分矩阵score: 得到前后两个步骤转移的总的所有路径的得分, 然后去更新previous矩阵

1) Expand the *previous* to:

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

2) Expand the *obs* to:

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

3) Sum *previous*, *obs* and *transition scores*:

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

Then:

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

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

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

- 预测：采用维特比解码，每个节点记录之前所有节点到当前节点的最优路径，最后一步通过回溯可得一条最优的路径。
  - $previous = [max(scores[00], scores[10]), max(scores[01], scores[11])]$

#### 2.4. 实验设置和结果分析：

- 实验对比：鲁棒性更好（不那么依赖词向量）+ BiLSTM-CRF可以提升模型效果
  - vs Conv-CRF网络 (F1)：再大部分的数据集和任务上BiLSTM-CRF效果更好
  - 鲁棒性 (F1)：和LSTM、CRF的其他组合变种相比，鲁棒性最好
  - vs 现有系统对比 (F1)：使用特备的word embedding和特征提取方式，可以达到考前的准确率
- 论文拓展：以BiLSTM为Baseline
  - OOV问题解决**: Neural Architectures for Named Entity Recognition
    - 增加语料库（无穷无尽）
    - 增加字符级别的词向量，通过网络提取字向量，再拼接起来成为词向量
  - 增加CNN抽取词语特征**: Named Entity Recognition with Bidirectional LSTM-CNNs
  - 通用神经网络 + CRF框架**: NCRF++ (pytorch)

#### 2.5. 代码实现：

- 第三方包安装：
  - `pip install pytorch-crf`
  - `pip install torchtext`
- 详见code