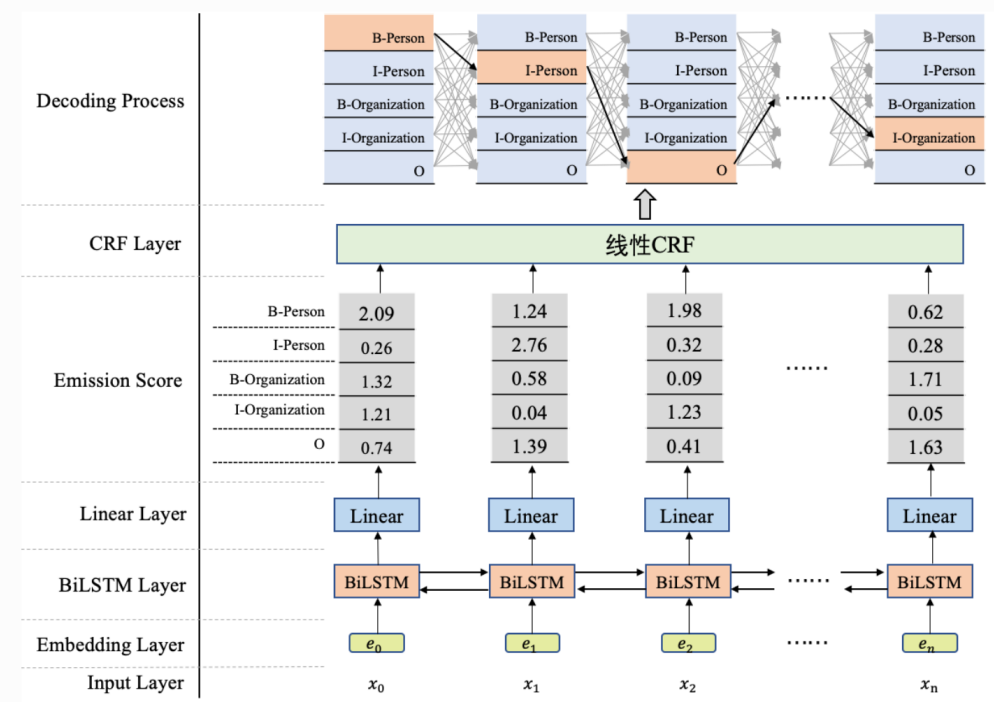
BI-LSTM-CRF

网络结构图



线性CRF

设$X=[x_1,x_2,\ldots,x_n],Y=[y_1,y_2,\ldots,y_n]$均为线性链表示的随机变量序列，若在给定随机变量序列X的条件下，随机变量序列Y的条件概率分布$P(Y|X)$构成条件随机场，即满足马尔可夫性：

- $P(y_i|X,y_1,\ldots,y_{i-1},y_{i+1},\ldots,y_n)=P(y_i|X,y_{i-1},y_{i+1}),i=1,2,\ldots,n$（在$i=1$和$n$的时候只考虑单边）

发射分数和转移分数

令X为输入变量，Y代表相应的标签序列

其中，每个输入x_i均对应着一个标签y_i，这一步对应的为**发射分数**，指示了当前的输入应该对应什么样的标签。在每个标签y_i之间也存在着连线，它表示的是当前位置的标签y_i向下一个位置标签y_{i+1}的一种转移。

发射分数

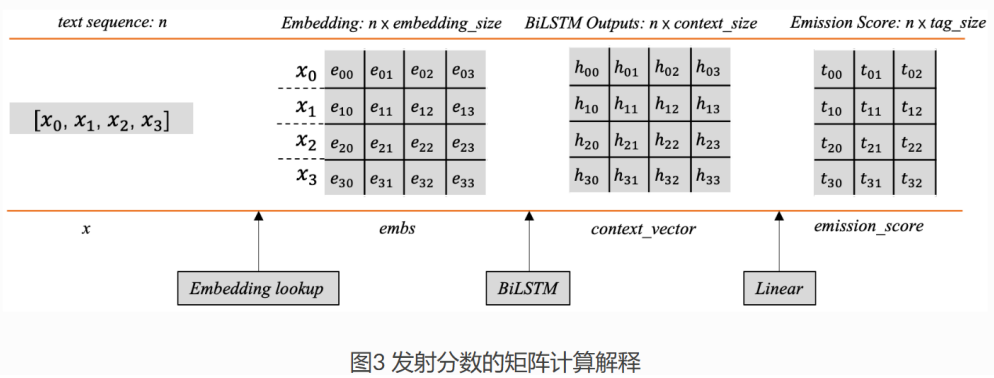


图3 发射分数的矩阵计算解释

给定文本序列$x=[x_1,x_2,x_3,\ldots,x_n]$映射为对应词向量之后，得到一个$(n\times\text{embedding_shape})$的词向量矩阵**embs**

随后将**embs**传入BiLSTM，每个词的位置都会产生一个上下文向量，所有的向量组合后会得到向量矩阵**context_vector**，其中每行代表对应单词经过BiLSTM后的上下文向量。

- **上下文向量**：可以指导当前位置应该输出的标签信息。
  - 问题：一般输出向量的维度并不等于标签的数量，不能直接用来指示输出什么标签。
    - 一般的做法是在后边加一层线性层，将**上下文向量**的维度映射为标签的数量，最终指导标签的输出。
  - $y = XW + b, X \in \mathbb{R}^{n \times \text{context\_size}}, W \in \mathbb{R}^{\text{context\_size} \times \text{tag\_size}}$

转移分数

转移分数表示一个标签向另一个标签转移的分数，分数越高，转移概率就越大。

	B-Person	I-Person	B-Organization	I-Organization	O
B-Person	0.37	0.49	0.35	0.38	0.64
I-Person	0.93	0.71	0.07	0.01	0.11
B-Organization	0.25	0.54	0.16	0.69	0.83
I-Organization	0.02	0.08	0.86	0.95	0.02
O	0.65	0.66	0.78	0.83	0.91

图4 转移分数矩阵图

为CRF中一个可学习的参数矩阵，可以显示的建模标签之间的转移关系。

损失函数

每个x_i都对应了一个发射分数向量，向量的维度即为标签数量。

期待解码为标签序列$y = [y_0, y_1, y_2, \ldots, y_n]$，形式化为对应的条件概率公式如下：

$$(y|x) = P(y_0, y_1, \ldots, y_n | x_0, x_1, \ldots, x_n)$$

假设标签数量为k，序列长度为n，一共有$N = k^n$条路径。

记S_i为第i条路径的分数，则某一特定标签序列出现的概率为

$$P(S_i) = \frac{e^{S_i}}{\sum_j^N e^{S_j}}$$

若有GT的路径，期待CRF解码出这一条序列，则出现的概率记为

$$P(S_{\text{GT}}) = -\frac{e^{S_{\text{GT}}}}{\sum_j^N e^{S_j}}$$

为了方便求解，将损失放入log空间进行求解，因为log函数本身**单调递增**，不影响迭代优化损失函数

$$\begin{aligned}
\text{loss} &= -\log \frac{e^{S_{\text{GT}}}}{\sum_j^N e^{S_j}} \\
&= -\left(\log(e^{S_{\text{GT}}}) - \log(\sum_j^N e^{S_j}) \right) \\
&= \log(\sum_j^N e^{S_j}) - \log(e^{S_{\text{GT}}}) \\
&= \log(e^{S_1} + e^{S_2} + \cdots + e^{S_N}) - S_{\text{GT}}
\end{aligned}$$

损失函数包含两部分，单条真实路径分数S_GT，归一化项，所有路径分数的log_sum_exp

全部路径的分数计算(前向算法)

全部路径的和，需要计算$N = k^n$条路径，指数级别的计算复杂度不可被接受。

动态规划DP：将 所有的路径的和 拆解为 路径上 每个位置的和 进行计算

计算目标：$\log(e^{S_1} + e^{S_2} + \cdots + e^{S_N})$

记变量

- $\text{emmision}_i = [x_{i0}\ x_{i1}\ \ldots\ x_{ik}]$为$i$位置对于$k$个标签的发射分数
- $\text{trans} = \begin{bmatrix} t_{00} & t_{01} & \ldots & t_{0k} \\ t_{10} & t_{11} & \ldots & t_{1k} \\ \vdots & \vdots & \vdots & \vdots \\ t_{k0} & t_{k1} & \ldots & t_{kk} \end{bmatrix}$为转移矩阵，$t_{ij}$代表由Tag j $\rightarrow$ Tag i的分数
- $\text{dp}_i = [s_{i0}\ s_{i1}\ \ldots\ s_{ik}]$，$s_{ij}$代表在第$i$次迭代计算到$i$位置时，以$j$标签结尾的路径分数之和。

迭代过程

1. 截止x_0位置
   - $\text{emmision}_0 = [x_{00}\ x_{01}\ \ldots\ x_{0k}]$
   - $\text{dp}_0 = [x_{00}\ x_{01}\ \ldots\ x_{0k}]$（因为是序列的开始部分）
   - 则截止x_0，将x_0标签的分数累计作为所有路径的分数为：
     - $\text{Total}_0 = \log(e^{x_{00}} + e^{x_{01}} + \cdots + e^{x_{0k}})$
2. 截止x_1位置 **将emission和dp向量expand为矩阵，使用矩阵的方法去计算各个位置的累计值**
   - $\text{emission}_1 = \begin{bmatrix} x_{10} & x_{10} & \ldots & x_{10} \\ x_{11} & x_{11} & \ldots & x_{11} \\ \vdots & \vdots & \vdots & \vdots \\ x_{1k} & x_{1k} & \ldots & x_{1k} \end{bmatrix}_{k\times k}$ **（列扩展）**
   - $\text{dp}_0 = \begin{bmatrix} x_{00} & x_{01} & \ldots & x_{0k} \\ x_{00} & x_{01} & \ldots & x_{0k} \\ \vdots & \vdots & \vdots & \vdots \\ x_{00} & x_{01} & \ldots & x_{0k} \end{bmatrix}_{k\times k}$ **（行扩展）**
   - 计算过程：
   - $$\begin{aligned} \text{scores} &= \text{dp}_0 + \text{trans} + \text{emission}_1 \\ &= \begin{bmatrix} x_{00} & x_{01} & \ldots & x_{0k} \\ x_{00} & x_{01} & \ldots & x_{0k} \\ \vdots & \vdots & \vdots & \vdots \\ x_{00} & x_{01} & \ldots & x_{0k} \end{bmatrix} + \begin{bmatrix} t_{00} & t_{01} & \ldots & t_{0k} \\ t_{10} & t_{11} & \ldots & t_{1k} \\ \vdots & \vdots & \vdots & \vdots \\ t_{k0} & t_{k1} & \ldots & t_{kk} \end{bmatrix} + \begin{bmatrix} x_{10} & x_{10} & \ldots & x_{10} \\ x_{11} & x_{11} & \ldots & x_{11} \\ \vdots & \vdots & \vdots & \vdots \\ x_{1k} & x_{1k} & \ldots & x_{1k} \end{bmatrix} \\ &= \begin{bmatrix} x_{00}+t_{00}+x_{10} & x_{01}+t_{01}+x_{10} & \ldots & x_{0k}+t_{0k}+x_{10} \\ x_{00}+t_{10}+x_{11} & x_{01}+t_{11}+x_{11} & \ldots & x_{0k}+t_{1k}+x_{11} \\ \vdots & \vdots & \vdots & \vdots \\ x_{00}+t_{k0}+x_{1k} & x_{01}+t_{k1}+x_{1k} & \ldots & x_{0k}+t_{kk}+x_{1k} \end{bmatrix} \end{aligned}$$
   - 如$x_{0k} + t_{1k} + x_{11}$就是x_{0k} (0时刻以Tag k结尾)，通过t_{1k} ($k \rightarrow 1$)转移到x_{11} (1时刻以Tag 1结尾)
   - 最后可以得到路径累计分数
   - $$\begin{aligned} \text{dp}_1 &= [\log(e^{\text{score}_{00}} + \cdots + e^{\text{score}_{0k}}) \quad \cdots \quad \log(e^{\text{score}_{k0}} + \cdots + e^{\text{score}_{kk}})] \\ &= [\log(e^{x_{00}+t_{00}+x_{10}} + \cdots + e^{x_{0k}+t_{0k}+x_{10}}) \quad \cdots \quad \log(e^{x_{00}+t_{k0}+x_{1k}} + \cdots + e^{x_{0k}+t_{kk}+x_{1k}})] \end{aligned}$$
   - 则截止x_1所有的路径和：

$$
\begin{aligned}
\text{total}_1 &= \log(e^{\text{dp}_1[0]} + \cdots + e^{\text{dp}_1[k]}) \\
&= \log(e^{\log(e^{x_{00}+t_{00}+x_{10}}+\cdots+e^{x_{0k}+t_{0k}+x_{10}})} + \cdots + e^{\log(e^{x_{00}+t_{k0}+x_{1k}}+\cdots+e^{x_{0k}+t_{kk}+x_{1k}})}) \\
&= \log((e^{x_{00}+t_{00}+x_{10}} + \cdots + e^{x_{0k}+t_{0k}+x_{10}}) + \cdots + (e^{x_{00}+t_{k0}+x_{1k}} + \cdots + e^{x_{0k}+t_{kk}+x_{1k}}))
\end{aligned}
$$

3. 以此递推即可，时间复杂度$O(nk^2)$（转移k^2，链长为n）

维特比解码

训练的要素：CRF损失函数，单条路径分数的计算，全部路径分数（归一化因子）已经齐全了，可以进行BiLSTM+CRF的训练。

最终Inference的时候，需要有从全部路径中查询最优路径的过程，使用Viterbi算法。

记变量：

- $\text{emmision}_i = [x_{i0}\ x_{i1}\ \dots\ x_{ik}]$为$i$位置对于$k$个标签的发射分数
- $\text{trans} = \begin{bmatrix} t_{00} & t_{01} & \dots & t_{0k} \\ t_{10} & t_{11} & \dots & t_{1k} \\ \vdots & \vdots & \vdots & \vdots \\ t_{k0} & t_{k1} & \dots & t_{kk} \end{bmatrix}$为转移矩阵，$t_{ij}$代表由$\text{Tag j} \to \text{Tag i}$的分数
- $\text{dp}_i = [s_{i0}\ s_{i1}\ \dots\ s_{ik}]$，$s_{ij}$代表在第$i$次迭代计算到$i$位置时，以$j$标签结尾的路径中，得分最高的路径的具体得分。
- $\text{beta}_i = [p_{i0}\ p_{i1}\ \dots\ p_{ik}]$，$p_{ij}$代表在第$i$次迭代计算到$i$位置时，以$j$标签结尾的路径中，得分最高的路径在前一个位置$i-1$处的标签索引。

迭代过程

1. 截止x_0位置
   - $\text{emmision}_0 = [x_{00}\ x_{01}\ \dots\ x_{0k}]$
   - $\text{dp}_0 = [x_{00}\ x_{01}\ \dots\ x_{0k}]$（因为是序列的开始部分）
   - $\text{beta}_0 = [-1, -1, \dots, -1]$(因为起始边前面没有任何路径，用$-1$来初始化)
2. 截止x_1位置 **矩阵扩展的位置很类似于前向算法，不再写一遍了**
   - 计算过程：
   - $$
     \begin{aligned}
     \text{scores} &= \text{dp}_0 + \text{trans} + \text{emission}_1 \\
     &= \begin{bmatrix} x_{00} & x_{01} & \dots & x_{0k} \\ x_{00} & x_{01} & \dots & x_{0k} \\ \vdots & \vdots & \vdots & \vdots \\ x_{00} & x_{01} & \dots & x_{0k} \end{bmatrix} + \begin{bmatrix} t_{00} & t_{01} & \dots & t_{0k} \\ t_{10} & t_{11} & \dots & t_{1k} \\ \vdots & \vdots & \vdots & \vdots \\ t_{k0} & t_{k1} & \dots & t_{kk} \end{bmatrix} + \begin{bmatrix} x_{10} & x_{10} & \dots & x_{10} \\ x_{11} & x_{11} & \dots & x_{11} \\ \vdots & \vdots & \vdots & \vdots \\ x_{1k} & x_{1k} & \dots & x_{1k} \end{bmatrix} \\
     &= \begin{bmatrix} x_{00}+t_{00}+x_{10} & x_{01}+t_{01}+x_{10} & \dots & x_{0k}+t_{0k}+x_{10} \\ x_{00}+t_{10}+x_{11} & x_{01}+t_{11}+x_{11} & \dots & x_{0k}+t_{1k}+x_{11} \\ \vdots & \vdots & \vdots & \vdots \\ x_{00}+t_{k0}+x_{1k} & x_{01}+t_{k1}+x_{1k} & \dots & x_{0k}+t_{kk}+x_{1k} \end{bmatrix}
     \end{aligned}
     $$
   - 得到路径最大分数：
   - $$
     \begin{aligned}
     \text{dp}_1 &= [\max(\text{score}_{00}, \dots, \text{score}_{0k}) \quad \cdots \quad \max(\text{score}_{k0}, \dots, \text{score}_{kk})] \\
     &= [\max(x_{00}+t_{00}+x_{10}, \dots, x_{0k}+t_{0k}+x_{10}) \quad \cdots \quad \max(x_{00}+t_{k0}+x_{1k}, \dots, x_{0k}+t_{kk}+x_{1k})]
     \end{aligned}
     $$
   - 根据max中的计算结果，可以填写记录beta_1的值
3. 以此递推即可，时间复杂度$O(nk^2)$（转移k^2，链长为n），最终结果应该可以从beta矩阵中回溯。