深度学习下的序列标注

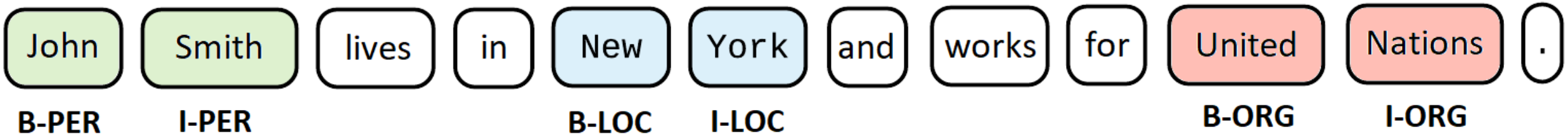
王瑞

上海交大计算机系

wangrui12@sjtu.edu.cn

序列标注

- **序列标注（Sequence Tagging/Labeling）**是NLP中最基础的任务，应用十分广泛，如分词、词性标注（POS tagging）、命名实体识别（Named Entity Recognition，NER）、关键词抽取、语义角色标注（Semantic Role Labeling）、槽位抽取（Slot Filling）等实质上都属于序列标注的范畴

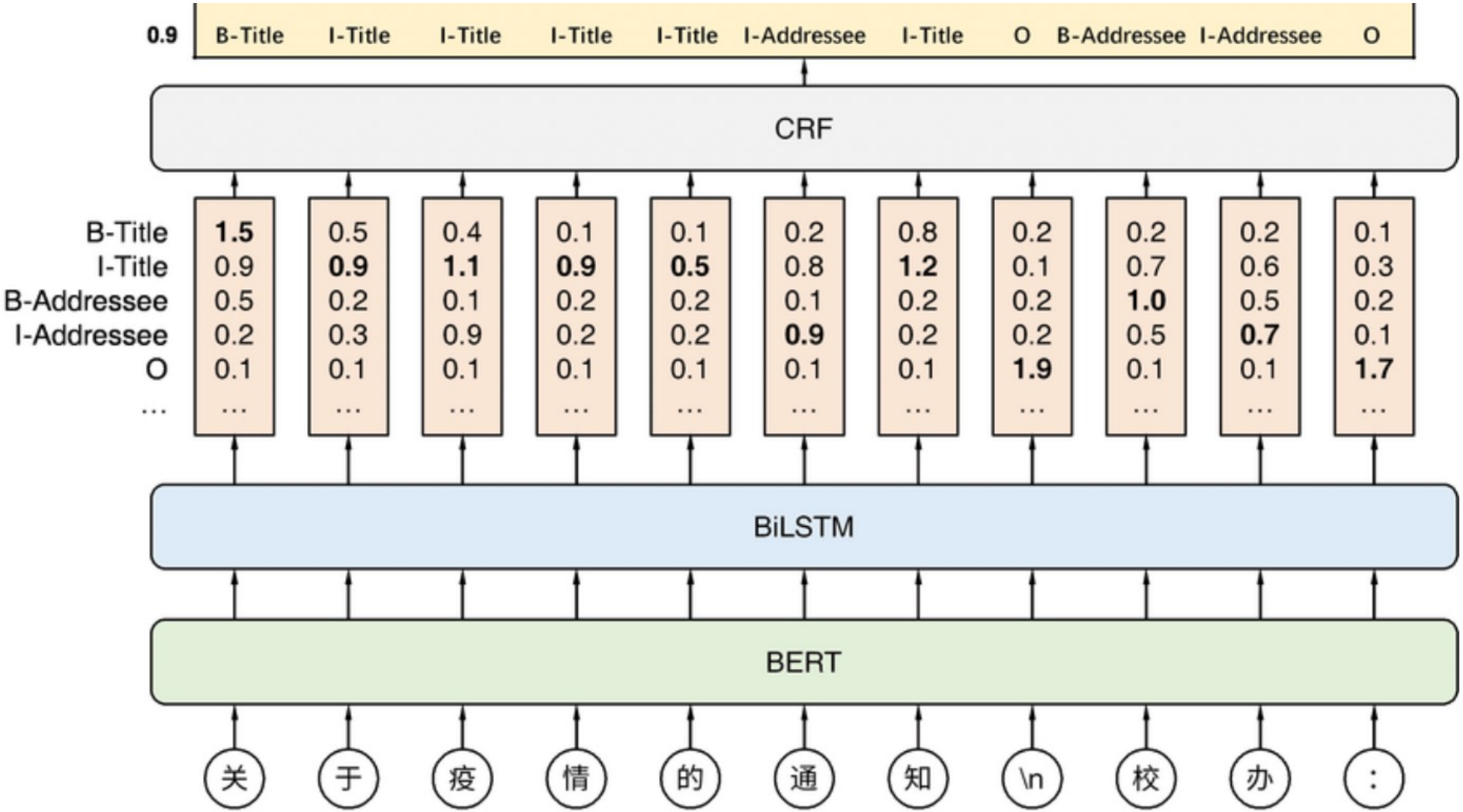


序列标注

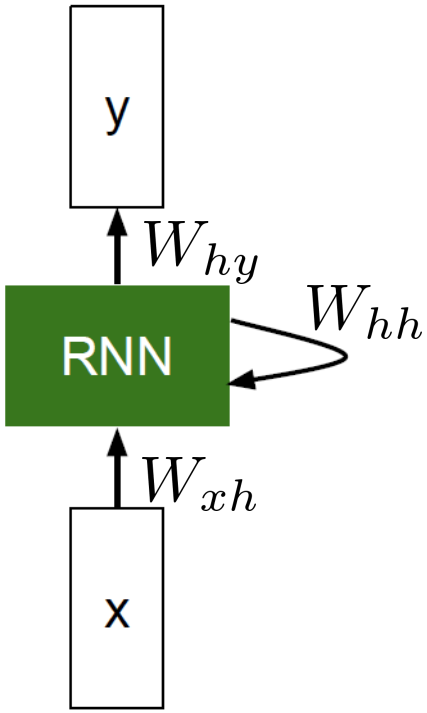
- 输入序列$\mathbf{X}=\{x_1, x_2, x_3, \dots, x_n\}$，输出标注序列$\mathbf{Y}=\{y_1, y_2, y_3, \dots, y_n\}$，其中$|y|=m$（y的种类数一共有m类）

- Y一般标注法(以NER为例)为:
  - B (begin): signifies beginning of an NE
  - I (inside)： signifies that the word is inside an NE
  - E (end)： signifies that the word is the end of an NE
  - S (singleton): signifies that the single word is an NE
  - O (outside): signifies that the word is just a regular word outside of an NE

序列标注：深度学习框架

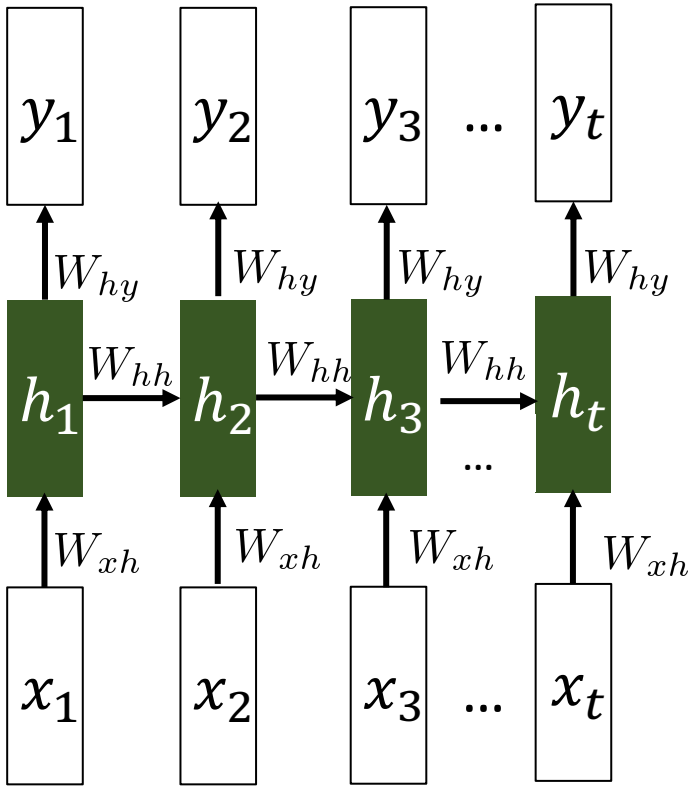

0.9
B-Title
I-Title
I-Title
I-Title
I-Title
I-Addressee
I-Title
O
B-Addressee
I-Addressee
O
CRF
B-Title
I-Title
B-Addressee
I-Addressee
O
...
1.5 0.9 0.5 0.2 0.1 ...
0.5 0.9 0.2 0.3 0.1 ...
0.4 1.1 0.1 0.9 0.1 ...
0.1 0.9 0.2 0.2 0.1 ...
0.1 0.5 0.2 0.2 0.1 ...
0.2 0.8 0.1 0.9 0.1 ...
0.8 1.2 0.2 0.2 0.1 ...
0.2 0.1 0.2 0.2 1.9 ...
0.2 0.7 1.0 0.5 0.1 ...
0.2 0.6 0.5 0.7 0.1 ...
0.1 0.3 0.2 0.1 1.7 ...
BiLSTM
BERT
关
于
疫
情
的
通
知
\n
校
办
:

RNN回顾



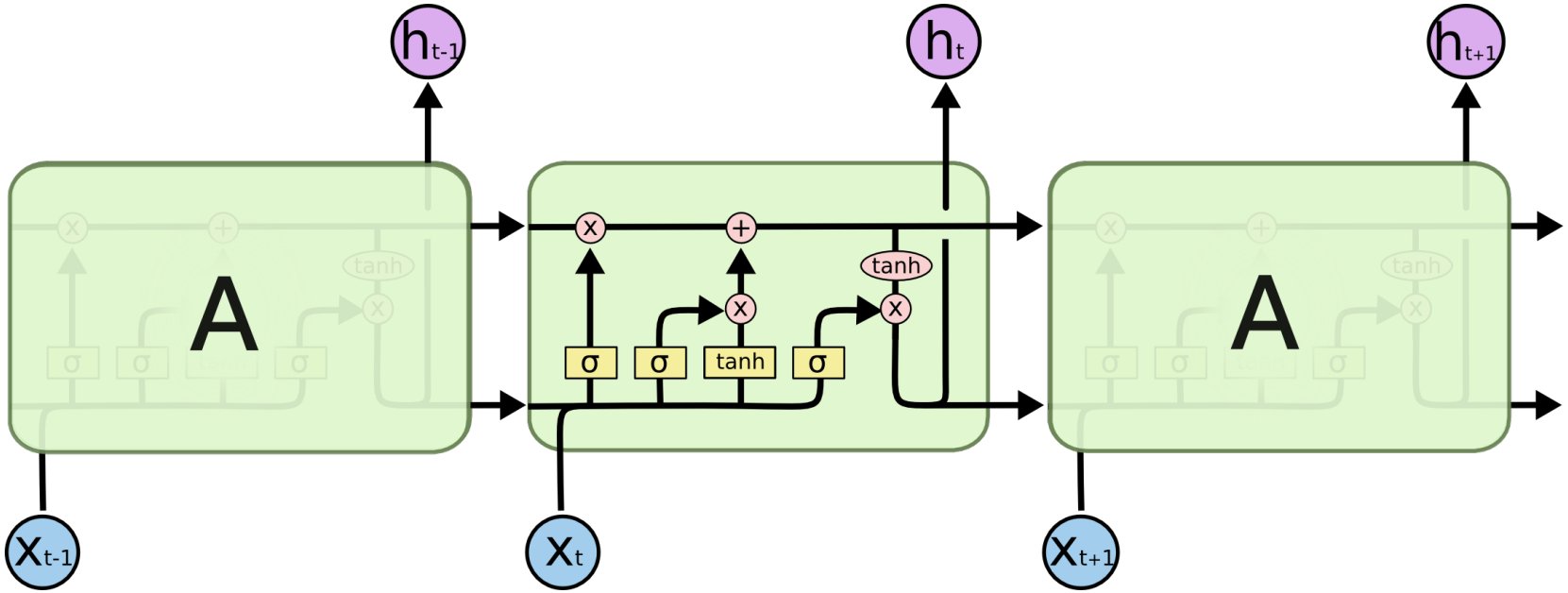
$$y_t = softmax(W_{hy}h_t)$$

Unfolding



LSTM回顾

- 存储细胞的内容可以在很长时间内保持不变（在输入门和遗忘门都关闭时）
- 允许信息跨多个时间步保留
- 允许梯度跨多个时间步流动
- 克服了梯度消失问题



已有方法表现如何

Table 2: Comparison of tagging performance on POS, chunking and NER tasks for various models.

		POS	CoNLL2000	CoNLL2003
Random	Conv-CRF (Collobert et al., 2011)	96.37	90.33	81.47
	LSTM	97.10	92.88	79.82
	BI-LSTM	97.30	93.64	81.11
	CRF	97.30	93.69	83.02
	LSTM-CRF	**97.45**	93.80	84.10
	BI-LSTM-CRF	97.43	**94.13**	**84.26**
Senna	Conv-CRF (Collobert et al., 2011)	97.29	94.32	88.67 (89.59)
	LSTM	97.29	92.99	83.74
	BI-LSTM	97.40	93.92	85.17
	CRF	97.45	93.83	86.13
	LSTM-CRF	97.54	94.27	88.36
	BI-LSTM-CRF	**97.55**	**94.46**	**88.83 (90.10)**

[Huang et al., 2015]

CRF回顾

- 对标签之间的转移(transition)概率进行建模
- 特征函数：
  - 输入序列，$\boldsymbol{X}$;
  - 当前输入的位置信息i;
  - 前一个输入的标签l_{i-1};
  - 当前输入的标签l_i;

$$f(X, i, l_{i-1}, l_i)$$

CRF回顾

- 从特征函数到概率

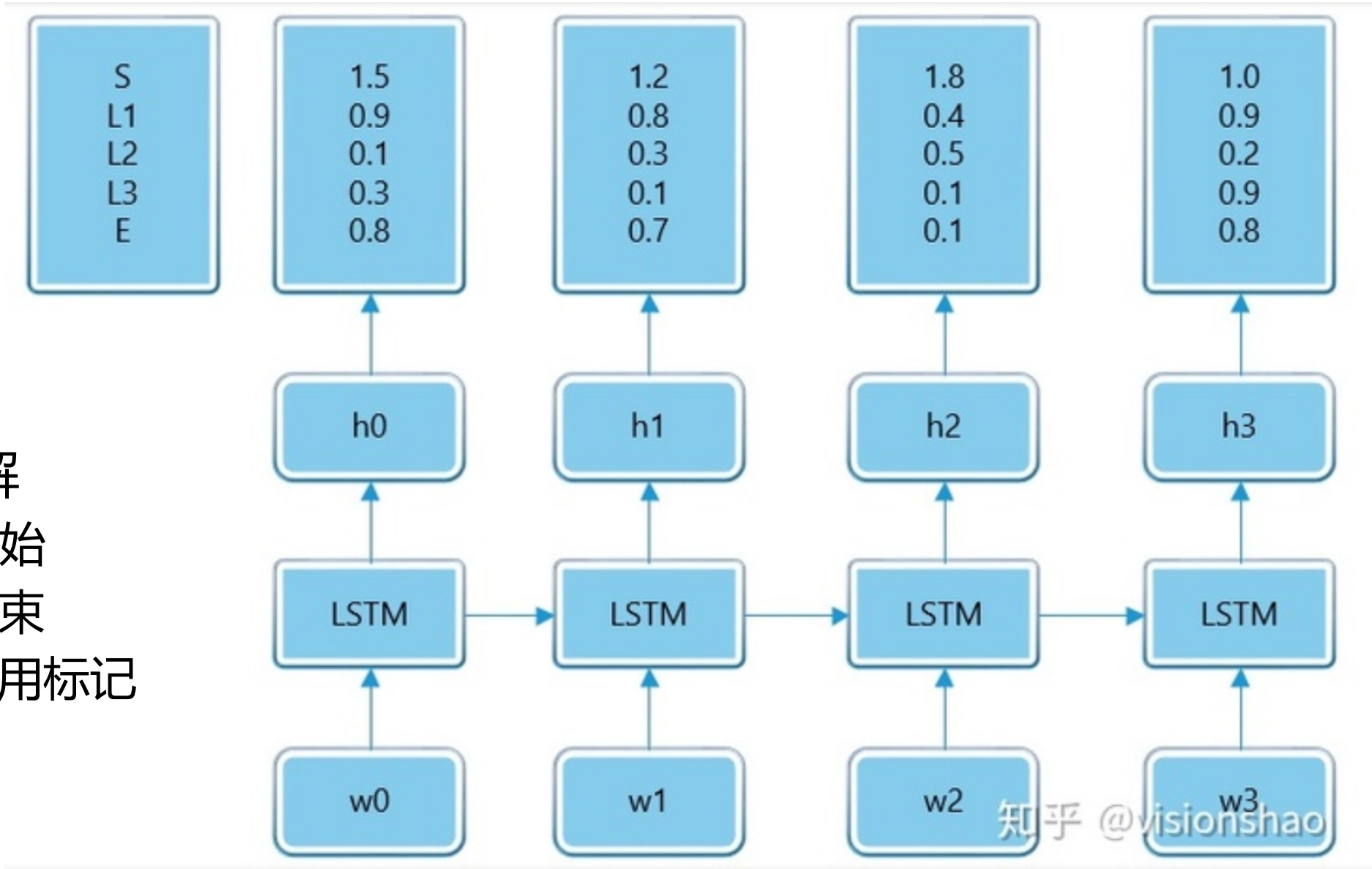
$$score(l|s) = \sum_{j=1}^{m} \sum_{i=1}^{n} \lambda_j f_j(s, i, l_i, l_{i-1})$$

- 归一化

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

序列标注：深度学习框架 （LSTM+CRF）

- 便于理解
  - S：起始
  - E：结束
  - L：通用标记

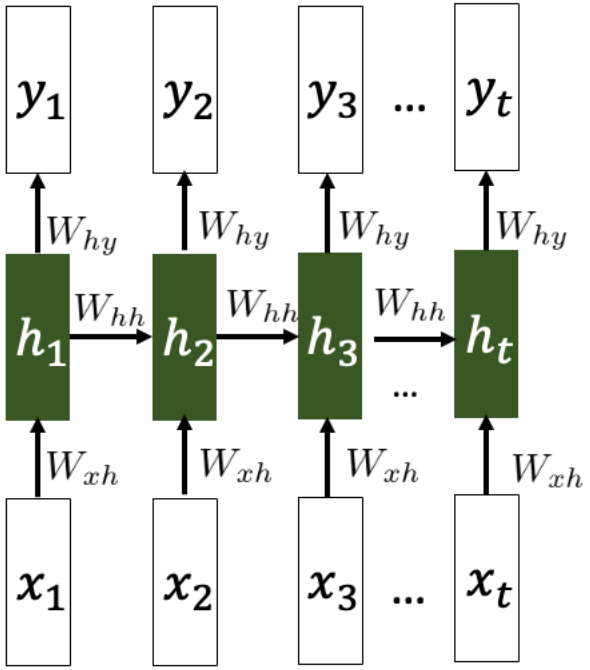


主要问题

- 训练：给定训练数据($\boldsymbol{X}, \boldsymbol{Y}$)，如何对LSTM+CRF模型进行训练？
- 推理：对于训练好的LSTM+CRF模型，给定输入$\boldsymbol{X}$，如何求得最可能的结果 $\hat{Y}$ (\hat 表示预测值)。

发射概率

- 发射(*emission*)概率e：时刻i处于状态h_i的条件下生成观测y_i的概率，在LSTM-CRF中，h_i是x_i一一对应的隐层状态，所以这个发射概率也可以理解成从序列x_i到标签y_i的概率。



$$e_i = p(y_i|x_i) = \frac{e^{score(y_i)}}{\sum_y e^{score(y)}}$$

概率取对数

- **为什么要取对数?**
  - 幂运算可能会导致数值很大，浮点数溢出
  - 相对于乘法，减法看起来简单一些

$$logp(y|x) = score(y) - log(\sum_y e^{score(y)})$$

词的标签分布概率

- 每个词x_i输入LSTM后，得到对应的发射概率e_i
- 整个序列$\boldsymbol{X}=\{x_1, x_2, x_3, \ldots, x_n\}$输入，得到对应的发射矩阵$E$，其中

$$e_i = E[i] \in R^m$$

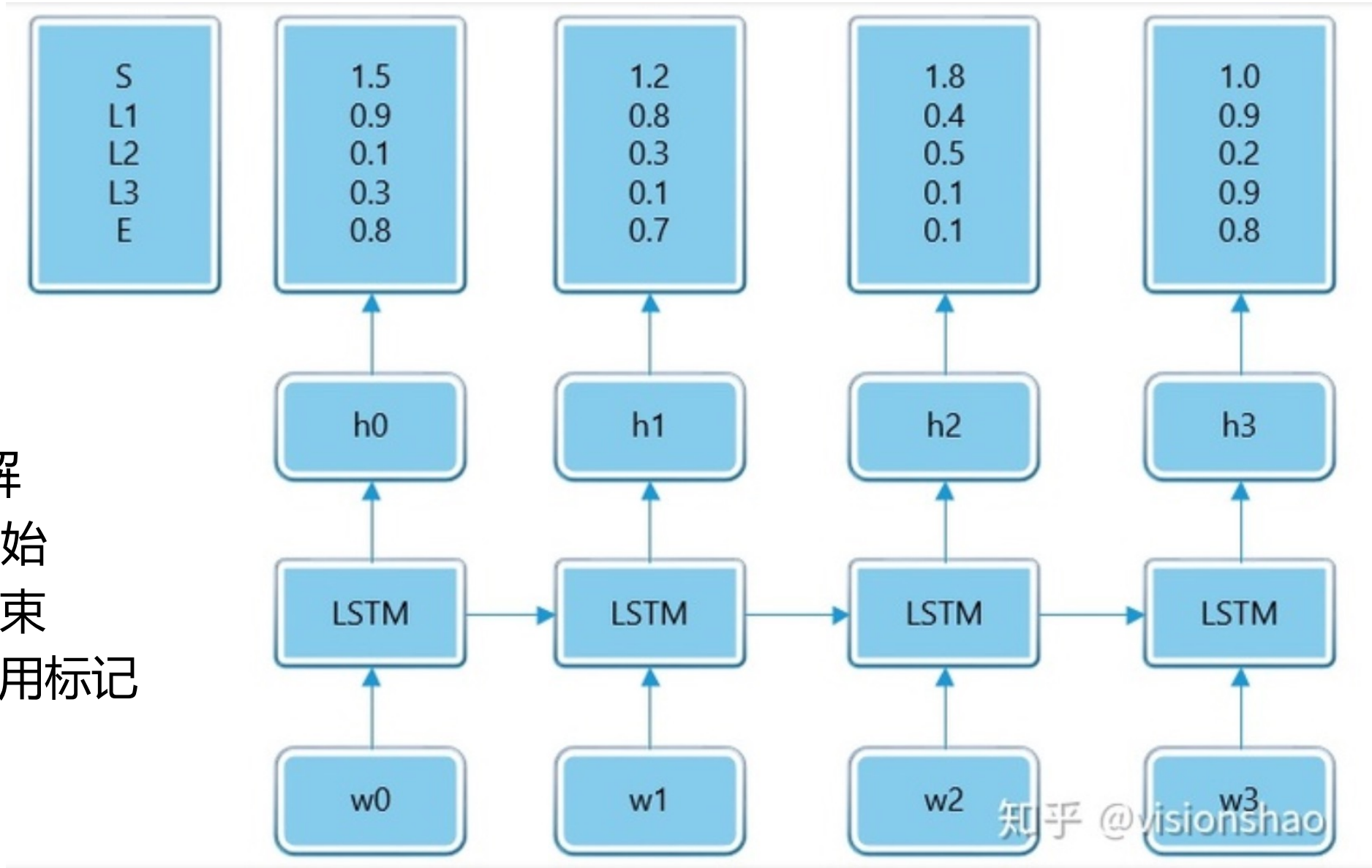
是一个m维的向量（y的种类数一共有m个）

- 其中第k类标签y_k的标签为：

$$e_i(y_k)$$

词的标签分布概率

- 便于理解
  - S：起始
  - E：结束
  - L：通用标记

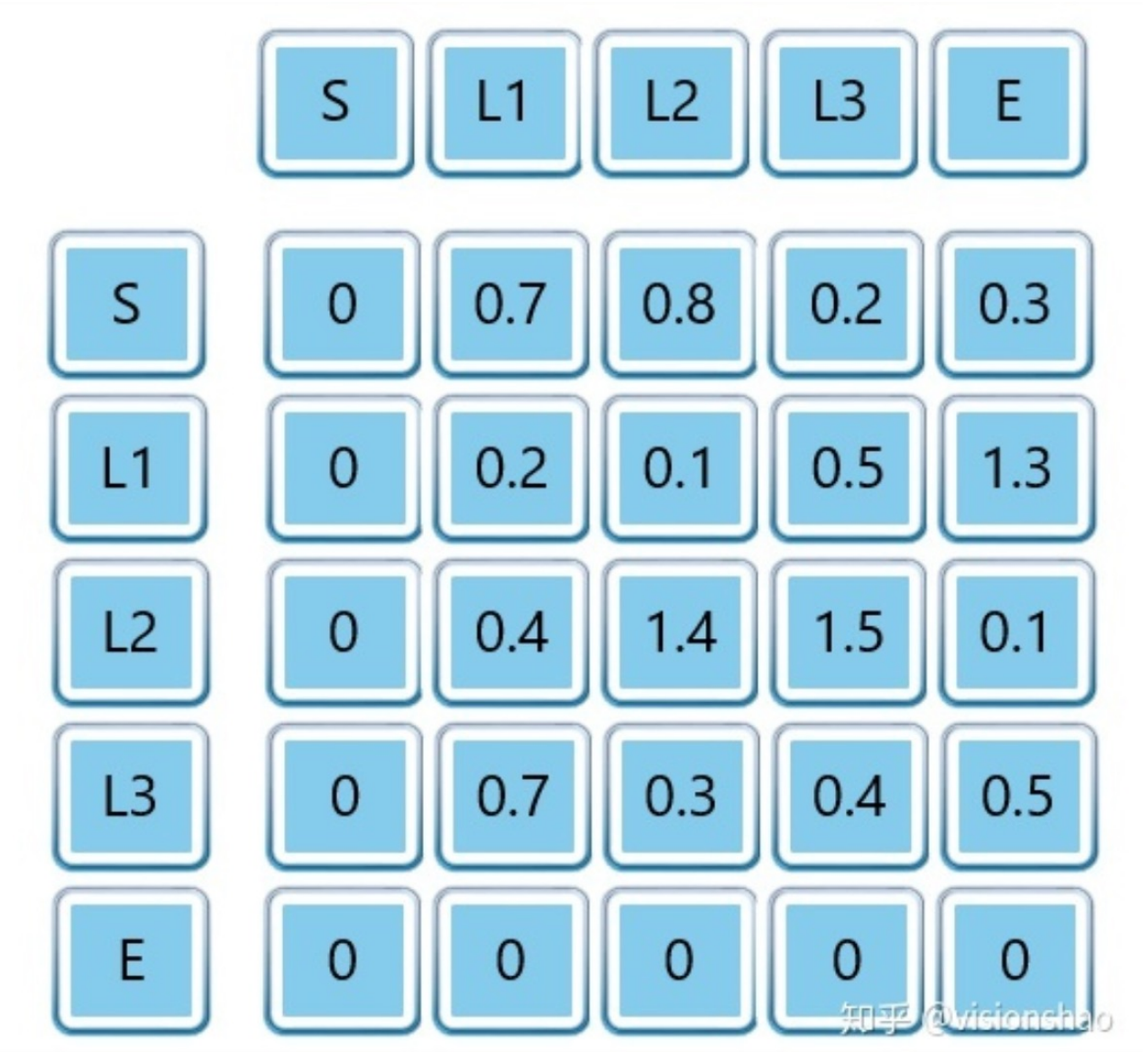


标签转移(transition)概率

- $\boldsymbol{Y}=\{y_1, y_2, y_3, \ldots, y_n\}$是一个序列（sequence），彼此间有依赖关系
  - 动词后面一般不再出现动词，代词后面动词概率大
  - 两个命名实体一般不连续出现
  - 中文中时间一般出现在地点前面
  - 中文人名一般不超过4个字
  - 。。。
- **如何能构建这个转移概率?**

标签转移矩阵

- 我们将标签y_{i-1}转移到y_i的概率定义为标签转移概率$T[i-1, i]$
- 这样就构建了一个标签转移矩阵T。



综合打分

- 对于一个标签序列Y综合打分为：发射概率+转移概率（log前提下）

$$score(Y) = \sum_{i=1}^{n} e_i + \sum_{i=2}^{n} (T[i-1, i])$$

全部可能

- $\boldsymbol{logp(y|x) = score(y) - log(\sum_y e^{score(y)})}$
  - 我们已经知道了$\boldsymbol{score(Y)}$
  - 还需要知道全部可能概率$\boldsymbol{log(\sum_y e^{score(y)})}$
- 但是，这种情况下共有 m^n个(m是标签种类，n是序列长度)可能的标记结果，如果列举所有的路径再求总分，那么计算量就很大。

- **动态规划**
  - 大致上，若要解一个给定问题，我们需要解其不同部分（即子问题），再根据子问题的解以得出原问题的解。
  - 通常许多子问题非常相似，为此动态规划法试图仅仅解决每个子问题一次，从而减少计算量：一旦某个给定子问题的解已经算出，则将其记忆化存储，以便下次需要同一个子问题解之时直接查表。
  - 这种做法在重复子问题的数目关于输入的规模呈指数增长时特别有用。

子问题

- 我们知道对于t时刻的输出y_t来说，其有m种可能，我们记录下从开始时刻到t时刻时，y_t取某一个值i时经过的所有路径的分数的指数和的对数:

$$logZ(y_t = i) = log \sum_{y_t} e^{score(start \rightarrow y_t)}$$

下一个子问题

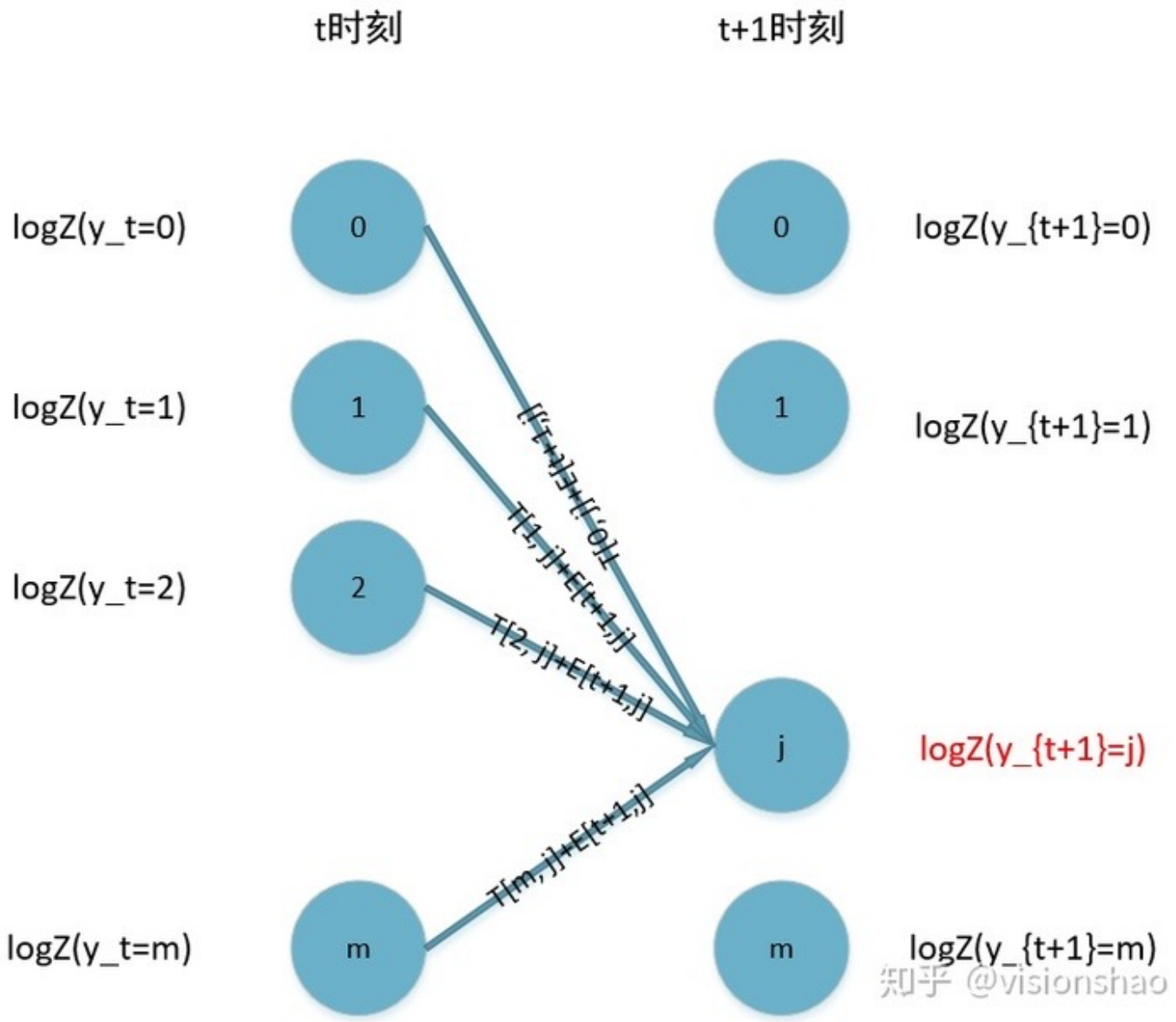
- 我们就可以利用此信息获得t+1时刻，$y_{t+1}=j$的所有路径的得分的指数和的对数.
- 我们知道对于$y_t=i$到$y_{t+1}=j$，路径得分变化是在原来的基础上加上

$$T[i,j] + E[t+1,j]$$

- 所以:

$$logZ(y_{t+1}=j) = log(\sum_{y_t=0}^{m-1} e^{score(start->y_t)}) + (T[i,j] + E[t+1,j])$$

计算图解


t时刻
t+1时刻
logZ(y_t=0)
logZ(y_t=1)
logZ(y_t=2)
logZ(y_t=m)
0
1
2
m
0
1
j
m
T[1, j]+E[t+1,j]
T[2, j]+E[t+1,j]
T[m, j]+E[t+1,j]
logZ(y_{t+1}=0)
logZ(y_{t+1}=1)
logZ(y_{t+1}=j)
logZ(y_{t+1}=m)
知乎 @visionshao

维护向量

- 这样就需要我们维护一个m维向量α_t，$\alpha_t[i]$表示时刻t，$y_t=i$时所有路径指数和的对数$\log Z(y_t=i)$，即：

$$\alpha_t = [logZ(y_t = 0), logZ(y_t = 1), \ldots, logZ(y_t = m)]$$

维护向量

- 如果我们想根据α_t得到α_{t+1}，则按照前面的公式来说，

$$\alpha_{t+1} = [\sum(\alpha_t + T[:,0] + E[t+1,1]), \sum(\alpha_t + T[:,0] + E[t+1,1]), \ldots, \sum(\alpha_t + T[:,m] + E[t+1,m])]$$

最终

当得到最后一个时刻 n 的 α 时，我们只需要用 $logsumexp(\alpha_n)$ 就可以得到最终所有路径的分数的指数和的对数，即 $logZ = log(\sum_y e^{score(y)})$ 。

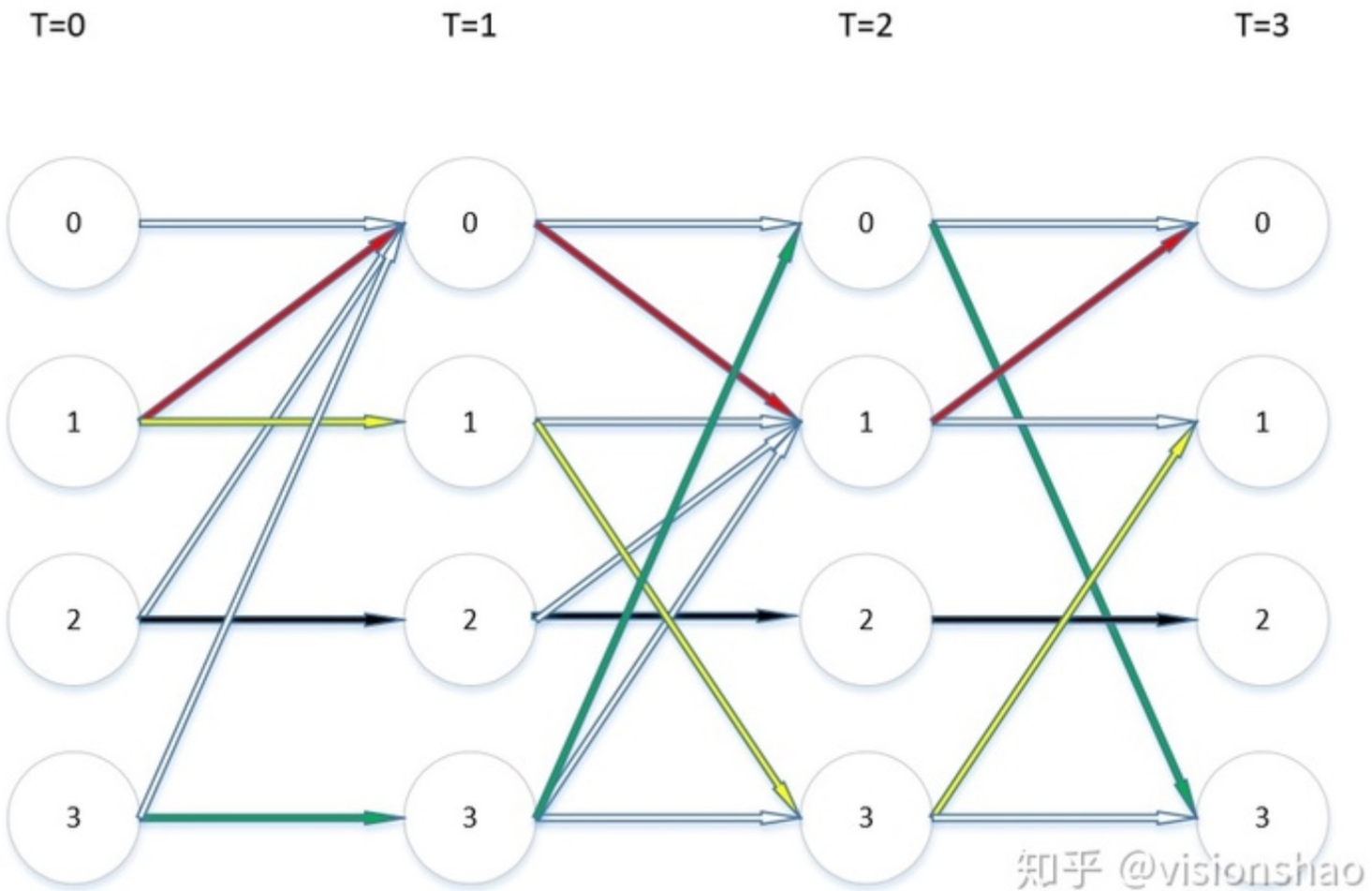
这样，我们就能最终得到 $logp(y|x) = score(y) - log(\sum_y e^{score(y)})$ 。

主要问题

- 训练：给定训练数据$(\boldsymbol{X}, \boldsymbol{Y})$，如何对LSTM+CRF模型进行训练？
- 推理：对于训练好的LSTM+CRF模型，给定输入$\boldsymbol{X}$，如何求得最可能的结果 $\hat{\boldsymbol{Y}}$ (\hat 表示预测值)。

得分最高的路径

带有颜色的箭头表示的路径是指向该节点的得分最高的路径

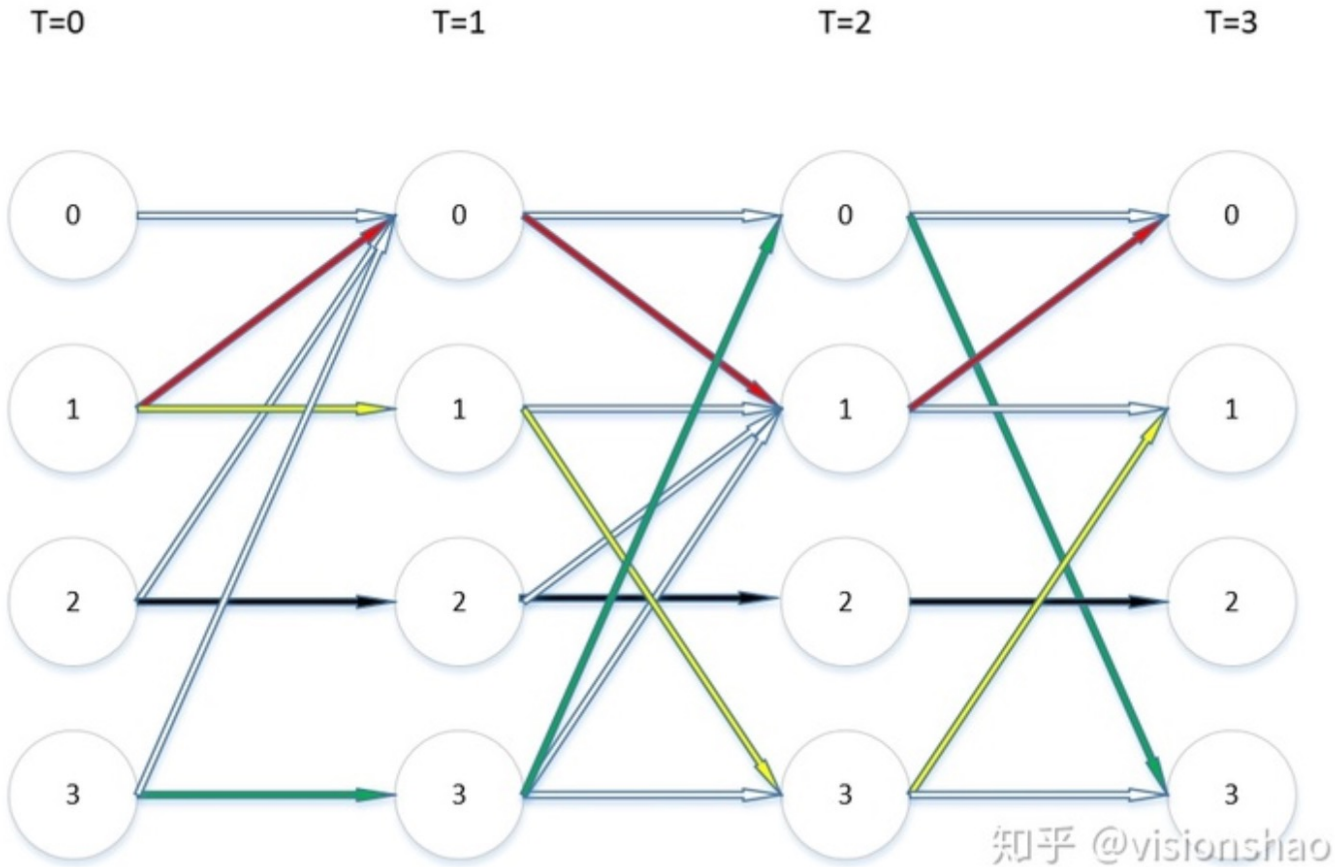


得分最高的路径

我们只需要找出得分最高的路径（某种颜色表示的路径）指向的节点，然后不断回溯即可找到整体的最优路径。

*n*个词

m类标签



得分最高的路径

- 要想实现这种方法，我们需要保存每个节点对应的得分最高的路径及其分数。
- 经过LSTM后得到的发射矩阵为$E \in R^{(n,m)}$
- 标签转移矩阵为T $\in R^{(m,m)}$
- 其中R表示矩阵的维度

t_0时刻

- 首先，t=0时刻，我们获取各个节点对应的最大路径的得分，

$$\beta_0 \in R^m$$

- 即节点的发射得分

$$E[0, :]$$

t_1时刻

- t_0时刻节点i的最大得分路径到节点j的路径得分：

$$\mathbf{M_1[i,j] = \beta \circ T}$$

- $\circ$表示element-wise乘积（如果取log就是相加）

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \circ \begin{bmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \\ b_{31} & b_{32} & b_{33} \end{bmatrix} = \begin{bmatrix} a_{11}\,b_{11} & a_{12}\,b_{12} & a_{13}\,b_{13} \\ a_{21}\,b_{21} & a_{22}\,b_{22} & a_{23}\,b_{23} \\ a_{31}\,b_{31} & a_{32}\,b_{32} & a_{33}\,b_{33} \end{bmatrix}.$$

t_1时刻

- 如果我们想求得t=1时刻到节点j的最大得分路径，只需要对 $\mathrm{M_1}[:,\mathbf{j}]$求最大值。
- 同理，t=1时刻，我们获取各个节点对应的最大路径的得分，

$$\beta_1 \in R^m$$

- 也得到了t=1时刻各个节点对应的最大路径在t=0时刻对应的节点向量

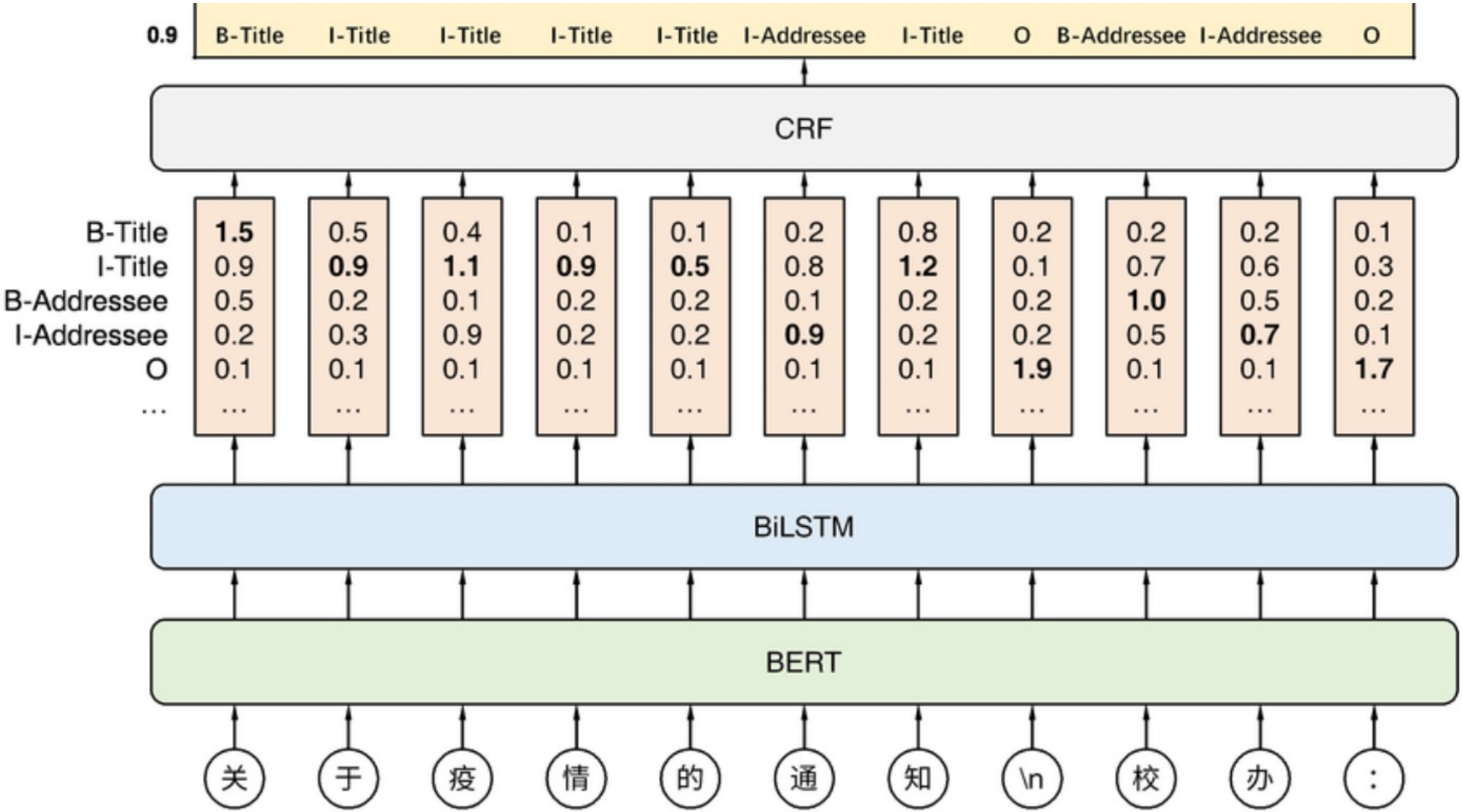
$$P_1 \in R^m$$

- 其中/表示index

$$I_0 = \mathrm{P}_1[I_1]$$

- 不断重复上面的步骤，直到得到$t = n$ -1时刻的β_{n-1}和 P_{n-1} 。这时β_{n-1}对应的是最后时刻的各个节点对应的最大路径得分。我们求得β_{n-1}中最大值对应的索引 I_{n-1}
- 这个索引表示最优路径的最后一个节点。然后我们将 I_{n-1}带入P_{n-1} ，得到 $I_{n-2} = P_{n-1}[I_{n-1}]$ ，表示最优路径倒数第二个节点，然后依次类推，知道得到 $I_0 = P_1[I_1]$
- 这时，最优路径为I=[I_0,I_1,...,I_{n-1}]

序列标注：深度学习框架


0.9
B-Title
I-Title
I-Title
I-Title
I-Title
I-Addressee
I-Title
O
B-Addressee
I-Addressee
O
CRF
B-Title
I-Title
B-Addressee
I-Addressee
O
…
1.5 0.9 0.5 0.2 0.1 …
0.5 0.9 0.2 0.3 0.1 …
0.4 1.1 0.1 0.9 0.1 …
0.1 0.9 0.2 0.2 0.1 …
0.1 0.5 0.2 0.2 0.1 …
0.2 0.8 0.1 0.9 0.1 …
0.8 1.2 0.2 0.2 0.1 …
0.2 0.1 0.2 0.2 1.9 …
0.2 0.7 1.0 0.5 0.1 …
0.2 0.6 0.5 0.7 0.1 …
0.1 0.3 0.2 0.1 1.7 …
BiLSTM
BERT
关
于
疫
情
的
通
知
\n
校
办
：

已有方法表现如何

Table 2: Comparison of tagging performance on POS, chunking and NER tasks for various models.

		POS	CoNLL2000	CoNLL2003
Random	Conv-CRF (Collobert et al., 2011)	96.37	90.33	81.47
	LSTM	97.10	92.88	79.82
	BI-LSTM	97.30	93.64	81.11
	CRF	97.30	93.69	83.02
	LSTM-CRF	**97.45**	93.80	84.10
	BI-LSTM-CRF	97.43	**94.13**	**84.26**
Senna	Conv-CRF (Collobert et al., 2011)	97.29	94.32	88.67 (89.59)
	LSTM	97.29	92.99	83.74
	BI-LSTM	97.40	93.92	85.17
	CRF	97.45	93.83	86.13
	LSTM-CRF	97.54	94.27	88.36
	BI-LSTM-CRF	**97.55**	**94.46**	**88.83 (90.10)**