NER 的含义

命名实体识别（NER）是指识别文本中具有特定意义的实体，主要包括人名、地名、机构名、专有名词等。命名实体识别是信息提取、问答系统、句法分析、机器翻译等应用领域的重要基础工具，作为结构化信息提取的重要步骤。

方法识别

先把解决问题的逻辑说一下，然后解释主要的代码。

代码是在 Tensorflow 下建立只有一个隐藏层的 DNN 来处理 NER 问题。

1.问题识别：

NER 是个分类问题。

给一个单词，我们需要根据上下文判断，它属于下面四类的哪一个，如果都不属于，则类别为 0，即不是实体，所以这是一个需要分成 5 类的问题：

- Person (PER)
- Organization (ORG)
- Location (LOC)
- Miscellaneous (MISC)

我们的训练数据有两列，第一列是单词，第二列是标签。

EU ORG

rejects O

German MISC

Peter PER

BRUSSELS LOC

2.模型：

接下来我们用深度神经网络对其进行训练。

模型如下：

输入层的 x^(t) 为以 x_t 为中心的窗口大小为 3 的上下文语境，x_t 是 one-hot 向量，x_t 与 L 作用后就是相应的词向量，词向量的长度为 d = 50 ：

$$\boldsymbol{x}^{(t)} = [\boldsymbol{x}_{t-1}\boldsymbol{L},\ \boldsymbol{x}_t\boldsymbol{L},\ \boldsymbol{x}_{t+1}\boldsymbol{L}] \in \mathbb{R}^{3d}$$

我们建立一个只有一个隐藏层的神经网络，隐藏层维度是 100，y^ 就是得到的预测值，维度是 5：

$$\boldsymbol{h} = \tanh(\boldsymbol{x}^{(t)}\boldsymbol{W} + \boldsymbol{b}_1)$$
$$\hat{\boldsymbol{y}} = \mathrm{softmax}(\boldsymbol{h}\boldsymbol{U} + \boldsymbol{b}_2)$$

用交叉熵来计算误差：

$$J(\theta) = \mathrm{CE}(\boldsymbol{y}, \hat{\boldsymbol{y}}) = -\sum_{i=1}^{5} y_i \log \hat{y}_i$$

J 对各个参数进行求导：

$$\frac{\partial J}{\partial \boldsymbol{U}} \quad \frac{\partial J}{\partial \boldsymbol{b}_2} \quad \frac{\partial J}{\partial \boldsymbol{W}} \quad \frac{\partial J}{\partial \boldsymbol{b}_1} \quad \frac{\partial J}{\partial \boldsymbol{L}_i}$$

$$\boldsymbol{U} \in \mathbb{R}^{100\times 5} \quad \boldsymbol{b}_2 \in \mathbb{R}^{5} \quad \boldsymbol{W} \in \mathbb{R}^{150\times 100} \quad \boldsymbol{b}_1 \in \mathbb{R}^{100} \quad \boldsymbol{L}_i \in \mathbb{R}^{50}$$

得到如下求导公式：

$$\begin{aligned}
\frac{\partial J}{\partial \boldsymbol{U}} &= \boldsymbol{h}^T(\boldsymbol{y} - \hat{\boldsymbol{y}}) \\
\frac{\partial J}{\partial \boldsymbol{b}_2} &= (\boldsymbol{y} - \hat{\boldsymbol{y}}) \\
\frac{\partial J}{\partial \boldsymbol{h}} &= (\boldsymbol{y} - \hat{\boldsymbol{y}})\boldsymbol{U}^T \\
\frac{\partial J}{\partial \boldsymbol{W}} &= (\boldsymbol{x}^{(t)})^T \left(\frac{\partial J}{\partial \boldsymbol{h}} \odot \tanh'(2(\boldsymbol{x}^{(t)}\boldsymbol{W} + \boldsymbol{b}_1)) \right) \\
\frac{\partial J}{\partial \boldsymbol{b}_1} &= \left(\frac{\partial J}{\partial \boldsymbol{h}} \odot \tanh'(2(\boldsymbol{x}^{(t)}\boldsymbol{W} + \boldsymbol{b}_1)) \right) \\
\frac{\partial J}{\partial \boldsymbol{x}^{(t)}} &= \left(\frac{\partial J}{\partial \boldsymbol{h}} \odot \tanh'(2(\boldsymbol{x}^{(t)}\boldsymbol{W} + \boldsymbol{b}_1)) \right) \boldsymbol{W}^T
\end{aligned}$$

在 TensorFlow 中求导是自动实现的，这里用 Adam 优化算法更新梯度，不断地迭代，使得 loss 越来越小直至收敛。