BiLSTM and CRF with Fine-Tuned BERT for Named Entity Recognition

Explained

Overview

Given a sequence $X = \{x_1, \cdots, x_n\}$, and a label sequence $y = \{\mathbf{start}(y_0), y_1, \cdots, y_n, \mathbf{end}(y_{n+1})\}$, where $y_i \in \mathcal{Y}$, and we denote the size of $|\mathcal{Y}| = m$.

The **score** of such label sequence is

$$\text{Score}(X, y) = \sum_{i=0}^{n} A[y_{i+1}][y_i] + \sum_{i=1}^{n} E[x_i][y_i],$$

where $A \in \mathbb{R}^{m \times m}$ is the transmition matrix, the i, j entry $A_{i,j}$ is the unnormalized probability of transfering to label i from label j, $E \in \mathbb{R}^{n \times m}$ is the emission matrix, the i, j entry $E_{i,j}$ is the unnormalized probability of i-th word being labeled with j. Both of the matrix are contructed by **trainable parameters** λ.

Afterwards, since we want a **probability distribution** $p(y|X, \lambda)$, the score should be normalized as:

$$p(y|X, \lambda) = \frac{\exp(\text{Score}(X, y))}{\sum_{\tilde{y} \in Y} \exp(\text{Score}(X, \tilde{y}))}$$

Then, denote the ground-truth label sequence of X is $\hat{y}$, we train the model to maximize $p(\hat{y}|X, \lambda)$, which is equivalent to **minimizing its negative log likelihood**:

$$\begin{aligned} \mathcal{L} &= -\log p(\hat{y}|X, \lambda) \\ &= \log \sum_{\tilde{y} \in Y} \exp(\text{Score}(X, \tilde{y})) - \text{Score}(X, \hat{y}) \end{aligned}$$

In terms of **inference**, given the unlabeled sequence X, we can simply select y with the biggest $p(y|X, \lambda)$ to be its label sequence, formally:

$$y = \arg\max_{y \in Y}(p(y|X, \lambda))$$

Function

- `_forward_arg`
  Based on

$$\log(\sum_{x \in \mathcal{X}} \sum_{y \in \mathcal{Y}} \exp(x + y)) = \log(\sum_{y \in \mathcal{Y}} \exp(\log(\sum_{x \in \mathcal{X}} \exp(x)) + y)),$$

we denote $s(X, y, i) = \text{Score}(X[0:i], y[0:i])$, the following equation can be derived:

$$
\begin{aligned}
\log \sum_{y \in Y} \exp(\text{Score}(X, y)) &= \log \sum_{y \in Y} \exp(s(X, y, n)) \\
&= \log(\sum_{y_0 \in \mathcal{Y}} \sum_{y_1 \in \mathcal{Y}} \cdots \sum_{y_n \in \mathcal{Y}} \exp(s(X, y, n)) \\
&= \log(\sum_{y_0 \in \mathcal{Y}} \sum_{y_1 \in \mathcal{Y}} \cdots \sum_{y_{n-1} \in \mathcal{Y}} \exp(\log(\sum_{y_n \in \mathcal{Y}} \exp(s(X, y, n-1))) + \text{Score}(X[n], y[n])) \\
&= \log(\sum_{y[0:n-1] \in Y[0:n-1]} \exp(\log(\sum_{y_n \in \mathcal{Y}} \exp(s(X, y, n-1))) + \text{Score}(X[n], y[n]))
\end{aligned}
$$

- the equation reveals the **Optimal Substructure** of computing the score of every possible tag sequence
- therefore, we can use dynamic programming to calculate $\sum_{\tilde{y} \in Y} \exp(\text{Score}(X, \tilde{y}))$ by iteratively computing the **log-exp-sum** of the prefix string.