京东智联云 & 贪心科技 GREEDY TECHNOLOGY

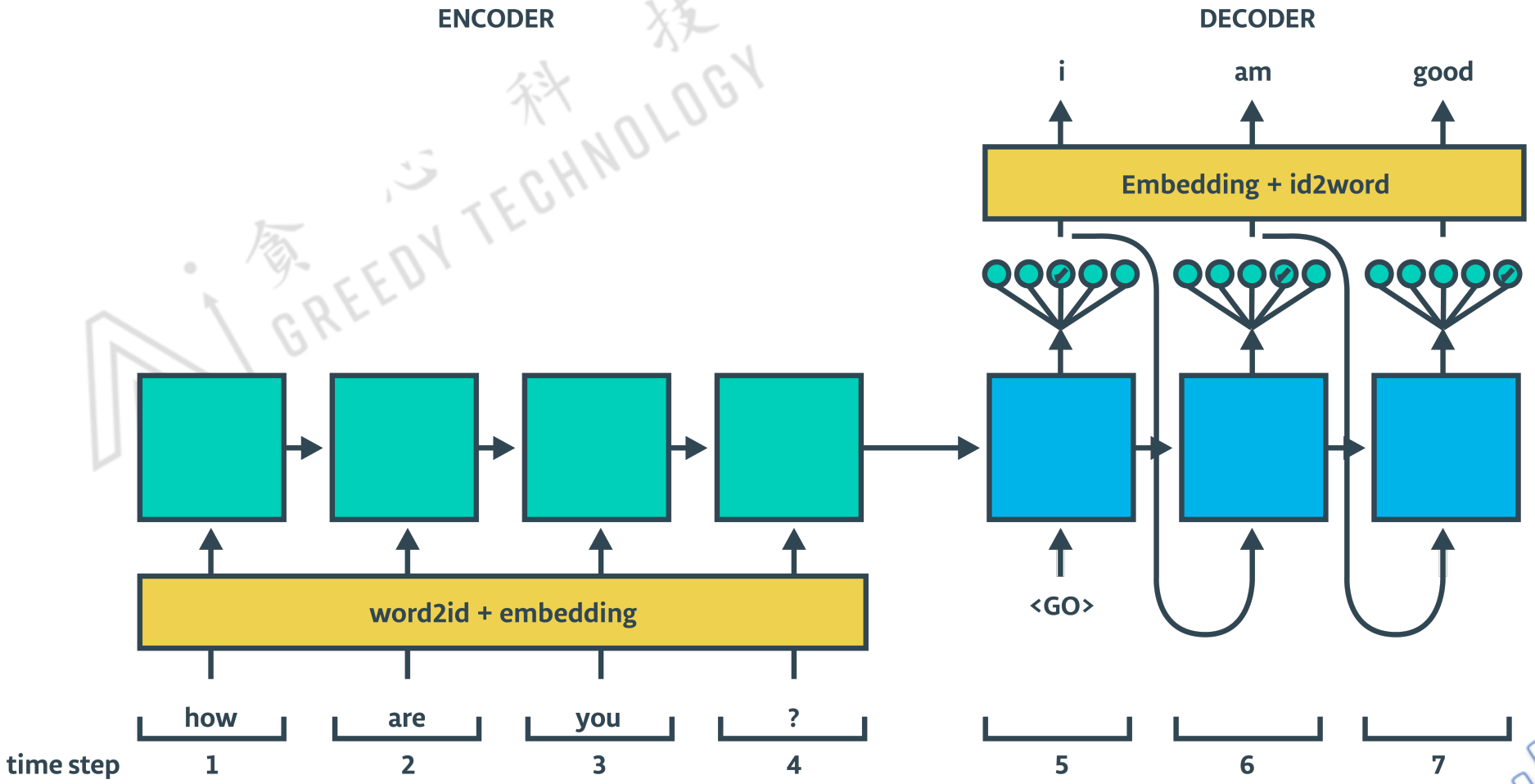
Machine Translation

Pytorch seq2seq + attention

NeuHub AI Links All

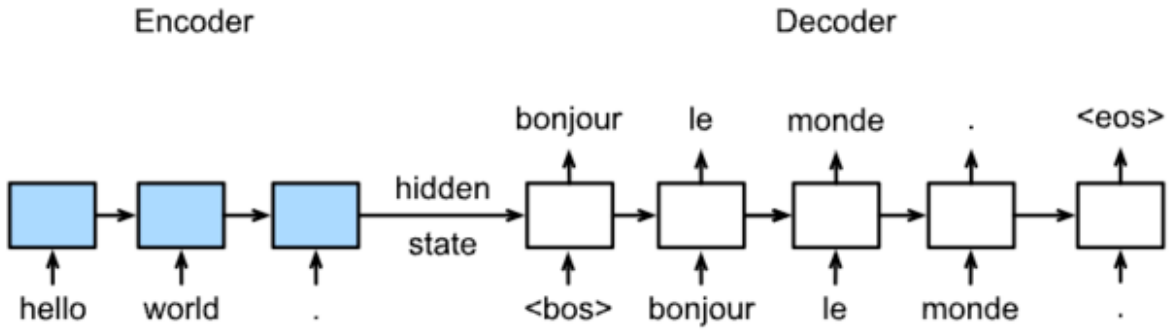
Goals to take away:

- 掌握 seq2seq于机器翻译中的应用
- 掌握seq2seq + attention 于机器翻译中的原理
- 掌握seq2seq+att 的模型框架及attention 计算
- 掌握python 实现bleu， NLTK调包使用bleu
- 掌握seq2seq+attention 在机器翻译中的Demo.

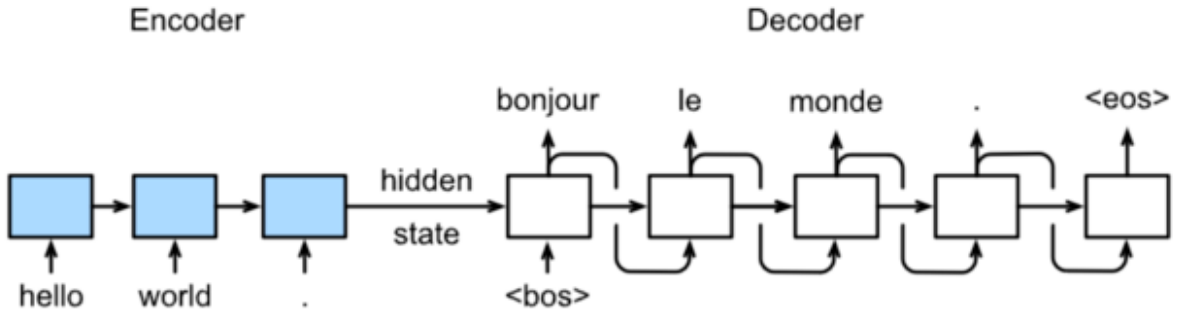


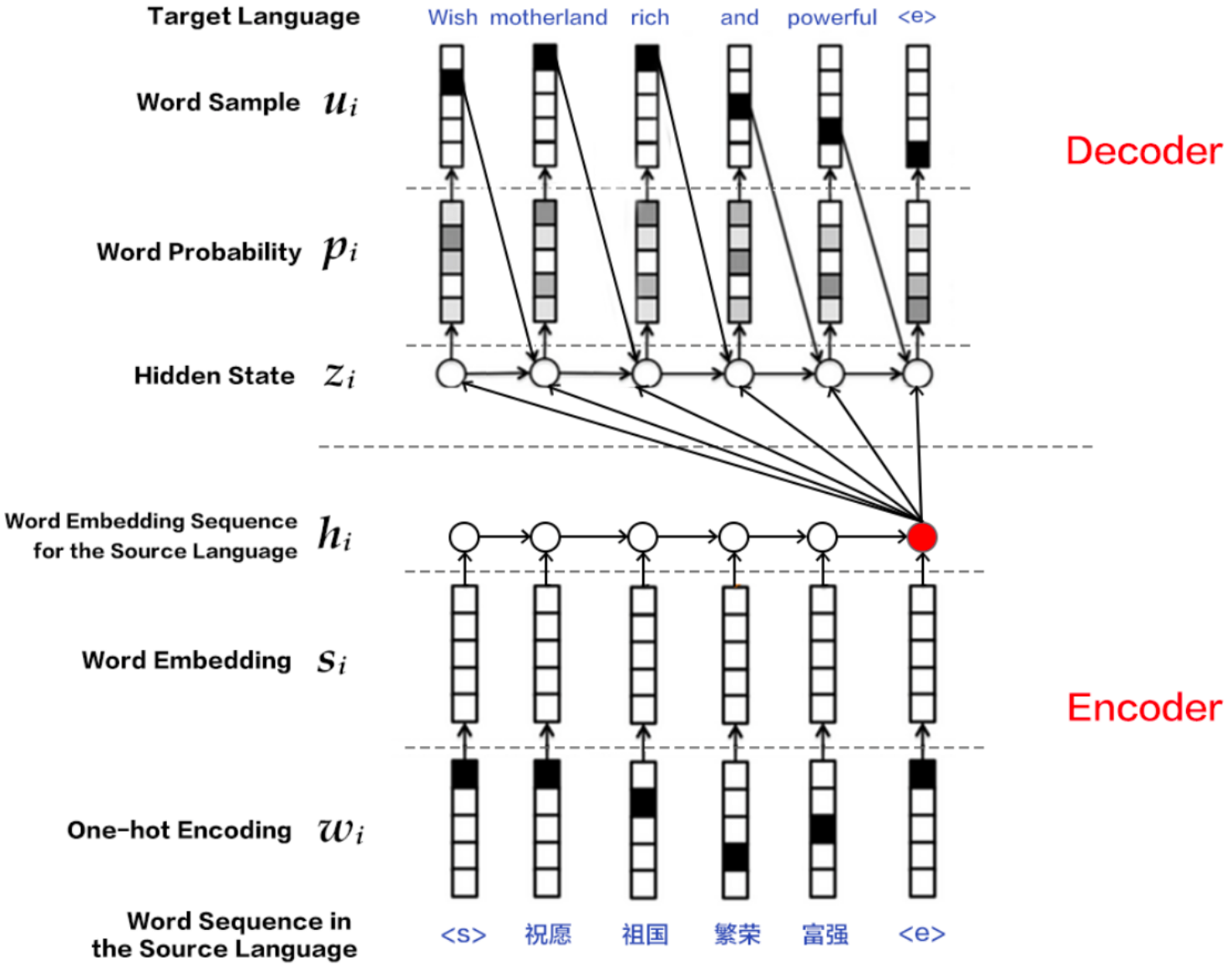
模型：

训练



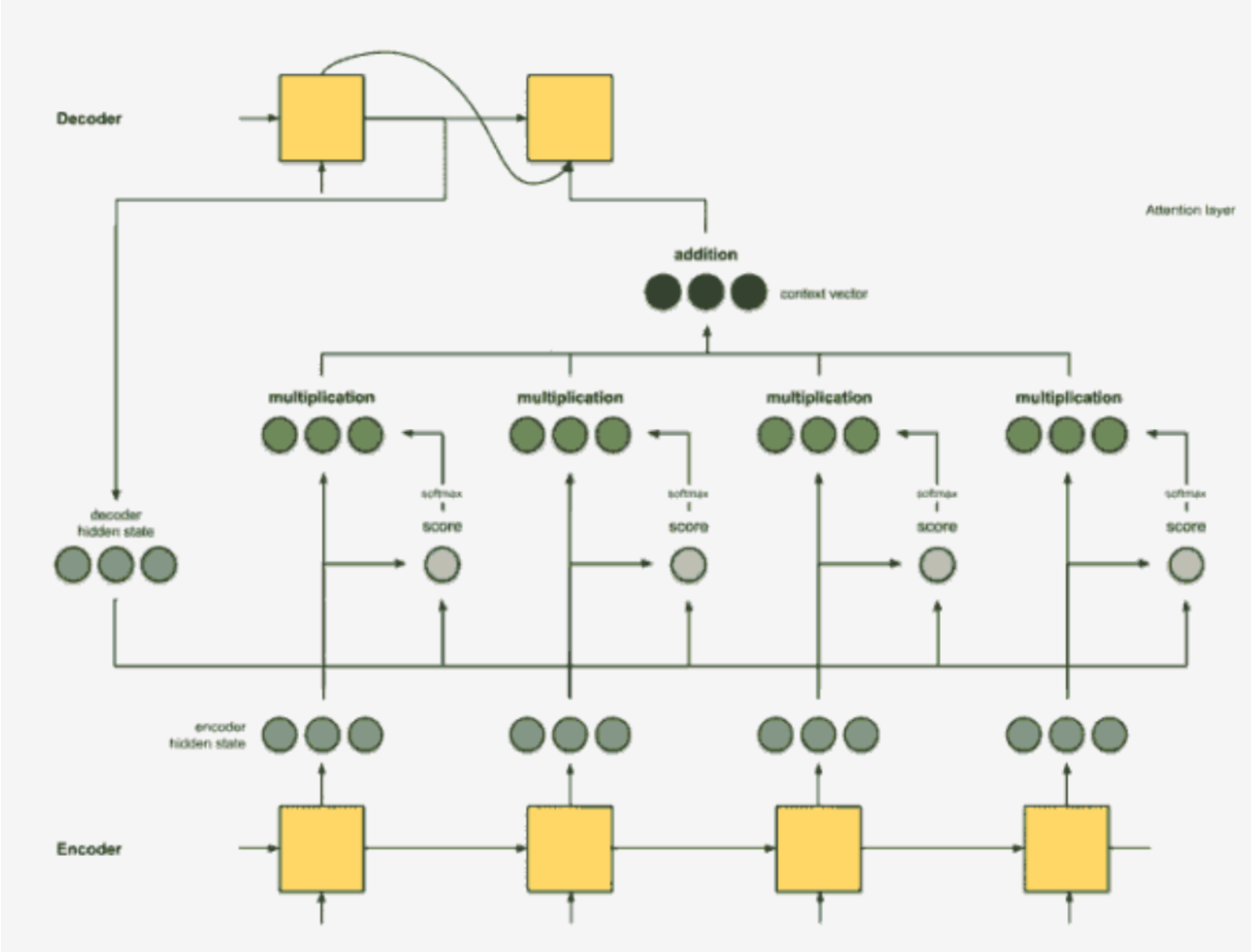
预测





Decoder

Encoder



京东智联云 & 贪心科技 GREEDY TECHNOLOGY

NeuHub
AI Links All

- 点积 attention score
  (Basic dot-product attention) :

这个就是我们常见的attention score计算方式

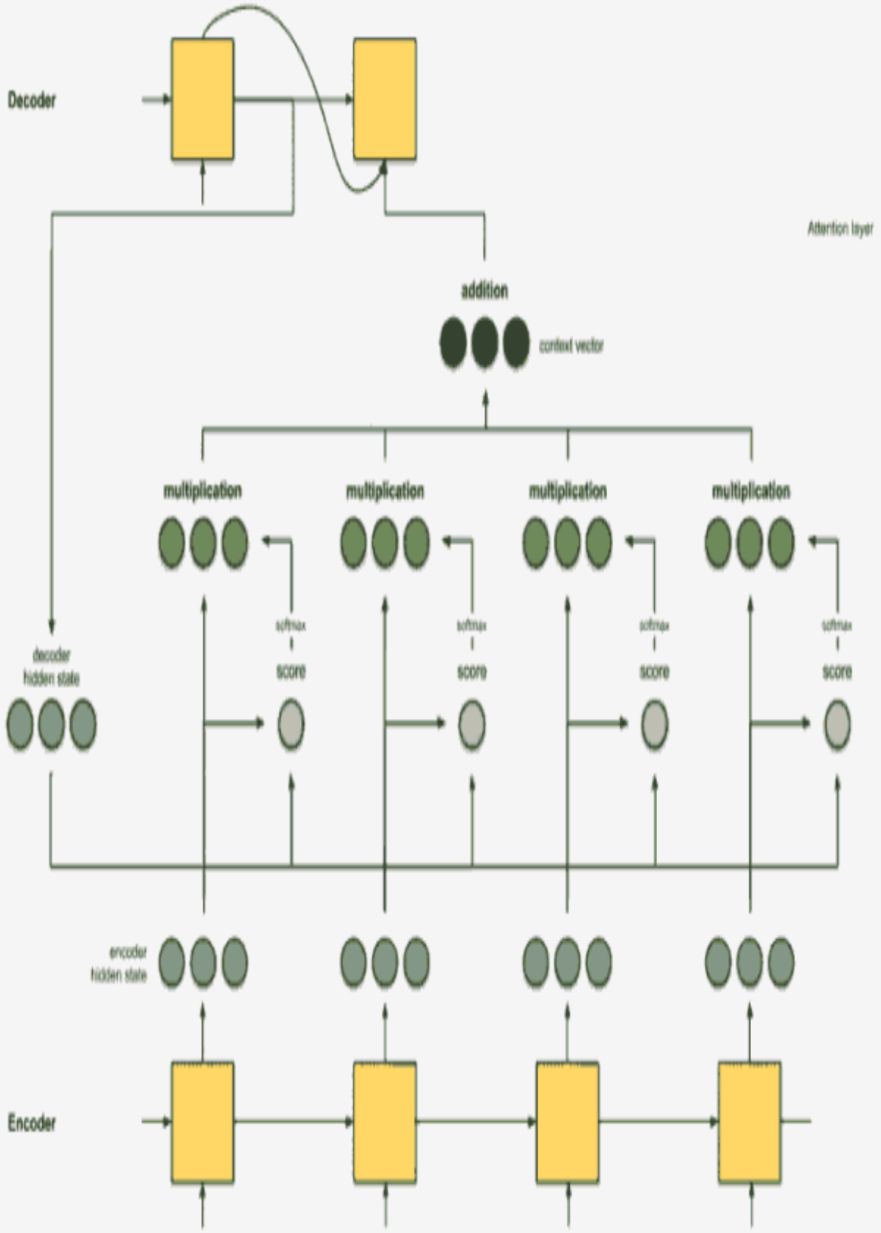
$$\boldsymbol{e}_i = \boldsymbol{s}^T \boldsymbol{h}_i \in \mathbb{R}$$

- 乘法 attention score
  (Multiplicative attention) :

$$\boldsymbol{e}_i = \boldsymbol{s}^T \boldsymbol{W} \boldsymbol{h}_i \in \mathbb{R}$$

- 加法 attention score
  (Additive attention:

$$\boldsymbol{e}_i = \boldsymbol{v}^T \tanh(\boldsymbol{W}_1 \boldsymbol{h}_i + \boldsymbol{W}_2 \boldsymbol{s}) \in \mathbb{R}$$



京东智联云 & 贪心科技 GREEDY TECHNOLOGY

- 首先我们利用RNN结构得到encoder中的hidden state $(h_1, h_2, \ldots, h_T)$ ，
- 假设当前decoder的hidden state 是 s_{t-1} ，我们可以计算每一个输入位置j与当前输出位置的关联性， $e_{tj} = a(s_{t-1}, h_j)$ ，写成相应的向量形式即为 $\overrightarrow{e_t} = (a(s_{t-1}, h_1), \ldots, a(s_{t-1}, h_T))$ ，其中 a 是一种相关性的算符，例如常见的有点乘形式 $\overrightarrow{e_t} = \overrightarrow{s_{t-1}}^T \vec{h}$ ，加权点乘 $\overrightarrow{e_t} = \overrightarrow{s_{t-1}}^T W\vec{h}$ ，加和 $\overrightarrow{e_t} = \vec{v}^T tanh(W_1\vec{h} + W_2\overrightarrow{s_{t-1}})$ 等等。
- 对于 $\overrightarrow{e_t}$ 进行softmax操作将其normalize得到attention的分布， $\overrightarrow{\alpha_t} = softmax(\overrightarrow{e_t})$ ，展开形式为 $\alpha_{tj} = \dfrac{exp(e_{tj})}{\sum_{k=1}^{T} exp(e_{tk})}$
- 利用 $\overrightarrow{\alpha_t}$ 我们可以进行加权求和得到相应的context vector $\overrightarrow{c_t} = \sum_{j=1}^{T} \alpha_{tj} h_j$
- 由此，我们可以计算decoder的下一个hidden state $s_t = f(s_{t-1}, y_{t-1}, c_t)$ 以及该位置的输出 $p(y_t|y_1, \ldots, y_{t-1}, \vec{x}) = g(y_{i-1}, s_i, c_i)$ 。

Review : Bleu

$$BLEU = BP \times exp(\sum_{n=1}^{N} W_n log P_n)$$

$$BP = \begin{cases} 1 & if \quad l_c > ls \\ e^{1-\frac{l_s}{l_c}} & if \quad l_c <= l_s \end{cases}$$

Python 来实现！

$$Count_{clip} = \min(Count, Max\_Ref\_Count)$$

$$P_n = \frac{\sum_i \sum_k min(h_k(c_i), max_{j \in m} h_k(s_{ij}))}{\sum_i \sum_k min(h_k(c_i))}$$

Reference

keon/**seq2seq**: https://github.com/keon/seq2seq

N_direction * n_layers