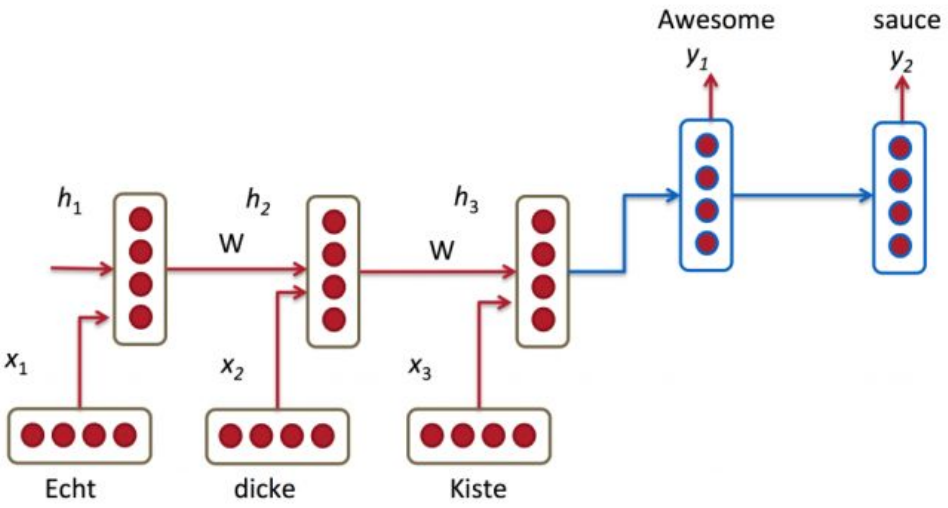
Attention

[[TOC]]

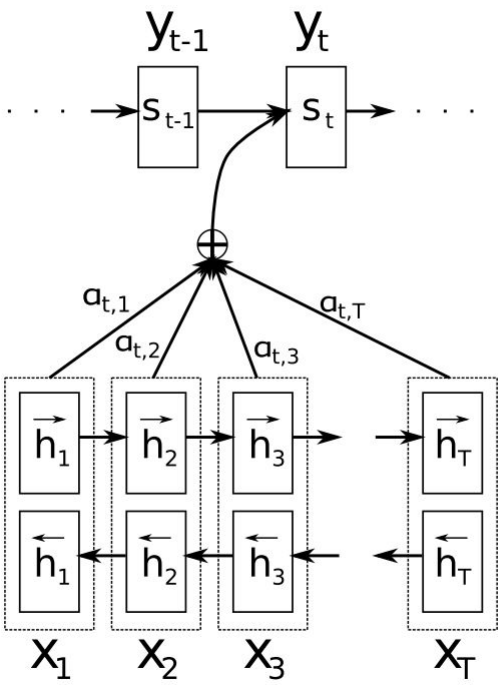
0.资料网址：

- 知乎好的博文
- 一个非常好的NLP学习资料库
- 面经

1. attention

- 在原始的seq2seq中，decoder模型只能利用encoder的最后一个Hidden state作为上一hidden state
  - 

    - 随着序列的边长，所有的信息都压缩到了一个向量中，序列增长信息丢失严重
- 因此，使用attention机制

- 

  - 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 $\alpha_{tj} = \frac{exp(e_{tj})}{\sum_{k=1}^{T} exp(e_{tk})}$
  - 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 $p(y_t | y_1, \ldots, y_{t-1}, \vec{x}) = g(y_{i-1}, s_i, c_i)$ 。

    这里关键的操作是计算encoder与decoder state之间的关联性的权重，得到Attention分布，从而对于当前输出位置得到比较重要的输入位置的权重，在预测输出时相应的会占较大的比重。

    通过Attention机制的引入，我们打破了只能利用encoder最终单一向量结果的限制，从而使模型可以集中在所有对于下一个目标单词重要的输入信息上，使模型效果得到极大的改善。还有一个优点是，我们通过观察attention 权重矩阵的变化，可以更好地知

总结：

- 在Decoder端，隐状态由 上一隐状态、上一输出、输入序列每个隐状态的加权和 决定
  - 输出由 当前隐状态 和 上一输出 决定
  - 为了计算 输入序列每个隐状态的加权和， 我们需要计算 上一隐状态 和 输入序列的所有隐状态的 相关性，而后softmax得到一个归一化的加权
- 的