基础课程 2：语言模型与神经网络

目录

- 语言模型
- 循环神经网络
- Attention机制
- Transformer网络

1. 语言模型

CHATGPT

自然语言是一种上下文相关的信息表达和信息传递的方式

- 语言模型是衡量一句话出现在自然语言中的概率的模型
- 语言模型的核心在于根据前文预测下一个词出现的概率
- 如何计算语言模型的概率？

$$P(s) = P(w_1) \times P(w_2|w_1) \times P(w_3|w_1, w_2) \ldots \times P(w_n|w_1, \ldots, w_{n-1})$$

易算　不麻烦　还可以　无法估算

- 马尔可夫假设：当前词出现的概率只和它前面的k个词相关

$$\begin{aligned} P(w_i|w_1, \ldots, w_{i-1}) &= P(w_i|w_{i-k}, \ldots, w_{i-1}) && \text{马尔可夫假设} \\ &= P(w_i) && \text{k=0, Unigram Model} \\ &= P(w_i|w_{i-1}) && \text{k=1, Bigram Model} \\ &= P(w_i|w_{i-2}, w_{i-1}) && \text{k=2, Trigram Model} \end{aligned}$$

- 用频率估计概率

$$\begin{aligned} P(w_i|w_1, \ldots, w_{i-1}) &= P(w_i|w_{i-k}, \ldots, w_{i-1}) && \text{马尔可夫假设} \\ &= \frac{P(w_{i-k}, \ldots, w_{i-1}, w_i)}{P(w_{i-k}, \ldots, w_{i-1})} && \text{条件概率} \\ &\approx \frac{count(w_{i-k}, \ldots, w_{i-1}, w_i) \Big/ count(all\ grams)}{count(w_{i-k}, \ldots, w_{i-1}) \Big/ count(all\ grams)} \\ &= \frac{count(w_{i-k}, \ldots, w_{i-1}, w_i)}{count(w_{i-k}, \ldots, w_{i-1})} && \text{概率估计} \end{aligned}$$

语言模型评价方法：

- 困惑度 (Perplexity)
  - 用来度量一个概率分布或概率模型预测样本的好坏程度；
  - 可以用来比较两个概率模型，低困惑度的概率模型能更好地预测样本。

在测试数据上

$$\begin{aligned} Perplexity\ (s) &= 2^{H(s)} \\ &= 2^{-\frac{1}{n}\log_2 P(w_1, \ldots, w_n)} \\ &= 2^{\log_2 P(w_1, \ldots, w_n)^{-\frac{1}{n}}} \\ &= P(w_1, \ldots, w_n)^{-\frac{1}{n}} \\ &= \sqrt[n]{1/P(w_1, \ldots, w_n)} \end{aligned}$$

PPL

参数规模问题：

$$P(w_i|w_1,\dots,w_{i-1}) = P(w_i|w_{i-k},\dots,w_{i-1})$$

$$= \frac{P(w_{i-k},\dots,w_{i-1},w_i)}{P(w_{i-k},\dots,w_{i-1})}$$

$$\approx \frac{count(w_{i-k},\dots,w_{i-1},w_i)}{count(w_{i-k},\dots,w_{i-1})}$$

随着k的增大，参数数目呈指数增长，无法存储

k=1 参数量：$|V|^2$

k=2 参数量：$|V|^3$

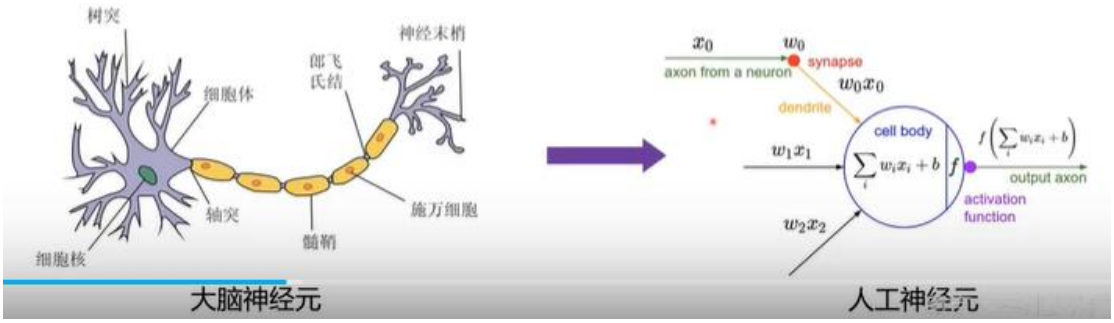
.....

k=n-1 参数量：$|V|^n$

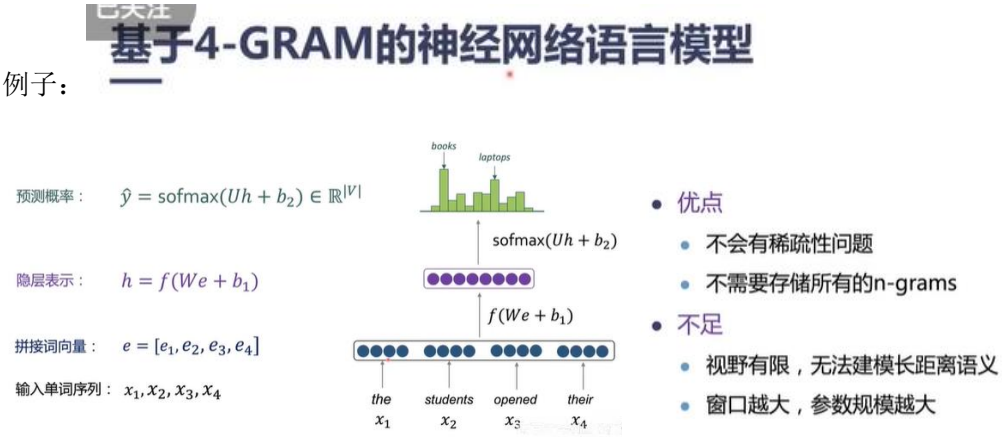
2. 循环神经网络

人工神经网络：

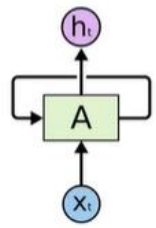
- 人工神经网络（Artificial Neural Network，ANN）从信息处理角度对人脑神经元网络进行抽象， 建立某种算法数学模型，从而达到处理信息的目的。



例子：

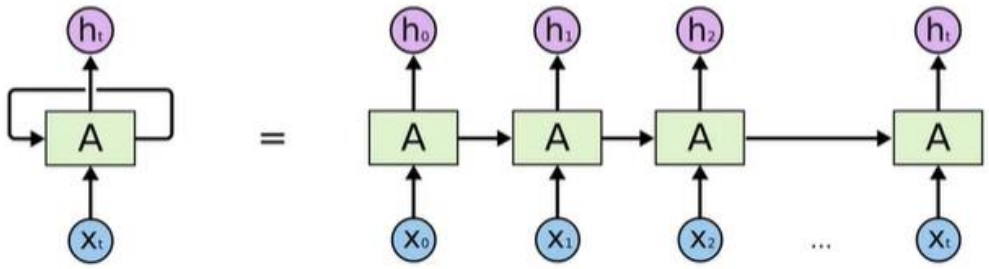


循环神经网络：



- 循环神经网络（Recurrent Neural Network，RNN）
  - 重复使用隐层参数
  - 可处理任意序列长度

$$h_t = f(W_h h_{t-1} + W_x x_t + b)$$



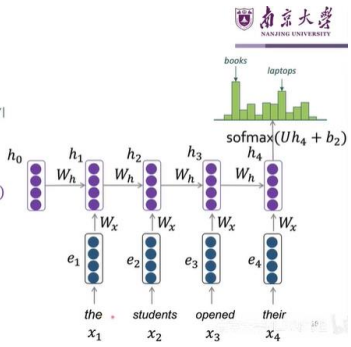
循环神经网络语言模型

预测概率： $\hat{y} = \text{sofmax}(Uh + b_2) \in \mathbb{R}^{|V|}$

隐层表示： $h_t = f(W_h h_{t-1} + W_x e_t + b)$

输入词向量序列： e_1, e_2, e_3, e_4

输入单词序列： x_1, x_2, x_3, x_4



- 优点
  - 能够处理任意长度序列
  - 能够使用历史信息
  - 模型参数量不随序列长度增加
- 不足
  - 逐步计算，速度较慢
  - 长期依赖问题

RNN-LM模型训练

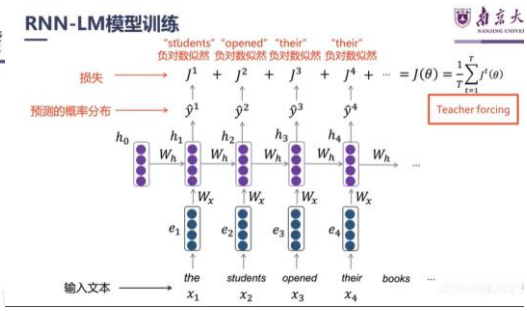
- 给定长度为T的输入文本：$x_1, \cdots, x_T$；
- 将文本输入到RNN-LM，计算每一步预测的单词分布$\hat{y}^t$；
- 计算每一步预测单词的概率分布$\hat{y}^t$和真实单词y^t（one-hot向量）之间的交叉熵：

$$J^t(\theta) = \text{CE}(y^t, \hat{y}^t) = -\sum_{w \in V} y_w^t \log \hat{y}_w^t = -\log \hat{y}_{x_{t+1}}^t$$

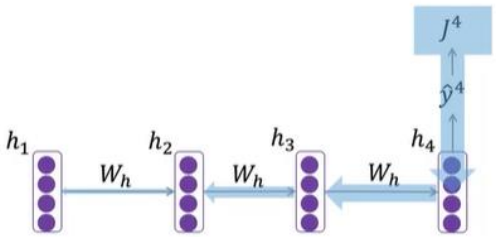
- 模型在输入文本上的训练损失为：

$$J(\theta) = \frac{1}{T}\sum_{t=1}^{T} J^t(\theta) = \frac{1}{T}\sum_{t=1}^{T} -\log \hat{y}_{x_{t+1}}^t$$

RNN-LM模型训练



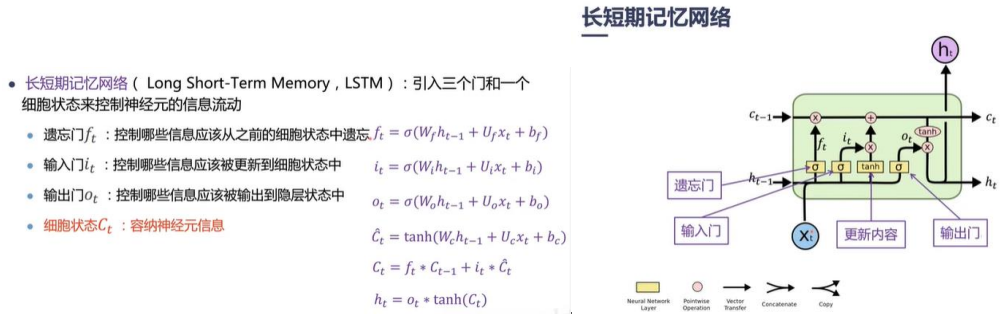
RNN：梯度爆炸，梯度消失



$$\frac{\partial J^4}{\partial h^1} = \frac{\partial h^2}{\partial h^1} \times \frac{\partial h^3}{\partial h^2} \times \frac{\partial h^4}{\partial h^3} \times \frac{\partial J^4}{\partial h^4}$$

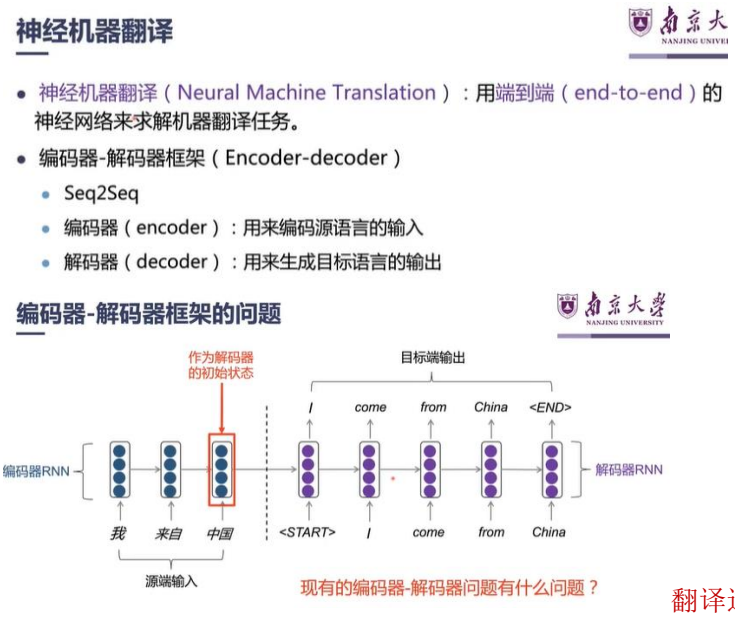
如果这些梯度很大或者很小怎么办？

长短记忆网络：



线性变换，环节远程梯度爆炸和梯度消失问题

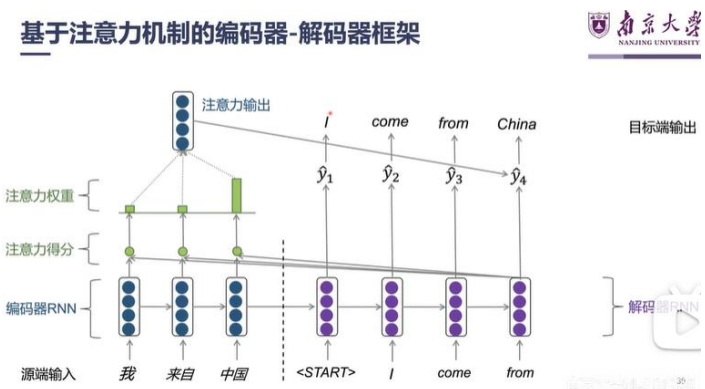
3. Attention 机制



翻译过程不具有可解释性

注意力机制：

- Attention Mechanism [Bahdanau et al., 2015]
  - 目标端解码时，直接从源端句子捕获对当前解码有帮助的信息，从而生成更相关、更更准确的解码结果
- 优点
  - 缓解RNN中的信息瓶颈问题
  - 缓解长距离依赖问题
  - 具有一定的可解释性



编码器-解码器中如何计算注意力？

- 编码器的隐层状态为$h_1, \dots, h_N \in \mathbb{R}^h$
- t时刻，解码器的隐层状态为$s_t \in \mathbb{R}^h$
- 对于t时刻，编码器隐层状态的注意力打分为：

$$e^t = [s_t^{\mathrm{T}} h_1, \dots, s_t^{\mathrm{T}} h_N] \in \mathbb{R}^N$$

- 利用softmax函数将注意力打分转换成概率化的注意力权重：

$$\alpha^t = \mathrm{softmax}(e^t) \in \mathbb{R}^N$$

- 利用注意力权重α^t得到编码器隐层状态的加权输出：

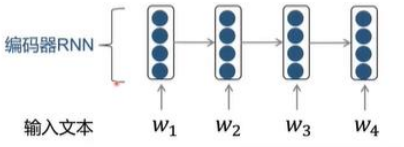
$$a^t = \sum_{i=1}^{N} \alpha_i^t h_i \in \mathbb{R}^h$$

文本分类模型的注意力可视化

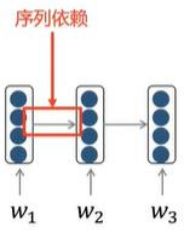
2. Transformer 网络

循环神经网络的问题

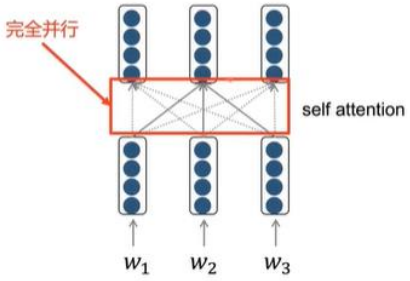
- 有限的信息交互距离
  - RNN能捕捉局部信息，但无法很好地解决长距离依赖关系（long-distance dependency）
  - 不能很好地建模序列中的非线性结构关系
- 无法并行
  - RNN的隐层状态具有序列依赖性
  - 时间消耗随序列长度的增加而增加



SELF ATTENTION



RNN网络



Attention网络

TRANSFORMER编码器

ATTENTION IS ALL YOU NEED

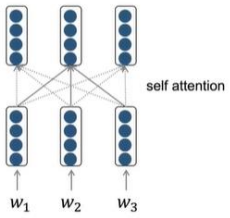
- Transformer [Vaswani et al., 2017]
  - 完全基于attention机制构建的神经网络模型
  - 直接建模输入序列的全局依赖关系
  - 并行计算

- 编码器
  - 输入编码+位置编码
  - 多头注意力机制
  - 残差连接&层正则
  - 前馈神经网络
  - 残差连接&层正则

位置编码

- 为什么需要？
  - 注意力计算：加权和
  - 无法考虑相对位置关系

$$a = \sum_{i=1}^{N} \alpha_i h_i \in \mathbb{R}^d$$



- 三角函数表示：直接根据正余弦函数计算位置编码
  - 不需要从头学习，直接计算得出

$$p_i = \begin{bmatrix} \sin(i/10000^{2*1/d}) \\ \cos(i/10000^{2*1/d}) \\ \vdots \\ \sin(i/10000^{2*\frac{d}{2}/d}) \\ \cos(i/10000^{2*\frac{d}{2}/d}) \end{bmatrix}$$

- 从头学习：随机初始化位置编码p_i，并跟随网络一起训练
  - 能更好地拟合数据

自注意力机制

- 计算attention所需要的queries，keys，values

$$Q = XW^Q \quad K = XW^K \quad V = XW^V$$

- 根据queries和keys计算attention打分E

$$E = QK^{\mathrm{T}}$$

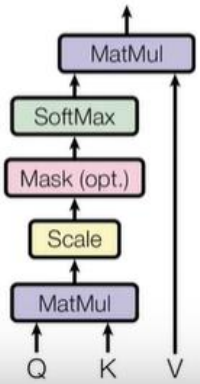
- 计算attention权重A

$$A = \mathrm{softmax}(\frac{E}{\sqrt{d_k}})$$

- 根据attention权重A和values计算attention输出O

$$O = AV$$

Scaled Dot-Product Attention

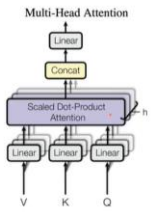


多头自注意力机制

- 并行地计算多个自注意力过程，并拼接输出结果

$$O_i = \mathrm{softmax}\left(\frac{Q_i K_i^T}{\sqrt{d_k}}\right) V_i$$

$$O = [O_1, \ldots, O_M]$$



残差连接

- Residual connections[He et al., 2016]
  - 将浅层网络和深层网络相连，有利于梯度回传
  - 使深处网络的训练变得更加容易

$$X^l = MultiHeadAttn(X^{l-1}) + X^{l-1}$$

前馈网络

- 两层前馈神经网络

TRANSFORMER解码器

- Cross Attention
  - 解码时需要关注源端信息
- Masked Attention
  - 解码时（训练）不应该看到未来的信息