第6章：语言模型

课件制作：曲彦儒、朱耀明、钱利华、屠可伟
讲解人：曲彦儒、朱耀明、钱利华

概率语言模型

目标

- 计算一个句子（一串单词组成的序列）的概率

$$P(w_1, w_2, w_3, \cdots, w_n)$$

应用

应用

- 在机器翻译里面，遴选更常见的词汇组合

$$P(\textbf{high}\ \text{winds tonight}) > P(\textbf{large}\ \text{winds tonight})$$

- 应用于句子的拼写纠错
  - The office is about 15 minuets from my home.

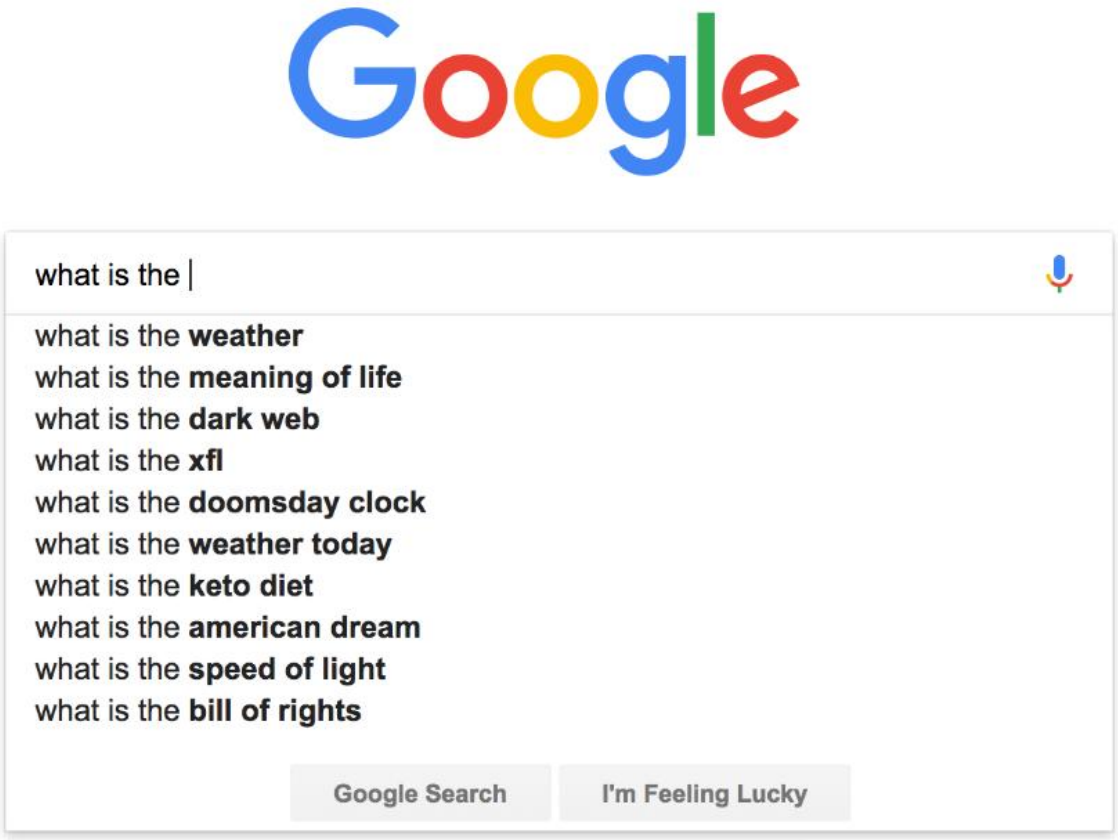
$$P(\text{about 15 minutes from}) > P(\text{about 15}\ \textbf{minuets}\ \text{from})$$

- 帮助语音识别

$$P(\text{I saw a van}) > P(\text{eye awe of an})$$

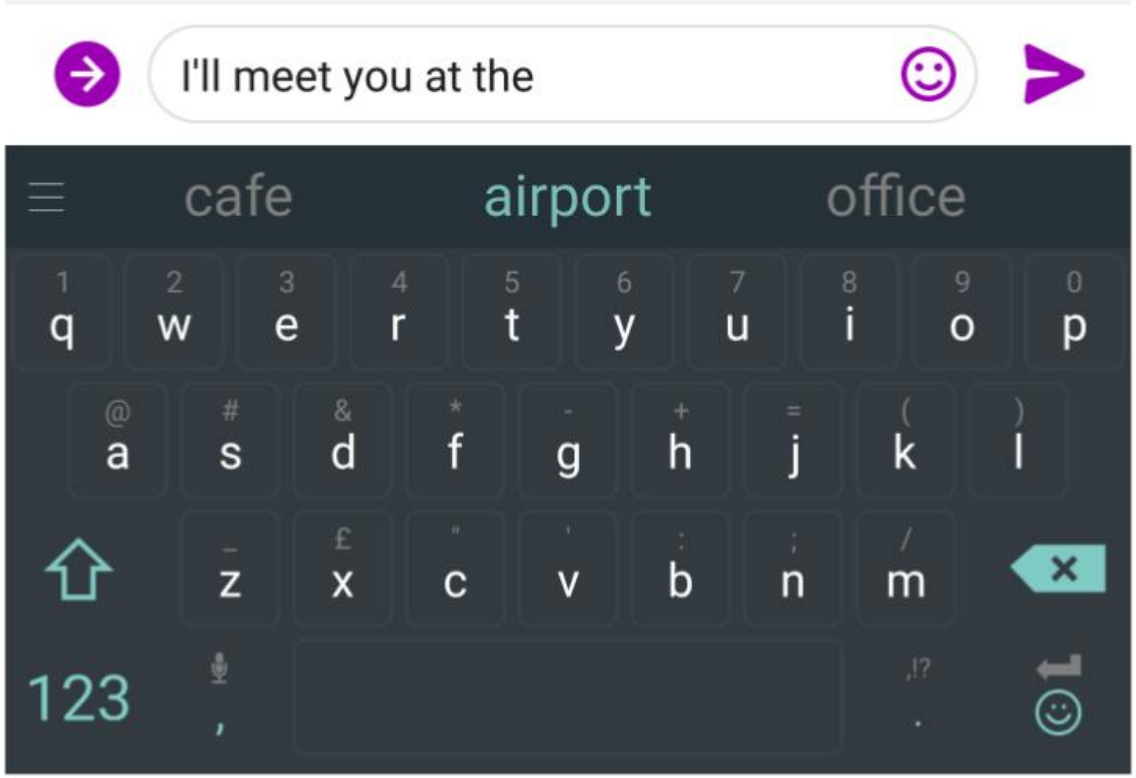
应用

应用


Google
what is the
what is the weather
what is the meaning of life
what is the dark web
what is the xfl
what is the doomsday clock
what is the weather today
what is the keto diet
what is the american dream
what is the speed of light
what is the bill of rights
Google Search
I'm Feeling Lucky

应用

应用


I'll meet you at the
cafe
airport
office
1 q 2 w 3 e 4 r 5 t 6 y 7 u 8 i 9 o 0 p
@ a # s & d * f - g + h = j (k) l
_ z £ x " c ' v : b ; n / m
123
,!?
.

6.1 概述

链式法则

- 概率中的链式法则

$$P(x_1, x_2, x_3, \ldots, x_n) = \prod_i^n P(x_i | x_1, \ldots, x_{i-1})$$

- 将链式法则用于语言模型，例如我们要计算句子的概率

“自然语言处理”

- $P(\text{“自然语言处理”}) = P(\text{自}) \times P(\text{然}|\text{自}) \times P(\text{语}|\text{自然}) \times P(\text{言}|\text{自然语}) \times P(\text{处}|\text{自然语言}) \times P(\text{理}|\text{自然语言处})$

未登录词

未登录词

- 有时句子里面会出现“未登录词”（Out Of Vocabulary = OOV words ）——这个单词没有在语料库里出现过
- 解决方案：创造一个专门表示未知单词的符号——[UNK]
  - 训练前，先创建词汇表，然后将所有不存在于词汇表里的单词替换为[UNK]。训练时，将所有[UNK]作为一个常规单词处理。
  - 使用时，将未登录词替换为[UNK]
- 替代方案：在字母层级而不是单词层级来建立语言模型

方法

设计语言模型

- *n*元语法模型
  - 应用链式法则。每个单词出现的概率基于它之前的*n*-1个单词
- 循环神经网络模型
  - 使用循环神经网络。每个单词出现的概率基于神经网络根据之前所有单词所生成的隐状态
- Transformer模型
  - 基于注意力机制

6.2 *n*元语法模型

一元语法模型

- 最简单的*n*元语法模型是一元语法模型，即*n*=1

$$P(x_1, x_2, x_3, \dots) = \prod_i P(x_i)$$

- 每个单词出现的概率是独立的
- 不考虑单词间的顺序关系（词袋模型，Bag-of-Words）

6.2 *n*元语法模型

二元语法模型

- 每个单词的出现仅依赖于上一个单词，即n=2

$$P(x_1, x_2, x_3, \dots) = \prod_i P(x_i | x_{i-1})$$

- 以此类推，可以构建三元、四元、五元语法模型等
- *n*元语法模型所做的马尔可夫假设忽视了语言中的长程依赖，但实际效果不错

最大似然估计（MLE）

- 似然度：训练数据在给定参数下的概率
- 最大化似然度 = 直接统计出现次数并归一化

$$P(x_i|x_{i-1},\ldots,x_{i-(n-1)}) = \frac{\text{count}(x_i,x_{i-1},\ldots,x_{i-(n-1)})}{\text{count}(x_{i-1},\ldots,x_{i-(n-1)})}$$

$$P(x_i|x_{i-1}) = \frac{\text{count}(x_i,x_{i-1})}{\text{count}(x_{i-1})}$$

$$P(x_i) = \frac{\text{count}(x_i)}{\#word}$$

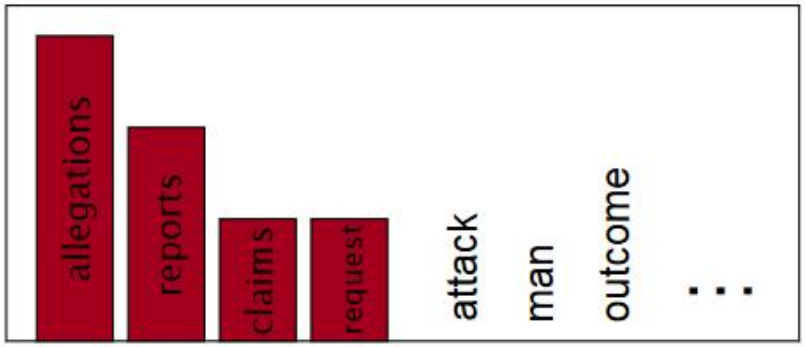
MLE的问题

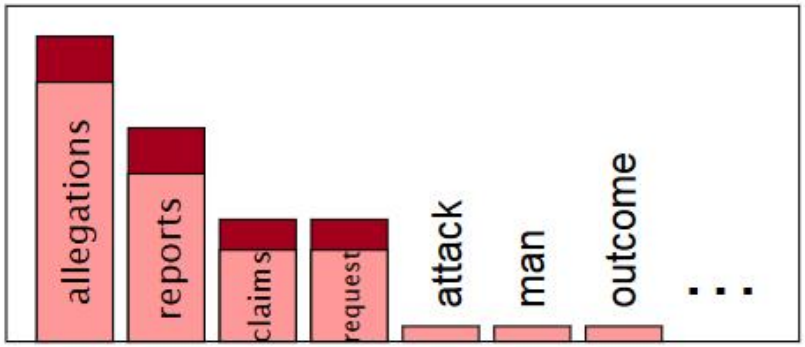
- 数据稀疏：n较大时，绝大部分n-gram都是没有被观测到的，即便它们是合乎语法的
  - 训练集
    - denied the allegations
    - denied the reports
    - denied the claims
    - denied the request
  - 测试集
    - denied the offer
  - 但是 $P(\text{offer}|\text{denied the}) = 0$

MLE的问题

平滑方法（Smoothing）

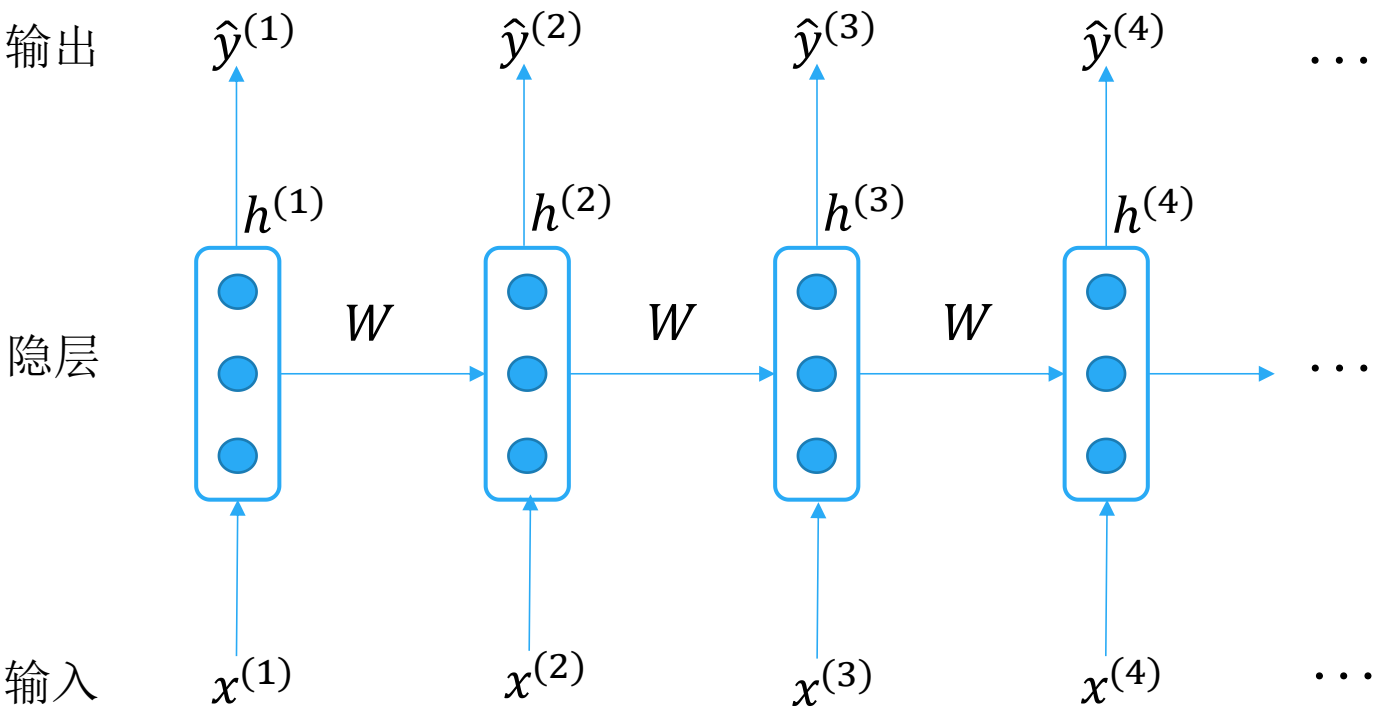
- 最简单的方法：在归一化前，对每个计数加上一个很小的值 $\lambda > 0$
- 例：P(x | denied the)
- 原始的统计信息（稀疏）
  - 3 allegations
  - 2 reports
  - 1 claims
  - 1 request
- 平滑后的统计信息
  - 3.1 allegations
  - 2.1 reports
  - 1.1 claims
  - 1.1 request
  - 0.1 for everything else





6.3 循环神经网络

- 循环神经网络（Recurrent neural network，简称RNN）是一类处理序列数据的神经网络
- 核心思想：每一时刻重复使用权重矩阵



RNN语言模型

输出分布：

$\hat{\mathrm{y}}^t = \mathrm{softmax}(Uh^t + b_2) \in R^{|V|}$

隐状态：

$h^t = \sigma(W_h h^{t-1} + W_e e^t + b_1)$

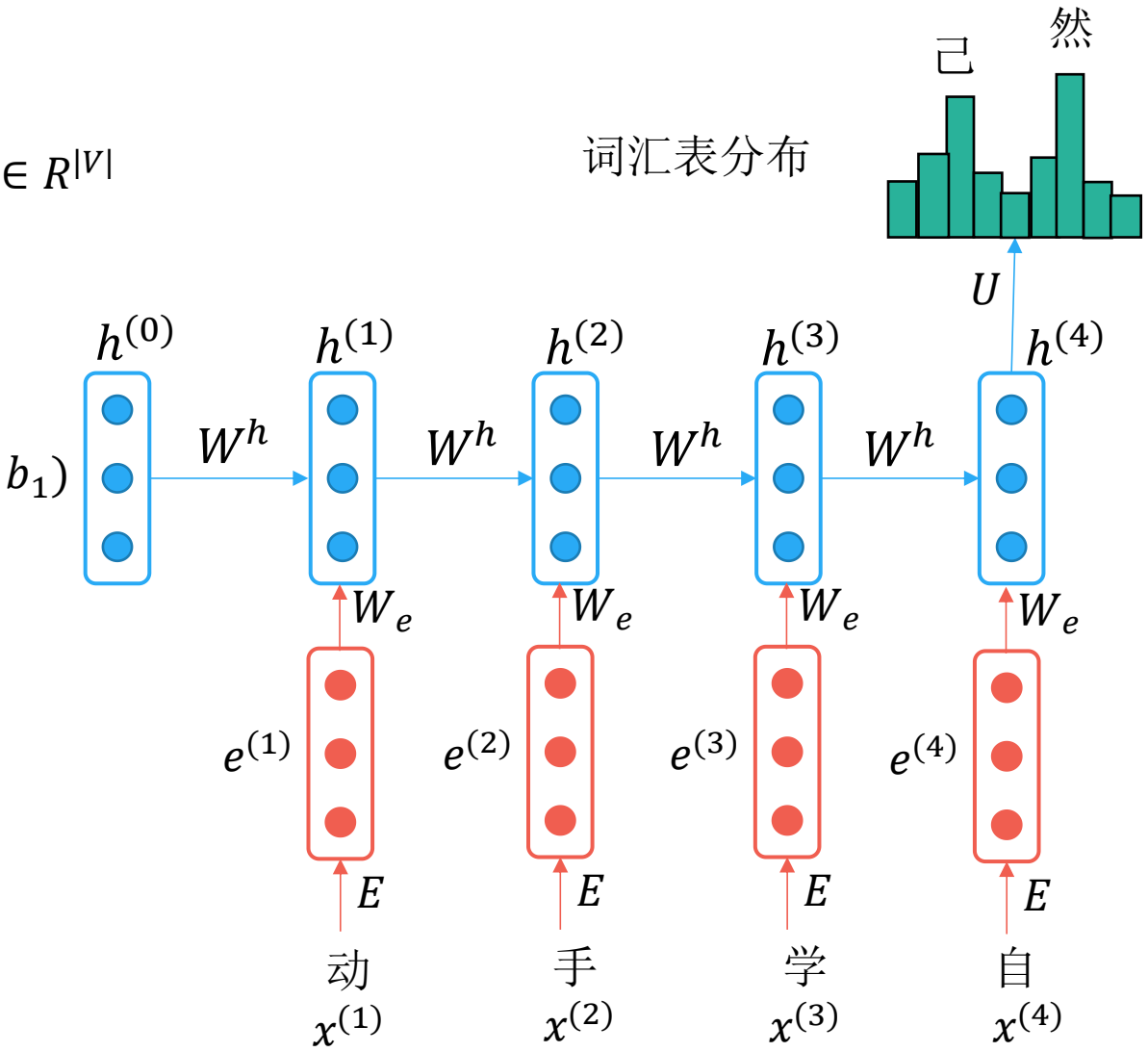
h^0是初始隐状态

词嵌入：

$e^t = Ex^t$

词/独热编码向量：

$x^t \in R^{|V|}$



RNN语言模型

优点

- RNN语言模型不再是一个*n*元语法（马尔可夫）模型
  - 每一步的预测都基于之前所有的历史信息
- 模型大小比较适中
  - 比典型的*n*元语法模型要小

RNN语言模型

训练

- 最大似然估计（Maximum Likelihood Estimation）
- 第t步的损失函数是预测的概率分布$\hat{y}^t$和真实的第t+1个词y^t(即x_{t+1}独热编码)之间的交叉熵（Cross Entropy）

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

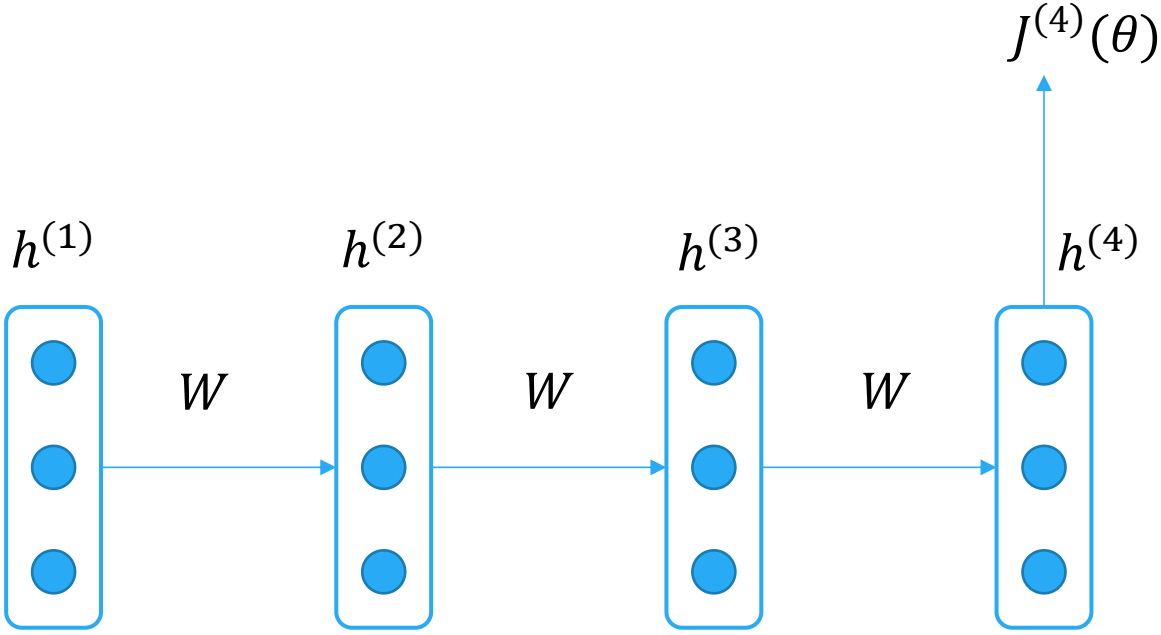
- 对每一步的损失函数求平均得到总的损失函数

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

- 优化方法：随机梯度下降（SGD）

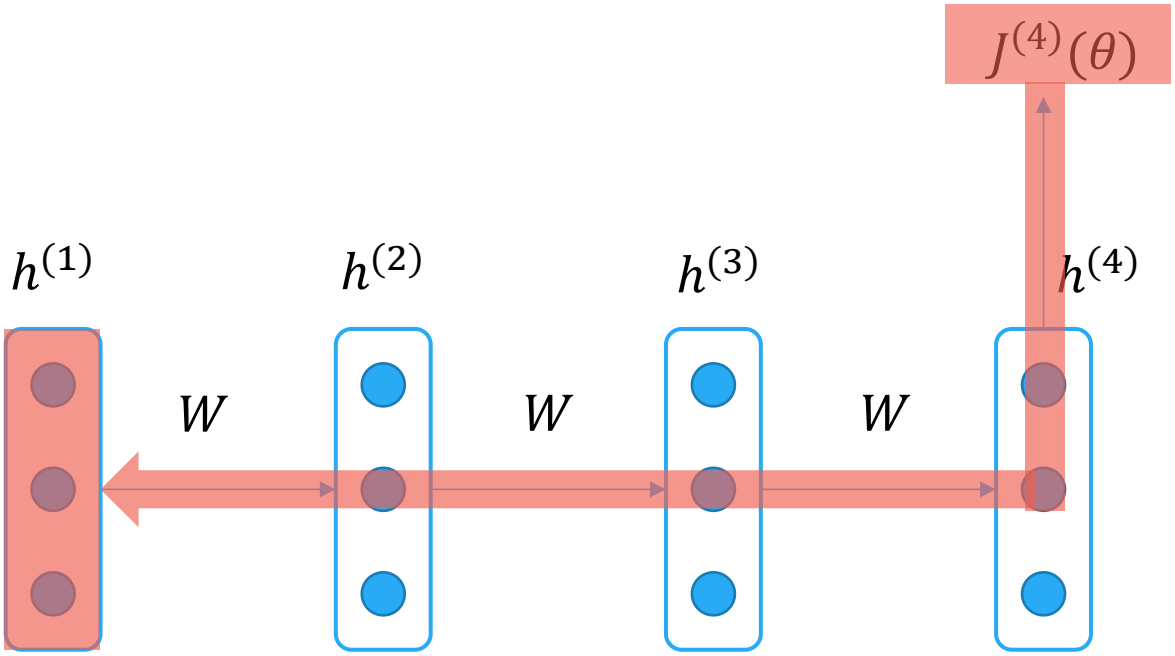
梯度消失

示例



梯度消失

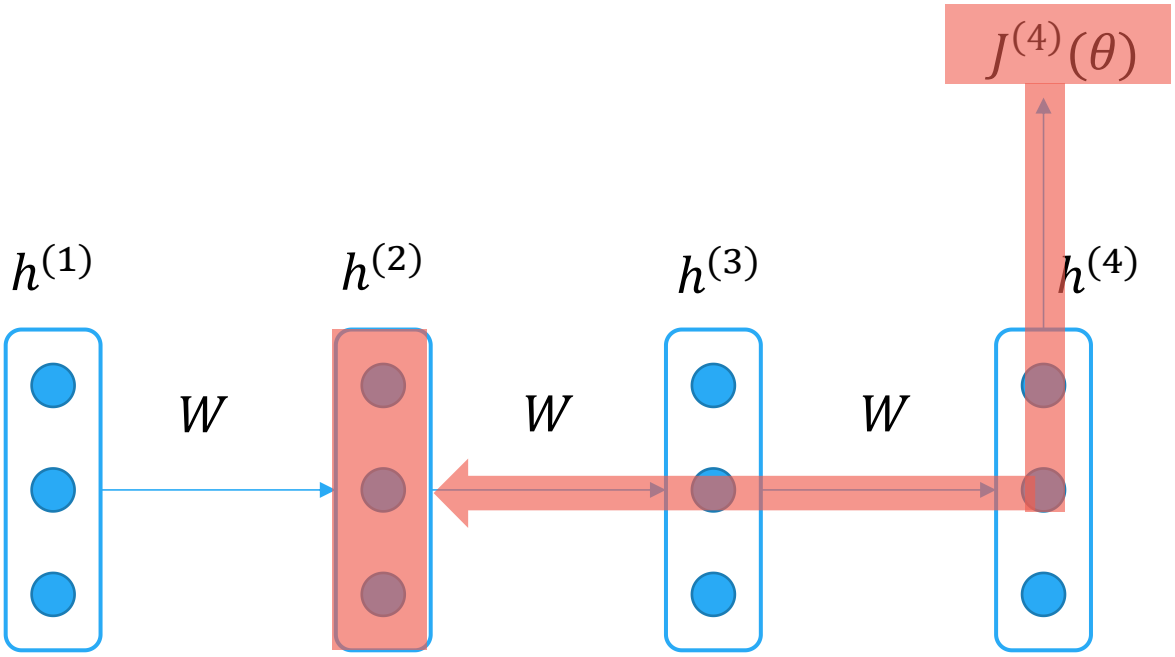
示例



$$\frac{\partial J^{(4)}}{\partial h^{(1)}} = ?$$

梯度消失

示例

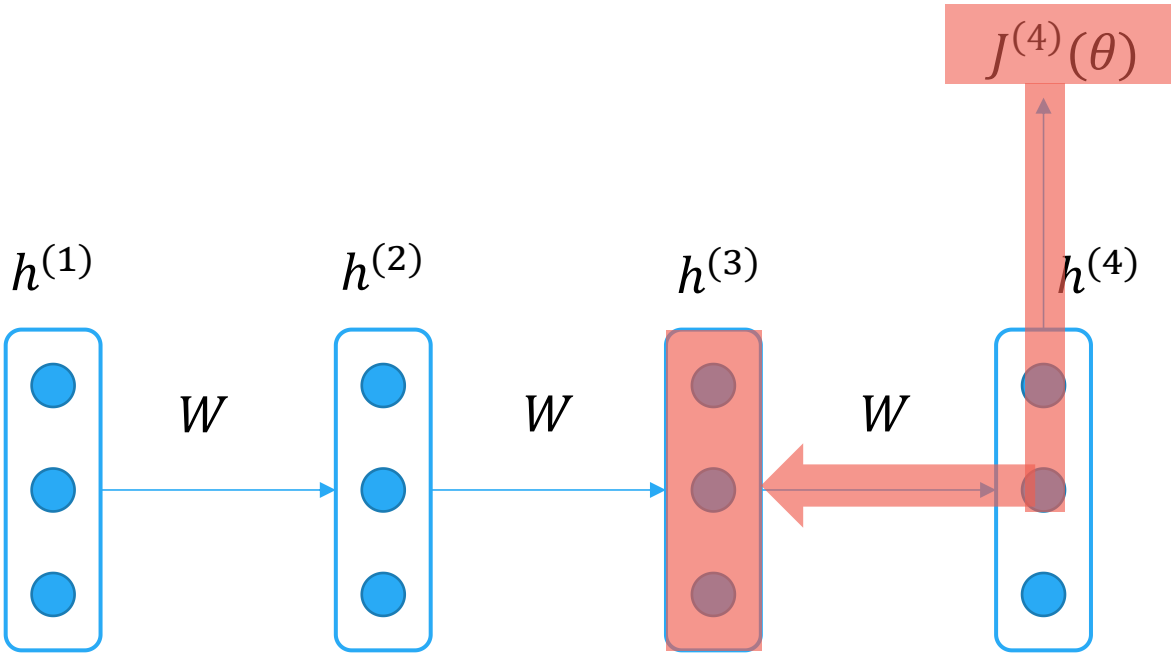


$$\frac{\partial J^{(4)}}{\partial h^{(1)}} = \frac{\partial h^{(2)}}{\partial h^{(1)}} \times \frac{\partial J^{(4)}}{\partial h^{(2)}}$$

链式法则

梯度消失

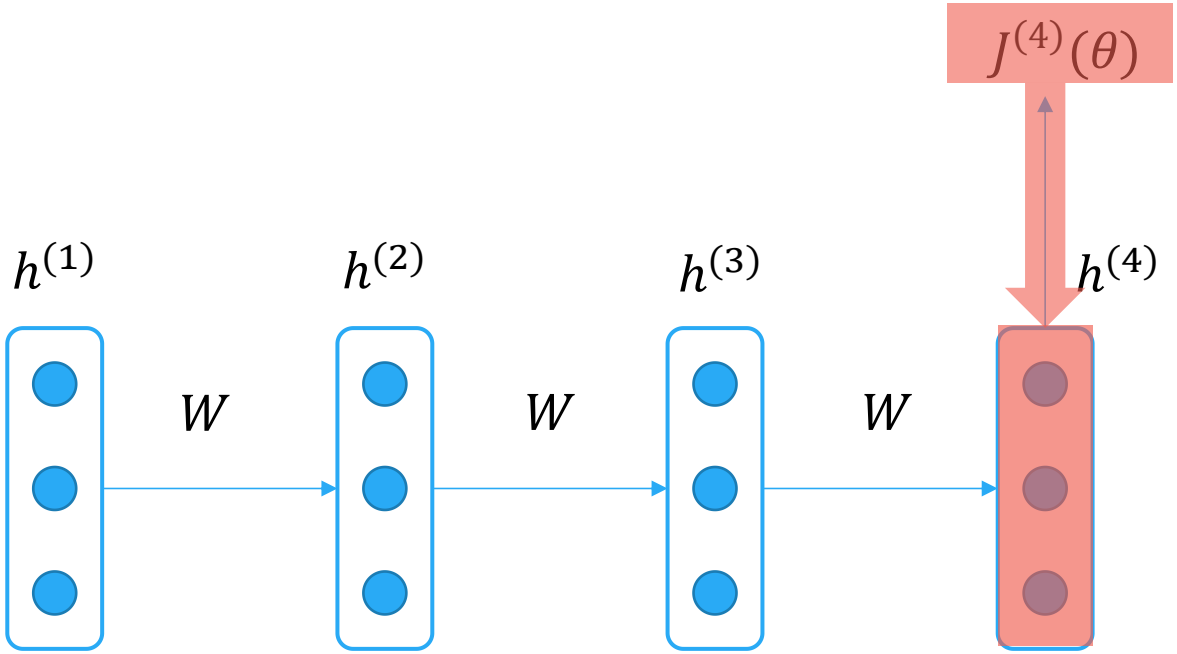
示例



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

梯度消失

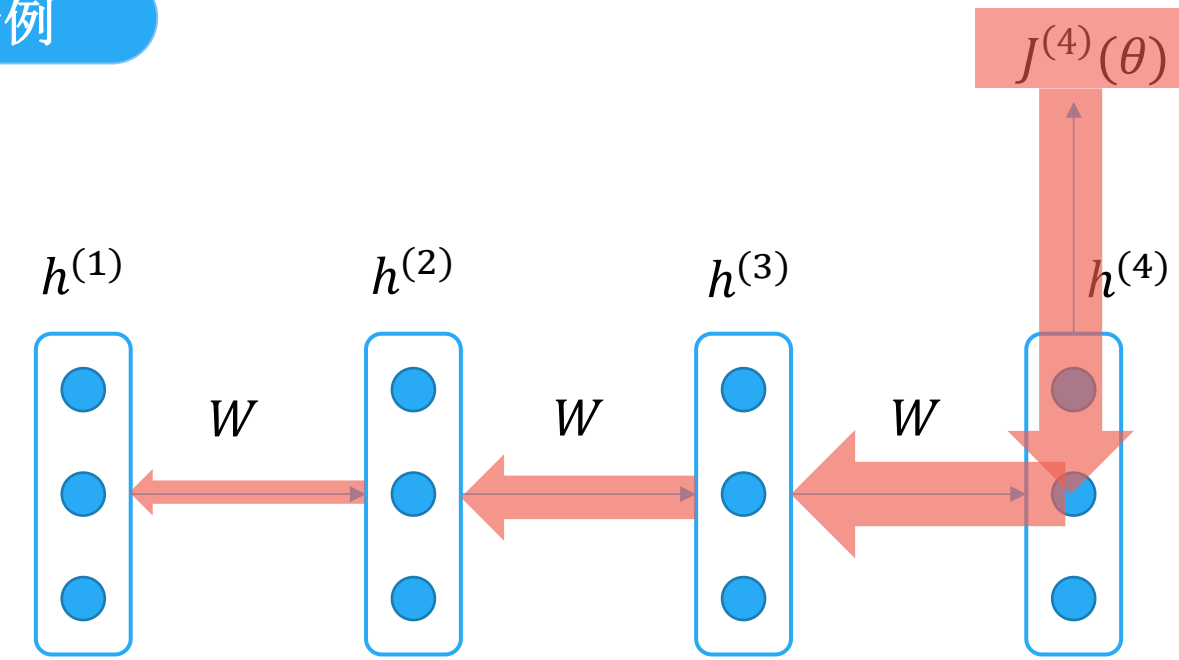
示例



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

梯度消失

示例



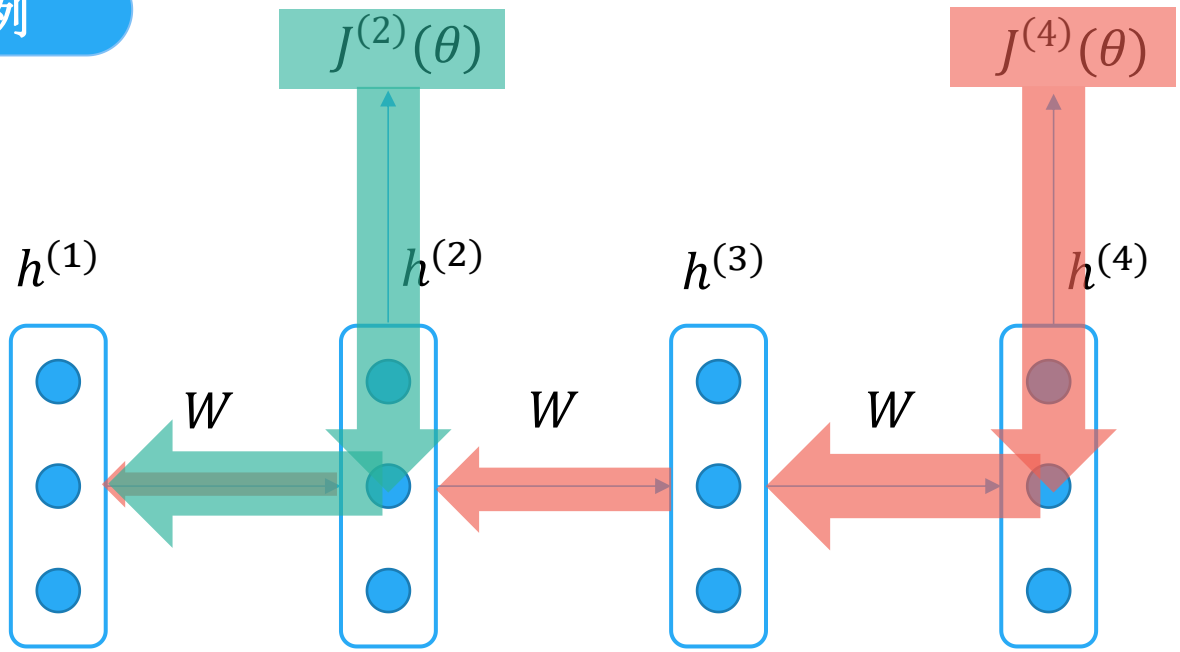
梯度消失：
当梯度回传路径越来越长，梯度变得越来越小

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

这几项数值都比较小

梯度消失

示例



由于距离比较近的梯度信号数值比较大，距离比较远的梯度信号被忽略

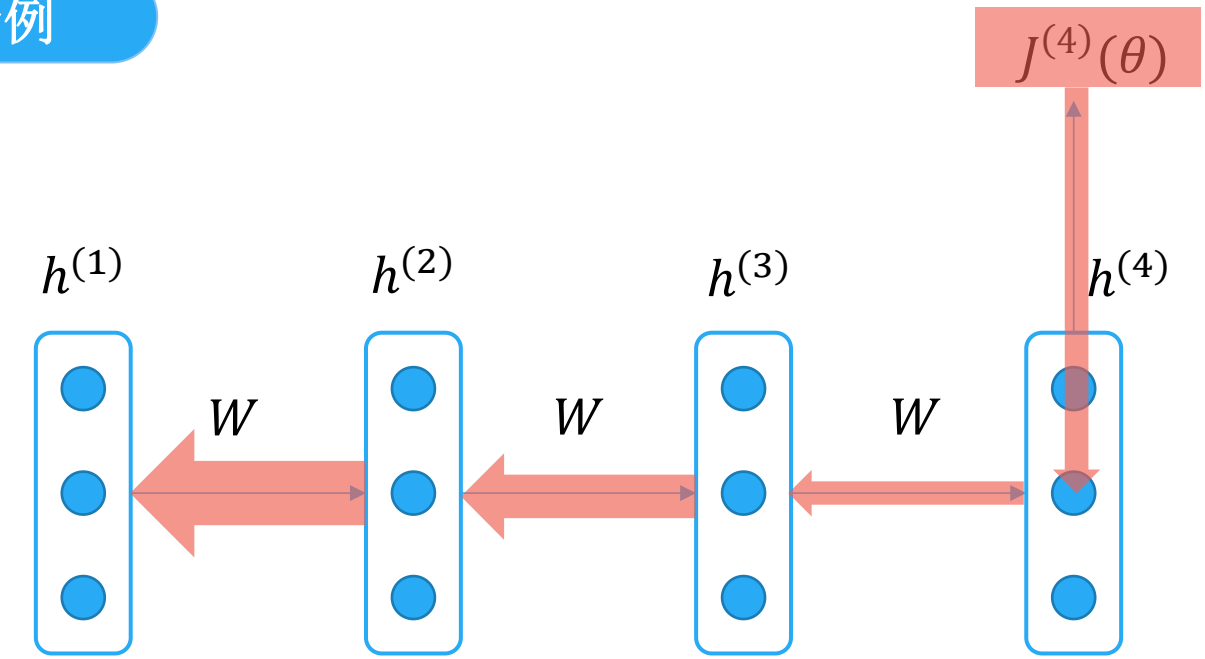
模型参数的更新可能基本只受近距离依赖关系影响，而不考虑长距离依赖

梯度消失

- 语言模型任务：The writer of the books _(is/are)
- 正确答案：The writer of the books is here
- 句法近因（syntactic recency）：The writer of the books is
- 顺序近因（sequential recency）：The writer of the books are
- 因为梯度消失问题的存在，RNN语言模型更容易学习到顺序近因，导致上述错误的出现。[Linzen et al, 2016]

梯度爆炸

示例



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

这几项数值都比较大

梯度爆炸

问题

- 梯度爆炸（gradient exploding）会导致更新过大，可能会导致参数到达一个非常差的区域（损失函数非常大）

梯度爆炸

梯度截断

- 梯度截断（Gradient clipping）是梯度爆炸的一种解决方法
  - 如果计算得到的梯度的模长大于设定的阈值时，将梯度按比例收缩，使梯度模长小于阈值，然后进行梯度更新
  - 直观解释：收缩后的梯度与原梯度方向一致，区别是步长更小

长短期记忆(Long Short-Term Memory)

- 长短期记忆(LSTM)是一种Hochreiter和Schmidhuber在1997年提出的用于解决梯度消失的循环神经网络
- 在第t步，网络有一个隐状态（hidden state）和一个单元状态（cell state）
  - 两个状态都是长度为n的向量
  - 单元（cell）中储存长期信息
  - LSTM可以对单元中的信息进行消除、读取、写入操作
- 信息的消/读/写用三个对应的门控(gate)来控制
  - 门控也是长度为n的向量
  - 在每一步中，门控的元素可以是打开（1），关闭（0）或者处于两者之间
  - 门控是动态改变的：它们的值是根据当前上下文计算得到的

长短期记忆网络

- 我们有一个序列输入x^t，然后我们计算隐状态h^t和单元状态c^t的序列
- 第t步的计算公式：

Sigmoid函数：所有门控中的值都在0到1之间

遗忘门控：控制上一个单元状态中信息的保留和遗忘

$$f^t = \sigma(W_f h^{t-1} + U_f x^t + b_f)$$

输入门控：控制新计算的单元内容哪一部分被写入单元状态

$$i^t = \sigma(W_i h^{t-1} + U_i x^t + b_i)$$

输出门控：控制单元状态哪一部分被输出到隐状态中

$$o^t = \sigma(W_o h^{t-1} + U_o x^t + b_o)$$

新单元内容：新计算得到的单元状态

$$\tilde{c}^t = \tanh(W_c h^{t-1} + U_c x^t + b_c)$$

单元状态：遗忘上一个单元状态的一部分内容，输入一部分新单元内容

$$c^t = f^t \cdot c^{t-1} + i^t \cdot \tilde{c}^t$$

隐状态：从单元状态中输出一部分内容

$$h^t = o^t \cdot \tanh c^t$$

均为长度为n的向量

门控使用按元素相乘

长短期记忆网络

缓解梯度消失

- LSTM的结构使得循环神经网络能够更容易保存多步之前的信息
  - 例如，如果遗忘门控被设置为每一步都记忆所有内容，则所有在单元中的信息都会被一直保存
  - 普通的循环神经网络很难学习到一个可以保存隐状态中所有信息的权重矩阵W_h

- LSTM不保证一定不会出现梯度爆炸或者梯度消失，但是它能让模型更容易学习长距离的依赖关系

门控循环单元（Gated Recurrent Units）

- 门控循环单元（GRU）是（Cho et al., 2014)提出的一种LSTM的简化替代
- 第t步,我们有输入x^t和隐状态h^t(没有单元状态)

更新门控：控制隐状态内容的更新和保留

$$u^t = \sigma(W_u h^{t-1} + U_u x^t + b_u)$$

重置门控：控制上一个隐状态中的哪一部分内容被用来计算新的内容

$$r^t = \sigma(W_r h^{t-1} + U_r x^t + b_r)$$

新的隐状态内容：重置门控选择上一个隐状态中有用的部分，使用这部分和当前的输入计算新的隐状态内容

$$\tilde{h}^t = \tanh(W_h(r^t \cdot h^{t-1}) + U_h x^t + b_h)$$

隐状态：更新门控同时控制隐状态内容的保留和更新

$$h^t = (1 - u^t) \cdot h^{t-1} + u^t \cdot \tilde{h}^t$$

6.4 注意力机制

- 循环神经网络的一个主要局限是不能很好地建模长距离依赖
- 为了更好地建模长距离依赖，我们引入注意力机制（attention mechanism）
- 注意力机制根据当前状态计算查询（query），根据每一个历史隐状态计算键（key），进而计算查询与键的匹配程度，即注意力分数（attention score）
- 对注意力分数归一化得到注意力分布，将所有历史隐状态的值（value）向量加权平均

$$\text{Attention}(\boldsymbol{q}, \boldsymbol{K}, \boldsymbol{V}) = \sum_i \frac{\exp(\boldsymbol{q}^{\mathrm{T}}\boldsymbol{k}_i)}{\sum_j \exp(\boldsymbol{q}^{\mathrm{T}}\boldsymbol{k}_j)}\boldsymbol{v}_i$$

6.4 注意力机制

注意力掩码

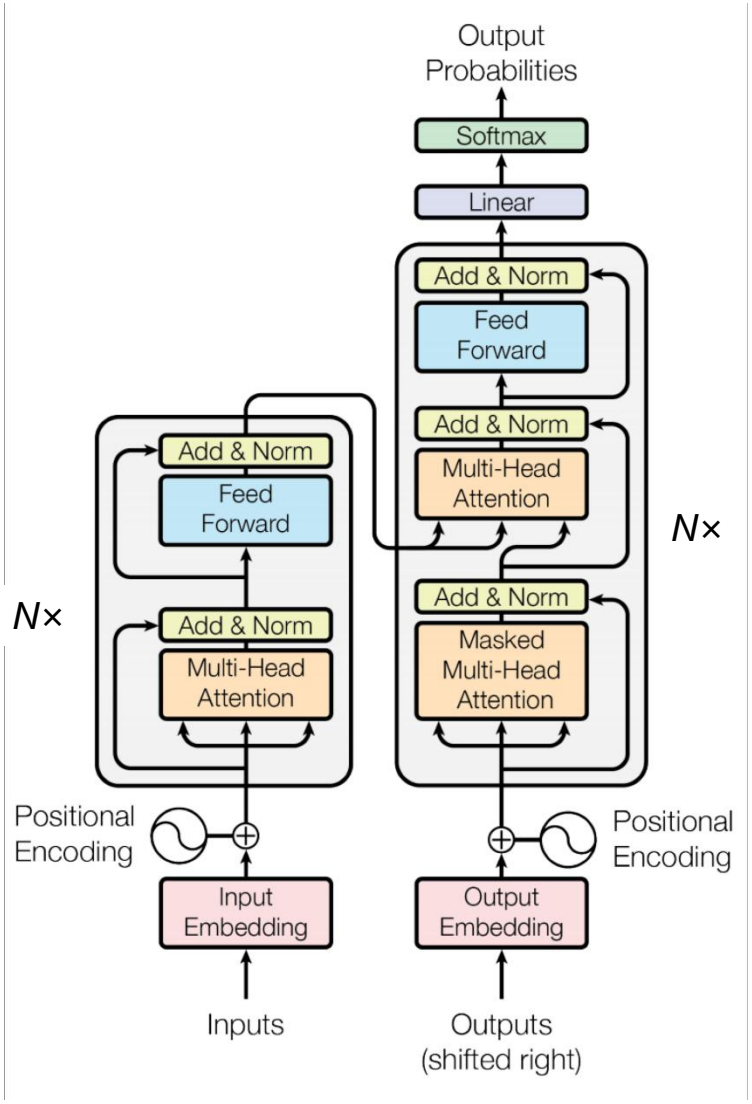
- 一个查询会对整个序列所有位置计算注意力，但是对于语言模型，每一步的查询应当只能看到该步及之前的输入
- 因此需要引入注意力掩码（attention mask），将每一步的查询对该步之后位置的注意力分数置为负无穷

缩放点乘注意力

- 随着查询和键维度的增大，不同的键所计算的内积的数值范围也会逐渐增大，由此会带来数值稳定性问题，可以采用缩放点乘注意力（scaled dot-product attention）

6.5 Transformer模型

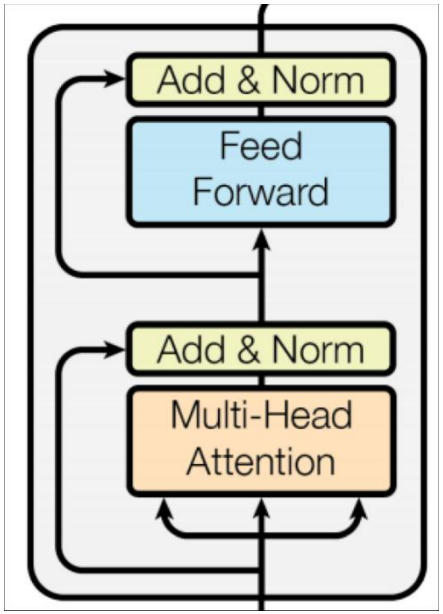
- Transformer模型 [Vaswani et al., 2017]
- 非循环（non-recurrent）的序列到序列编码器-解码器模型
- 任务：机器翻译
  - 编码器处理输入句子
  - 解码器依次预测输出句子的词



"Attention Is All You Need", Vaswani et al, 2017

Transformer块

- 每一个块包含两个子层
  - 多头注意力（Multi-head attention）
  - 2层前向神经网络（使用ReLU）



- 每一个子层还包含残差连接（Residual connection）和层归一化（Layer Norm）

$$\text{LayerNorm}(x + \text{Sublayer}(x))$$

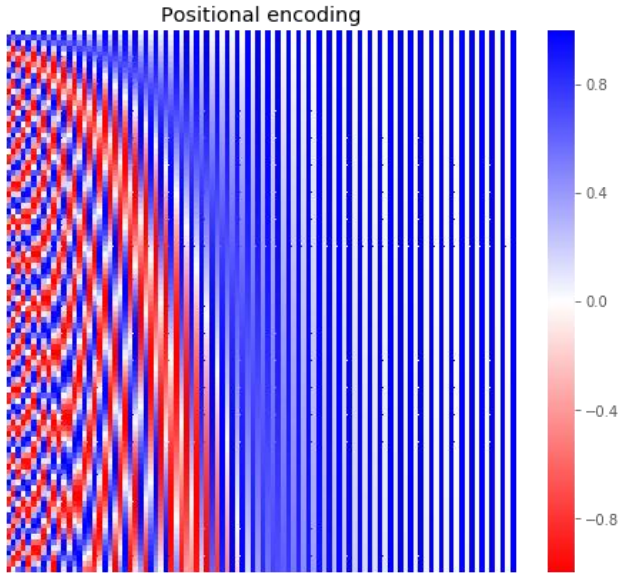
- LayerNorm标准化输入的均值和方差
  - Layer Normalization

位置编码

- 前述编码器忽略了单词的位置信息
- 为了表示位置信息，在输入词向量上加上了位置编码（Positional Encoding），使得相同词在不同位置的表示不相同

$$PE_{(pos,2i)} = \sin(pos/10000^{2i/d_{\text{model}}})$$

$$PE_{(pos,2i+1)} = \cos(pos/10000^{2i/d_{\text{model}}})$$



sinusoidal positional encoding

评价

困惑度（Perplexity）

- 直观上，语言模型对于（未见过的）真实的句子应当给出更高的概率
- 对于测试集中的所有句子$x_1, x_2, \cdots, x_m$
  - 概率为

$$\Pi_{i=1}^{m} P(x_i)$$

  - 概率对数为

$$\Sigma_{i=1}^{m} \log_2 P(x_i)$$

  - 平均每个单词的概率对数为

$$l = \frac{1}{M} \Sigma_{i=1}^{m} \log_2 P(x_i)$$

M为测试集中单词总数

- 困惑度 $= 2^{-l}$

评价

困惑度

困惑度越小，语言模型越好

- 如果语言模型对于测试集每个句子的概率都估算为1
  - 困惑度=1
- 如果语言模型对于每个单词的概率都估算为$1/V$
  - 困惑度=V
- 如果语言模型对于某个句子的概率估算为0
  - 困惑度=正无穷

评价

困惑度

- 注意：使用不同词表的语言模型，其困惑度不可比！
  - 极端例子：如果一个语言模型词表为空，把所有词汇都当做[UNK]，那么该语言模型的困惑度一定会等于1（困惑度的最小值）

总结

- 语言模型
  - *n*元语法模型
  - 循环神经网络
    - 梯度消失与梯度爆炸
    - 长短期记忆与门控循环单元
  - 注意力机制
  - Transformer模型
  - 评价指标