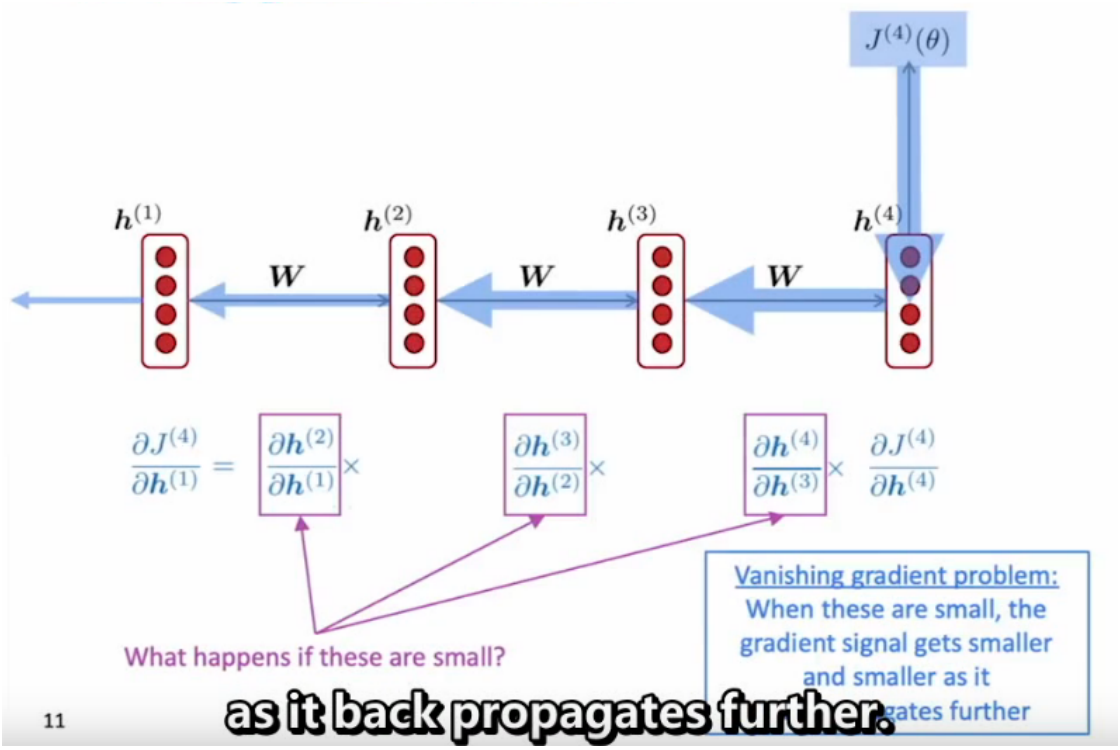
Lecture07 Vanishing Gradients and Fancy RNNs

创建时间： 2019/10/27 20:18 **更新时间：** 2020/1/13 21:00
作者： wjj4work@163.com

vanishing gradient



随着反向传播，梯度会越来越小、

Vanishing gradient proof sketch

- Recall: $h^{(t)} = \sigma\left(W_h h^{(t-1)} + W_x x^{(t)} + b_1\right)$
- Therefore: $\frac{\partial h^{(t)}}{\partial h^{(t-1)}} = \text{diag}\left(\sigma'\left(W_h h^{(t-1)} + W_x x^{(t)} + b_1\right)\right) W_h$ (chain rule)
- Consider the gradient of the loss $J^{(i)}(\theta)$ on step *i*, with respect to the hidden state $h^{(j)}$ on some previous step *j*.

$$\frac{\partial J^{(i)}(\theta)}{\partial h^{(j)}} = \frac{\partial J^{(i)}(\theta)}{\partial h^{(i)}} \prod_{j<t\leq i} \frac{\partial h^{(t)}}{\partial h^{(t-1)}} \quad \text{(chain rule)}$$

$$= \frac{\partial J^{(i)}(\theta)}{\partial h^{(i)}} W_h^{(i-j)} \prod_{j<t\leq i} \text{diag}\left(\sigma'\left(W_h h^{(t-1)} + W_x x^{(t)} + b_1\right)\right) \quad \text{(value of } \frac{\partial h^{(t)}}{\partial h^{(t-1)}}\text{)}$$

如果权重矩阵很小，梯度就会更小
考虑L2范式：

$$\left\|\frac{\partial J^{(i)}(\theta)}{\partial h^{(j)}}\right\| \leq \left\|\frac{\partial J^{(i)}(\theta)}{\partial h^{(i)}}\right\| \|W_h\|^{(i-j)} \prod_{j<t\leq i} \left\|\text{diag}\left(\sigma'\left(W_h h^{(t-1)} + W_x x^{(t)} + b_1\right)\right)\right\|$$

梯度会指数级变小
Pascanu等人证明了如果W的最大特征值是小于1，那么梯度会指数缩小；
有一个类似的证明：W最大的特征值>1，爆炸梯度

为什么梯度消失是个问题？

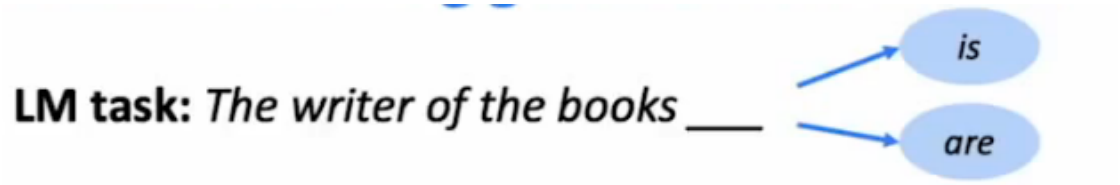
- 来自远处的梯度信号由于比来自近处的梯度信号小得多而丢失。所以模型的权值只会根据近期影响来更新，这将失去长期影响（long-term effects）
  而有些情况下会需要学习long-term influences
- 另一种解释是:梯度可以被看作是过去对未来影响的量度
  如果在更长的距离内，梯度变得越来越小(从第t步到第t+n步)，那么我们无法判断:
  1. 第t步和t+n步之间**没有相关性**
  2. 我们用了**错误的参数**来捕获t和t+n之间的真正依赖关系

例子1：

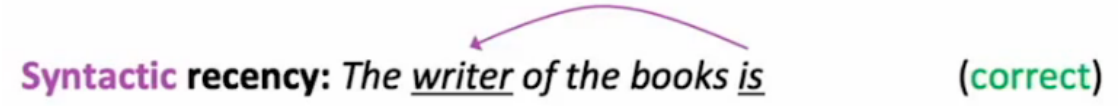
LM task: *When she tried to print her tickets, she found that the printer was out of toner. She went to the stationery store to buy more toner. It was very overpriced. After installing the toner into the printer, she finally printed her _____*

要学习这个训练示例，RNN-LM需要对第7步中的“tickets”和最后的目标单词“tickets”之间的依赖关系建模。但是如果梯度太小，模型无法学习到之间的依赖关系。因此，该模型无法预测类似的长距离依赖关系

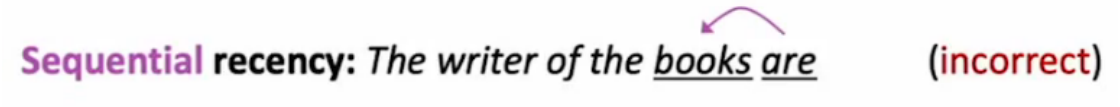
例子2：



语法近因：



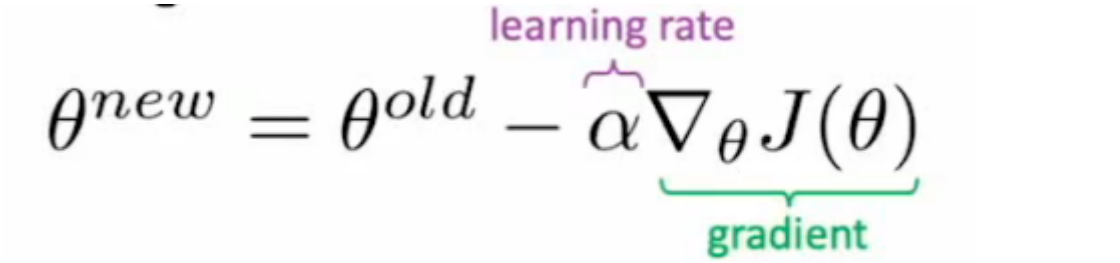
顺序近因：



由于梯度消失，RNN-LMs更善于从顺序最近性而不是句法最近性中学习，所以他们犯这种错误的频率比我们希望的要高[Linzen et al 2016]。

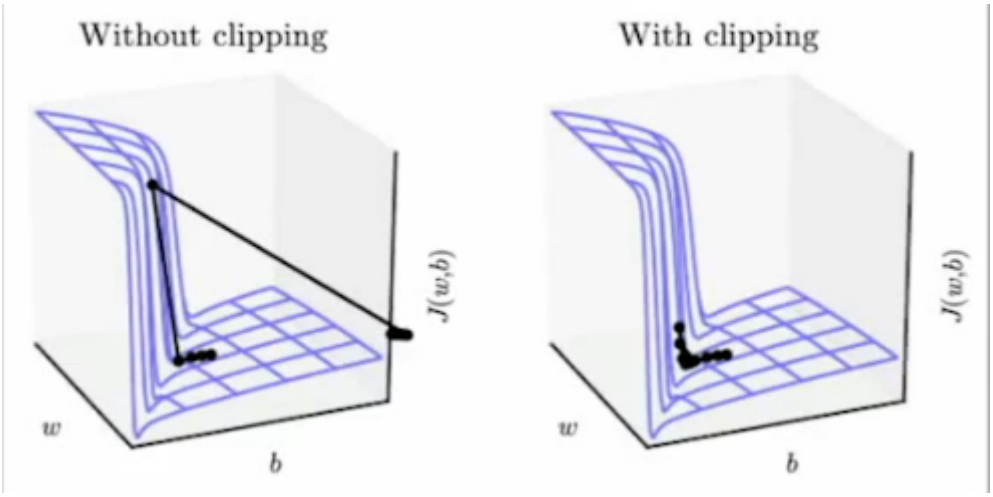
为什么梯度爆炸是个问题？

如果梯度过大，则SGD更新步骤过大，模型参数也会大幅度的改变，得到坏的参数



在最坏的情况下，这将导致网络的Inf或NaN(然后你必须从一个较早的检查点重新开始训练)

解决办法： Gradient clipping梯度裁剪
如果梯度的范数大于某个阈值，则在应用SGD更新之前将其缩小（选择更小的步长）



- 左图，由于没有进行clipping，步进长度过大，导致损失函数更新到一个不理想的区域

- 右图进行clipping后每次步进减小，能更有效的达到极值点区域。

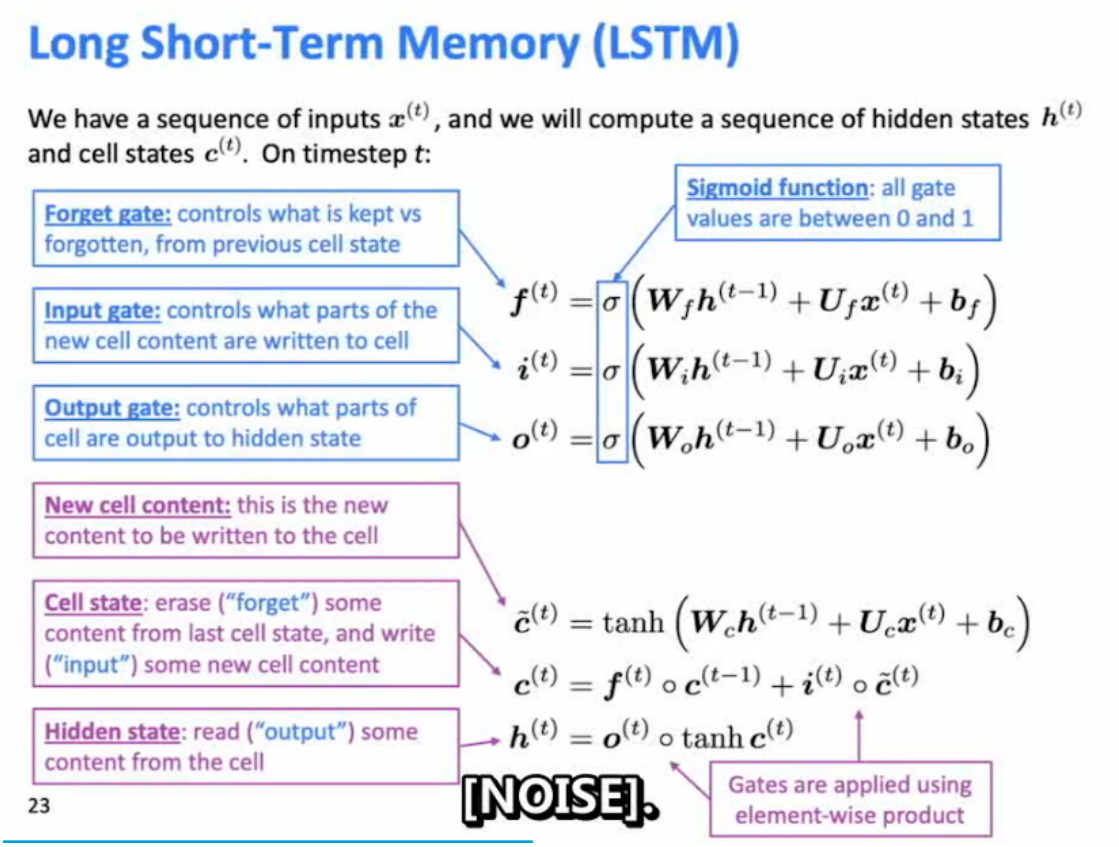
怎么解决梯度消失的问题

RNN很难学习在多个时间步长的情况下保存信息，隐藏状态不断被重写

Long Short-Term Memory(LSTM)

在第 t 步，有一个隐藏状态 h(t) 和一个单元状态 c(t)

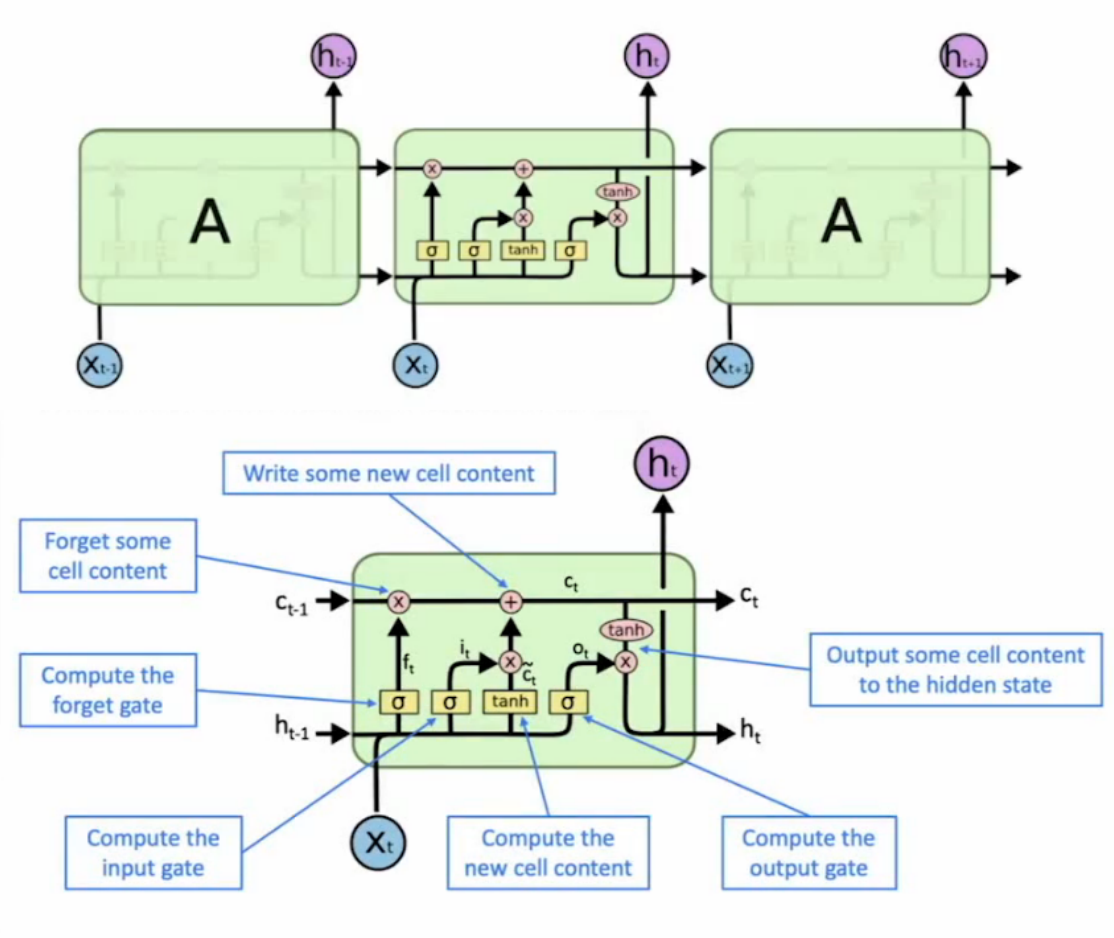
- 都是长度为 n 的向量
- 单元存储长期信息
- LSTM可以从单元中删除、写入和读取信息，由三个对应的门控制
  - 门也是长度为n的向量
  - 在每个时间步上，门的每个元素都可以open(1)、closed(0)或介于两者之间。
  - 门是动态的，值基于当前上下文计算



遗忘门：控制上一个单元状态的保存与遗忘
输入门：控制写入单元格的新单元内容的哪些部分
输出门：控制单元的哪些内容输出到隐藏状态

新单元内容：这是要写入单元的新内容
单元状态：删除(“忘记”)上次单元状态中的一些内容，并写入(“输入”)一些新的单元内容
隐藏状态：从单元中读取(“output”)一些内容
Sigmoid函数：所有的门的值都在0到1之间

通过逐元素的乘积来应用门
这些是长度相同的向量



LSTM怎样解决梯度消失问题的？

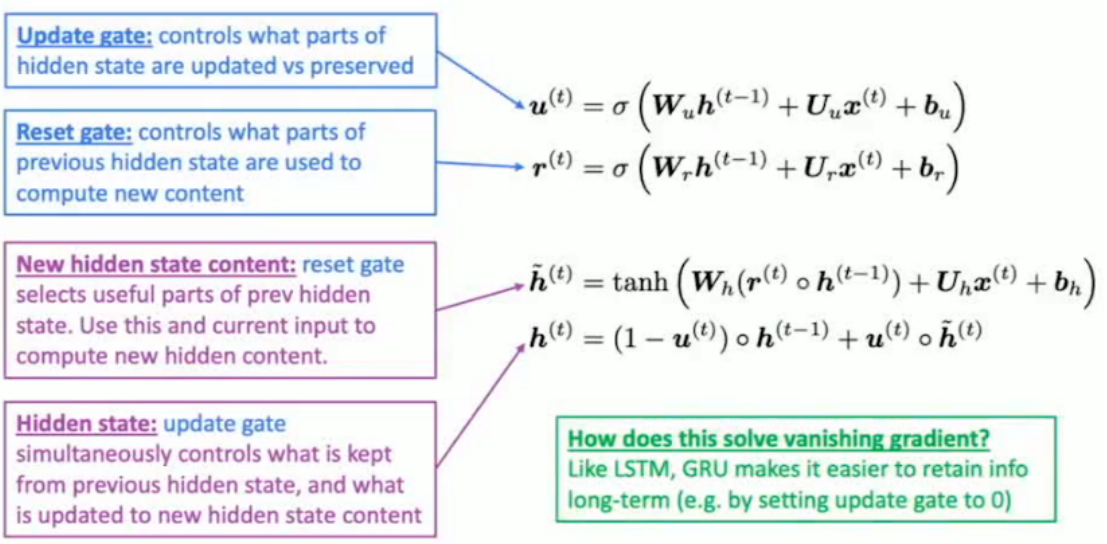
- LSTM体系结构使RNN更容易在多个时间步长的情况下保存信息
  - 例如，如果“遗忘门”设置为在每一步都记住所有内容，那么单元格中的信息将被无限保存。相比之下，普通的RNN很难学习一个保持信息隐藏状态的递归权矩阵Wh。
- LSTM并不保证没有梯度消失/爆炸，但是它确实为模型提供了一种更容易的方法来学习长距离的依赖关系。

LSTM的成功

- 2013-2015年，LSTM开始实现最先进的结果成功的任务 - 手写识别、语音识别、机器翻译、解析、图像字..... LSTM成为主导方法
- 现在(2019年)，其他方法(如Transformers)在某些任务上变得更加主导。
  - 在2016年WMT中，总结报告包含“RNN”44次
  - 在2018年WMT中，总结报告包含“RNN”9次，“Transformers” 63次

Gated Recurrent Units (GRU)

在每个时间步 t 上，我们都有输入 x(t)和隐藏状态 h(t)(没有单元状态)



LSTM vs GRU

- GRU计算速度更快，参数最少
- 没有明确证据证明那个效果最好
- LSTM是一个很好的默认选择（尤其数据包含很长的依赖关系过有很多训练数据时）
- 经验法则：先从LSTM开始，更有效率就用GRU

总结

梯度消失/爆炸是很多神经结构（前馈、卷积），尤其是深度结构的一个问题：较底层的训练会很难

解决方案：大量新的深层前馈 / 卷积架构，添加更多的直接连接(从而使梯度可以流动)
例：

- 残差连接 ResNet

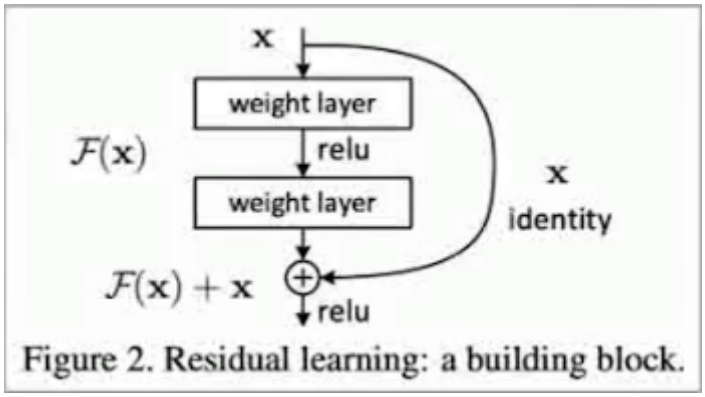
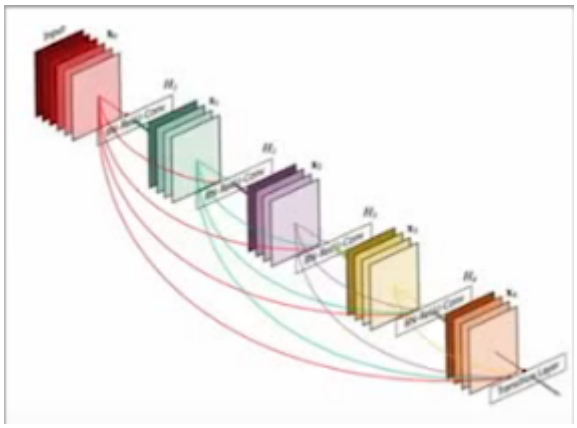


Figure 2. Residual learning: a building block.

标识连接默认保存信息

- 密集连接 DenseNet



直接连接所有内容

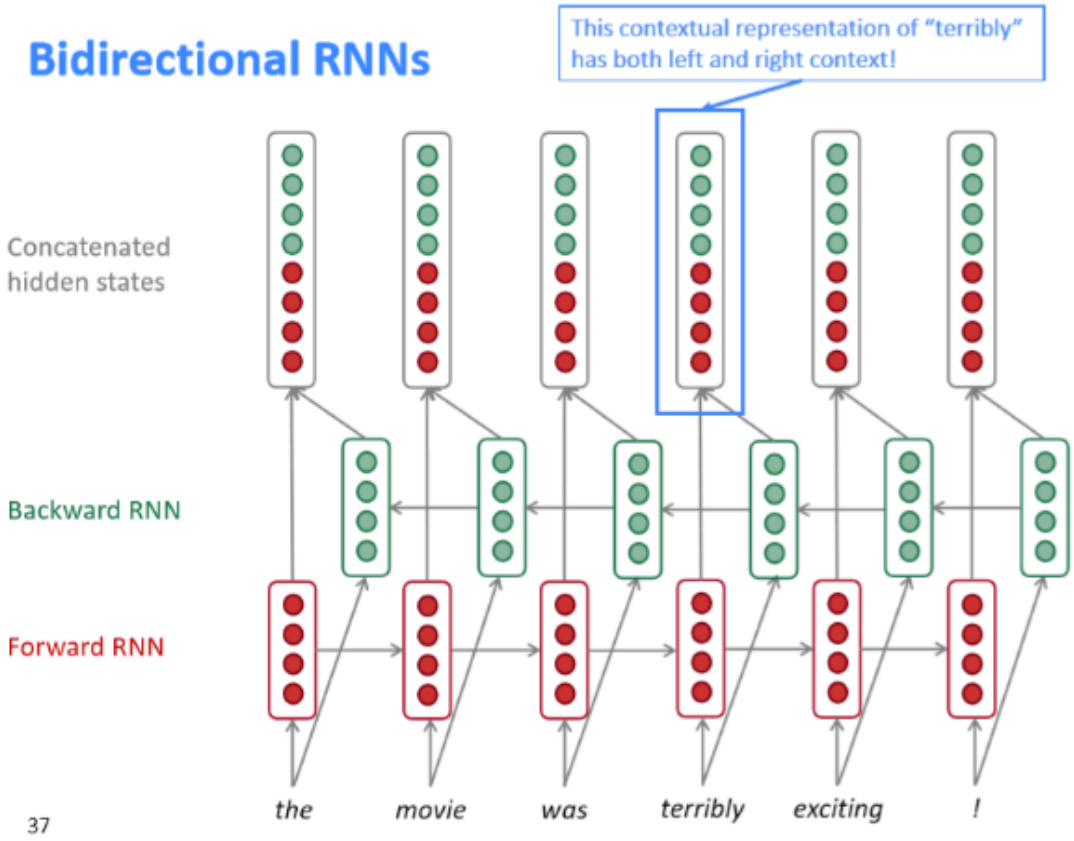
- Highway connections
  与残差连接一样，但标识链接与转换层由动态门控制
  应用与深度前馈/卷积网络

由于反复乘以一个相同的权重矩阵，RNN是不稳定的

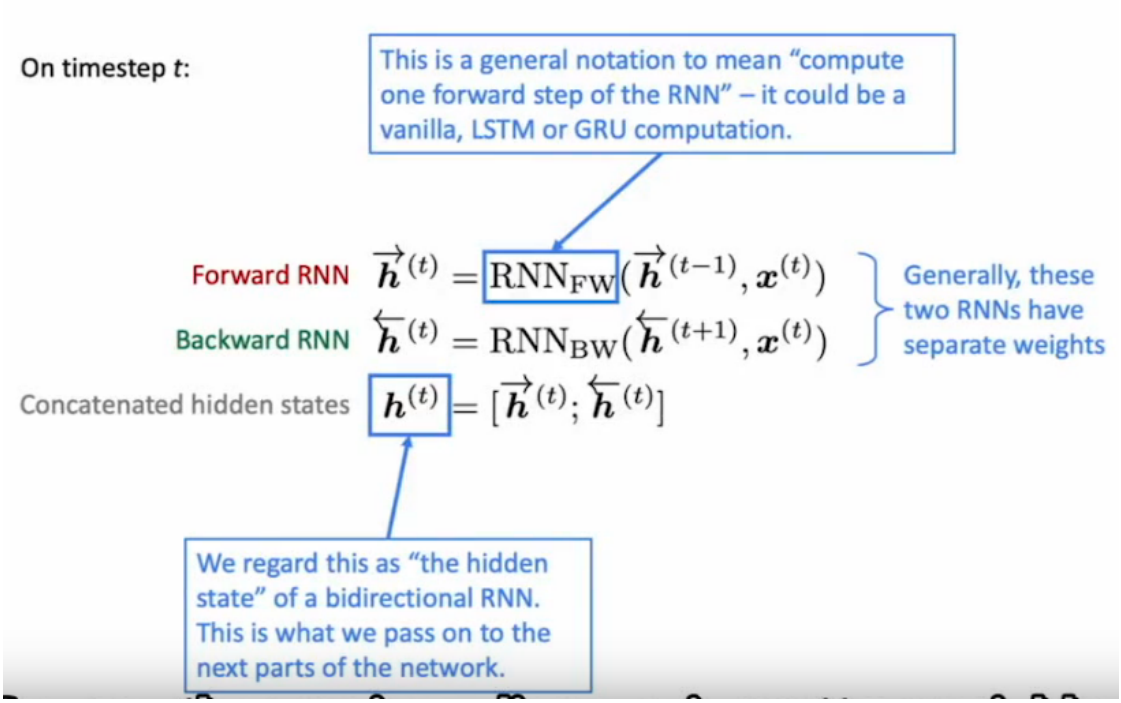
Bidirectional RNNs

用于情感分类

我们可以把这种隐藏状态看作是这个句子中单词“terribly”的一种表示。我们称之为上下文表示。这些上下文表示只包含左上下文的信息（the movie was），而右侧上下文（exciting ！）使terribly的意思由否定变肯定。



这样得到的隐藏状态同时包含左右上下文信息



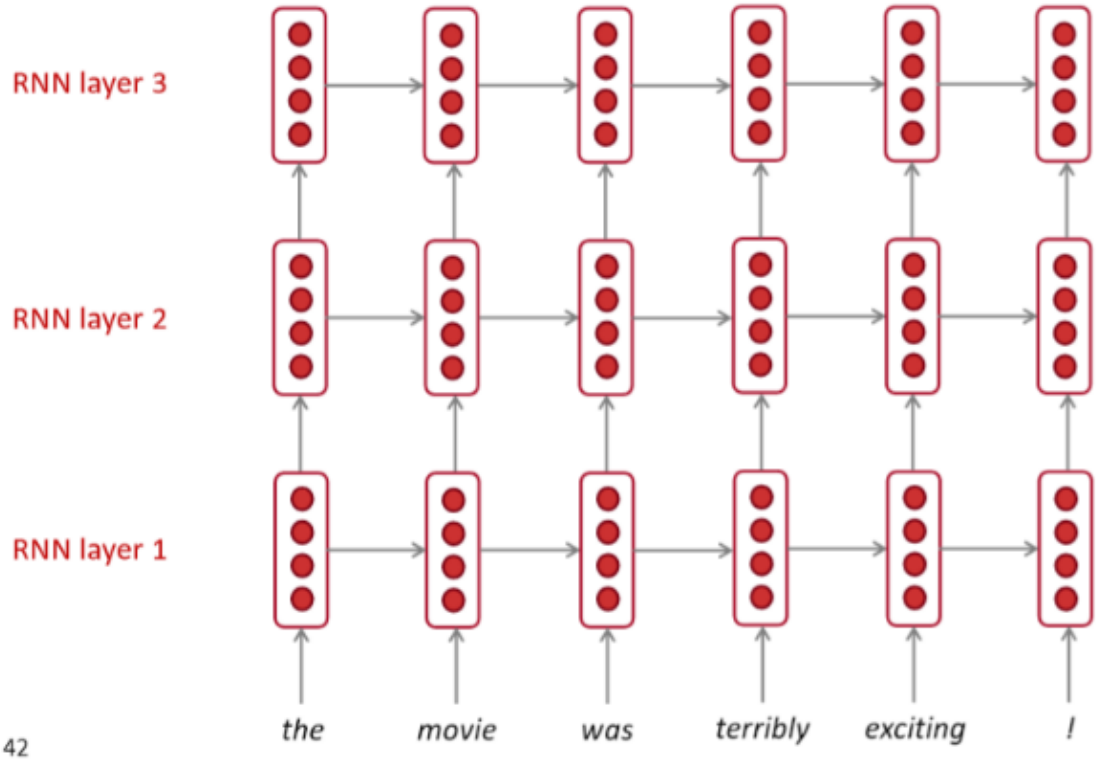
一般来说，这两个RNNs有各自的权重。当有大量训练数据时，可能相同的权重的效果更好。

注意：

- 双向RNN只适用于整个输入序列，不适用于语言模型，因为语言模型只有左侧上下文；
- BERT(来自transformer的双向编码器表示)是一个基于双向性的强大的预训练的上下文表示系统

Multi-layer RNNs (stacked RNNs)

允许网络计算更加复杂的表示：低层RNNs计算低级别特征，高层RNNs计算高级别特征（语义）



课堂问题：以什么样的顺序计算？
答：比较灵活，可以逐词一层一层计算。也可以逐层计算；
但如果变为双向RNN，就没有灵活性，只能逐层计算

- 多层RNN性能好（但没有卷积或前馈网络深）
- 在2017年的一篇论文，Britz et al 发现在神经机器翻译中，RNN编码器最好是2到4层，RNN解码器4层
- 但是，训练更深RNN需要加入skip-connections / dense-connections
- RNN无法并行化，计算代价过大，所以不会过深
- Transformer-based 的网络(如BERT)可以多达24层。他们有很多skipping-like的连接