lecture06 The probability of a sentence Recurrent Neural Networks and Language Models

创建时间：	2019/10/8 15:08	**更新时间：**	2020/1/13 20:59
作者：	hisaishi@sina.com		

Language Modeling

任务是预测下一个单词是什么
更为正式的解释见下图：

More formally: given a sequence of words $x^{(1)}, x^{(2)}, \ldots, x^{(t)}$, compute the probability distribution of the next word $x^{(t+1)}$:

$$P(x^{(t+1)} | x^{(t)}, \ldots, x^{(1)})$$

where $x^{(t+1)}$ can be any word in the vocabulary $V = \{w_1, \ldots, w_{|V|}\}$

即给定一个单词序列，计算下一个单词的概率分布
同样也可以理解为语言模型是一个将概率分配给一段文本的系统

n-gram Language Models

定义:n个字母是由n个连续的单词组成的块。
e.g. the students opened their_____
unigrams："the"，"students"， "opened"， "their'
bigrams： “the students”、 “students opened”、 “opened their”
trigrams： “students opened their”
4-grams: “the students opened their"

思路：收集不同的n-grams频率的统计数据，并利用这些数据来预测下一个字。

一、首先，我们做一个 简化假设 ：x(t+1) 只依赖于前面的n-1个单词

$$P(x^{(t+1)}|x^{(t)}, \ldots, x^{(1)}) = P(x^{(t+1)}|\overbrace{x^{(t)}, \ldots, x^{(t-n+2)}}^{n-1 \text{ words}}) \quad \text{(assumption)}$$

prob of a n-gram → $P(x^{(t+1)}, x^{(t)}, \ldots, x^{(t-n+2)})$
$=$
prob of a (n-1)-gram → $P(x^{(t)}, \ldots, x^{(t-n+2)})$

(definition of conditional prob)

Q：但是怎么得到n-gram和(n-1)-gram的概率？
A：通过在一些大型文本语料库中计算它们

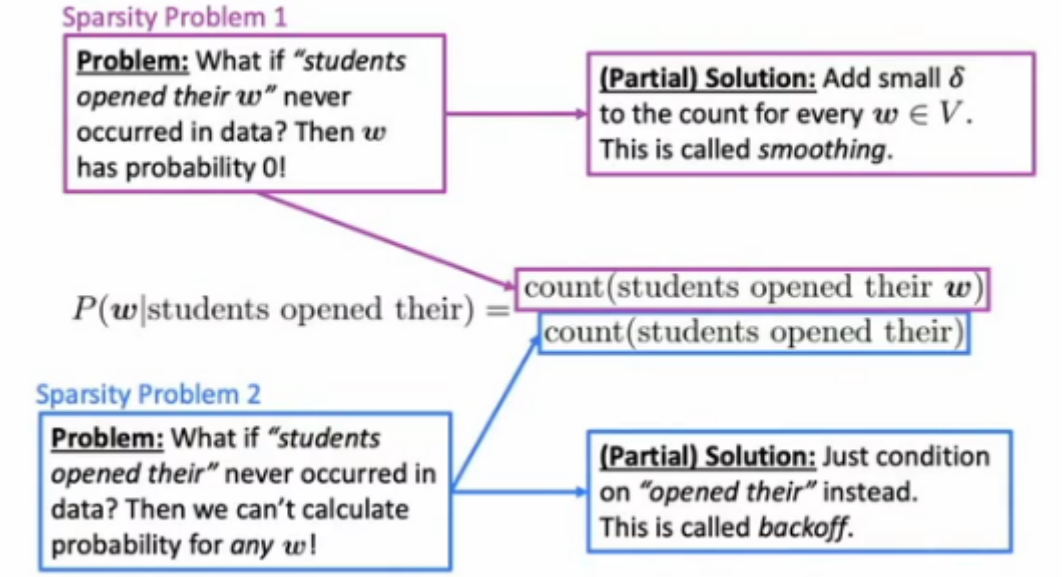
二、假设我们在学习一个4-gram的语言模型
e.g. as the proctor started the clock， the students opened their_____
下一个单词仅依赖于他前面的三个单词，即“students opened their”

$$P(\boldsymbol{w}|\text{students opened their}) = \frac{\text{count}(\text{students opened their } \boldsymbol{w})}{\text{count}(\text{students opened their})}$$

假设:

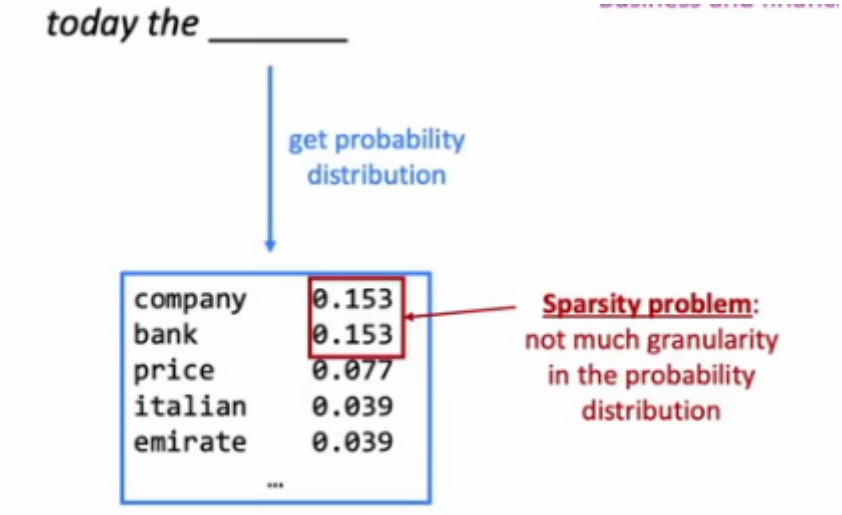
- "students opened their”出现了1000次
- “students opened their books”出现了400次
  P(books | students opened their) = 0.4
- “students opened their exams”出现了100次
  P(exams| students opened their) = 0.4

但是如果使用了太多的训练语料库，可能会使预测结果与上下文内容偏离
另一个问题是稀疏问题，可能在训练语料库中并没有这样的n-gram短语，分子为0
解决办法：增加一个很小的delta，使在计算中每个单词至少有一个很小的概率
还有一种稀疏问题，万一分母为0，即在训练预料中并没有出现这样的(n-1)-gram
解决办法：退回到(n-2)-gram,在例子中，即“opened their”



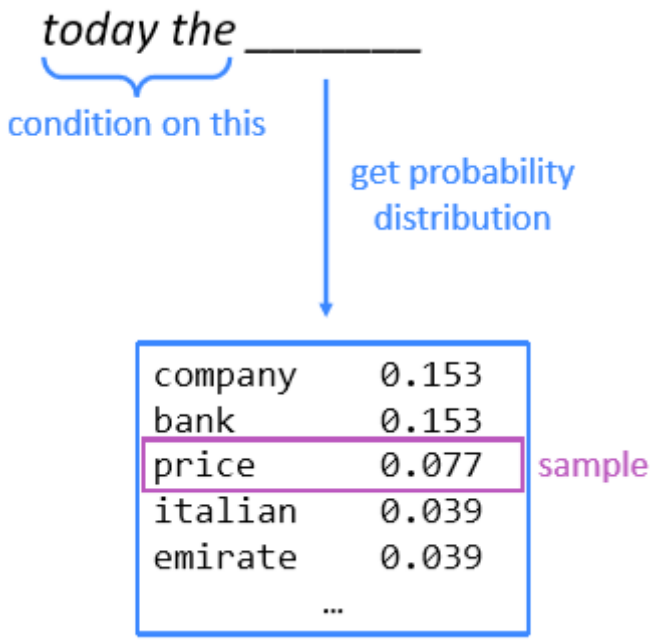
当n越大时，稀疏问题越严重。通常n不大于5。
存储问题：需要存储语料库中所有出现的n-grams的次数，增大n也会增加次数和增加模型大小

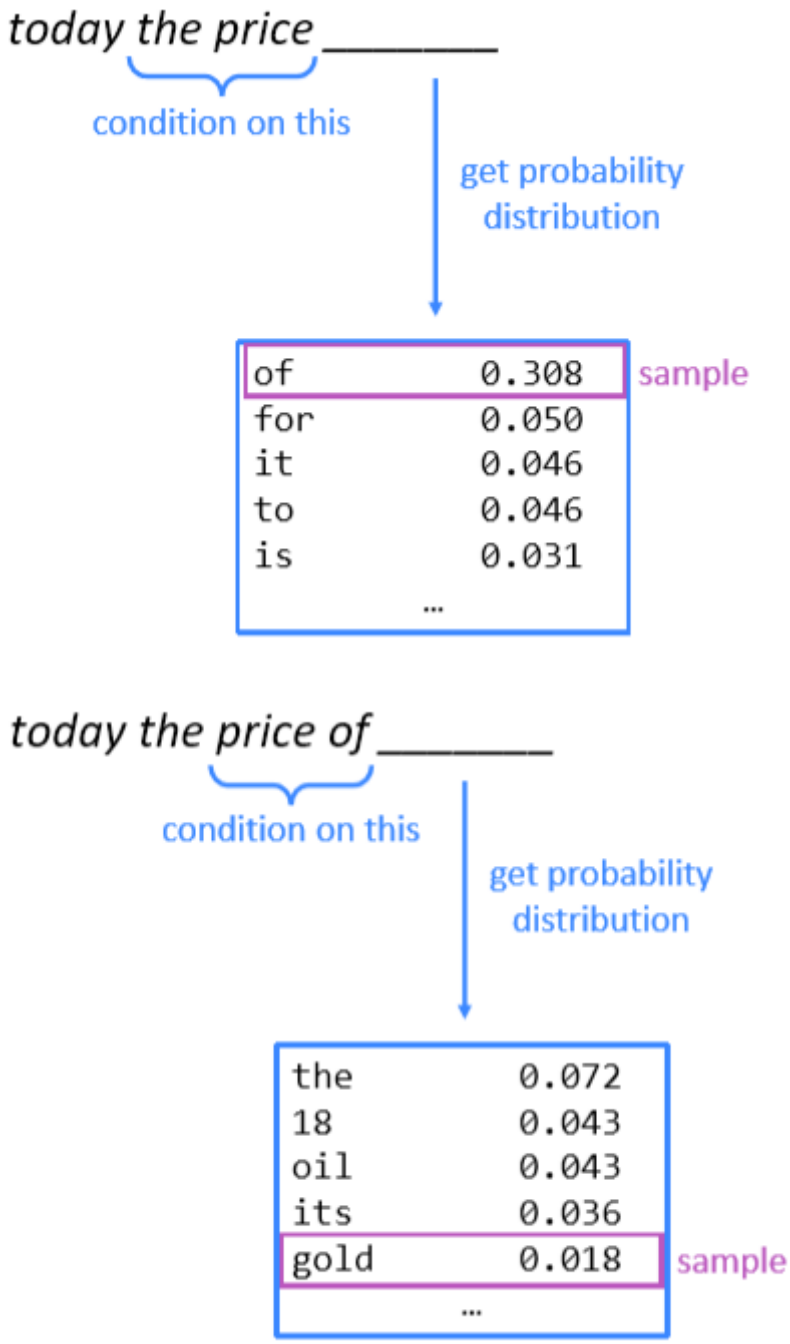
Example：



概率分配的粒度不大

同样也可以使用语言模型来生成文本：





会得到如下文本：

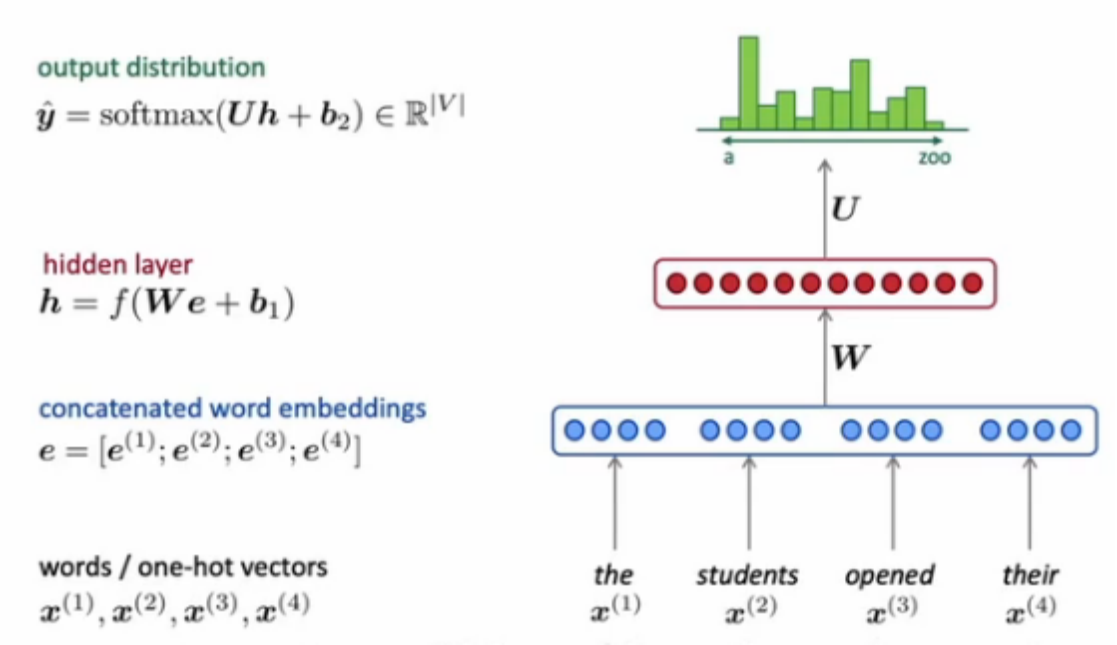
> today the price of gold per ton , while production of shoe lasts and shoe industry , the bank intervened just after it considered and rejected an imf demand to rebuild depleted european stocks , sept 30 end primary 76 cts a share .

符合语法但是不连贯，如果我们想要建立良好的语言模型，我们需要考虑的不仅仅是三个单词。但n的增加加剧了稀疏性问题，增加了模型的大小。

#课堂问题：语言模型怎么知道何时加入逗号？
回答：将逗号等其他符号看作是单词的一种

怎么构建一个神经语言模型？

丢弃除窗口大小的上下文，假设窗口大小为4 ，使用和NER一样的网络结构

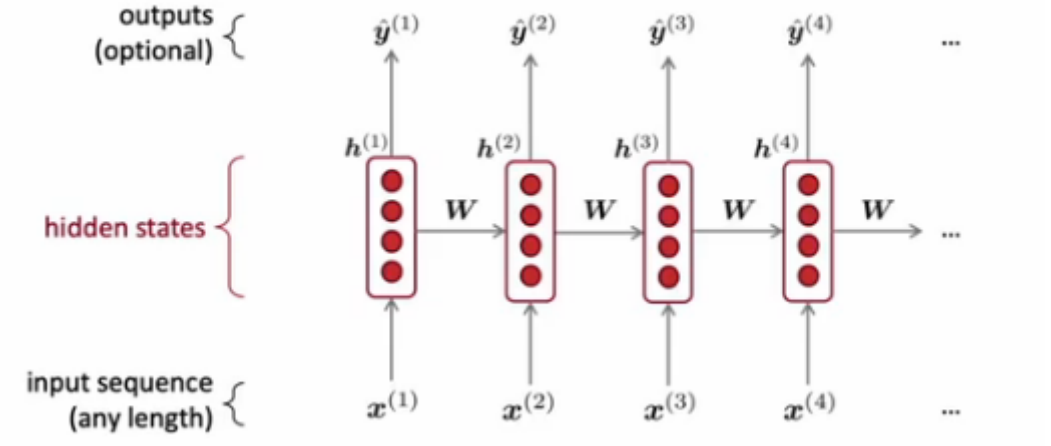


与n-gram相比的好处：

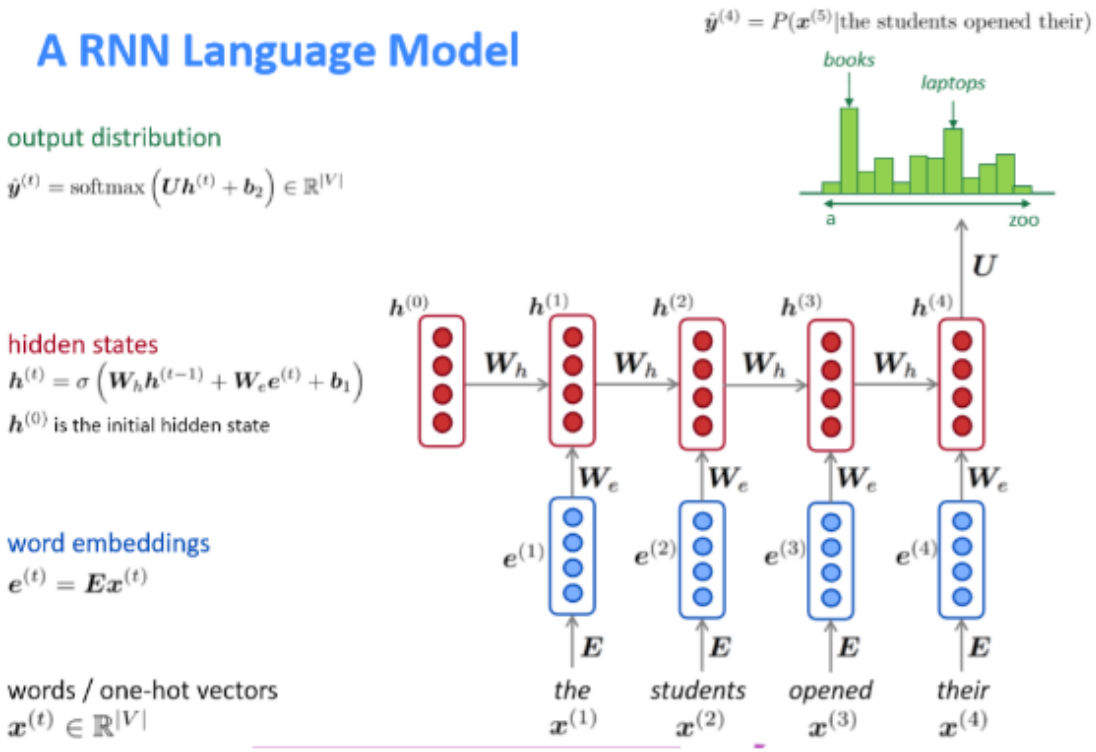
- 没有稀疏问题
- 不需要存储所有出现的n-grams
  仍存在的问题：
- 窗口大小很小，无论多大，都会丢失有用的上下文
- 如果增大窗口大小，同时也会增加权重的大小
- Window can never be large enough !
- x(1) 和 x(2) 乘以完全不同的权重。输入的处理 不对称

我们需要一个可以处理任何长度输入的神经结构！

递归神经网络RNN



- 输入的句子可以是任何长度；
- 有一个隐藏状态的序列，有多少输入就有多少个隐藏状态；
- 每个隐藏状态的计算都是基于上一个隐藏状态和当前步的输入
- ！使用相同的权重矩阵W ——使输入任何长度成为可能



将最后一个隐藏状态输入一个线性层，再经过softmax函数得到输出

优点：

- 能处理任何长度的输入
- 步骤t的计算(理论上)可以使用许多步骤返回的信息
- 模型大小不会增加
- 每个时间步使用相同的权重矩阵——效率高

缺点：

- 计算慢，不能同时计算所有的隐藏状态，只能按序计算
- 很难获取到很多步之前的信息

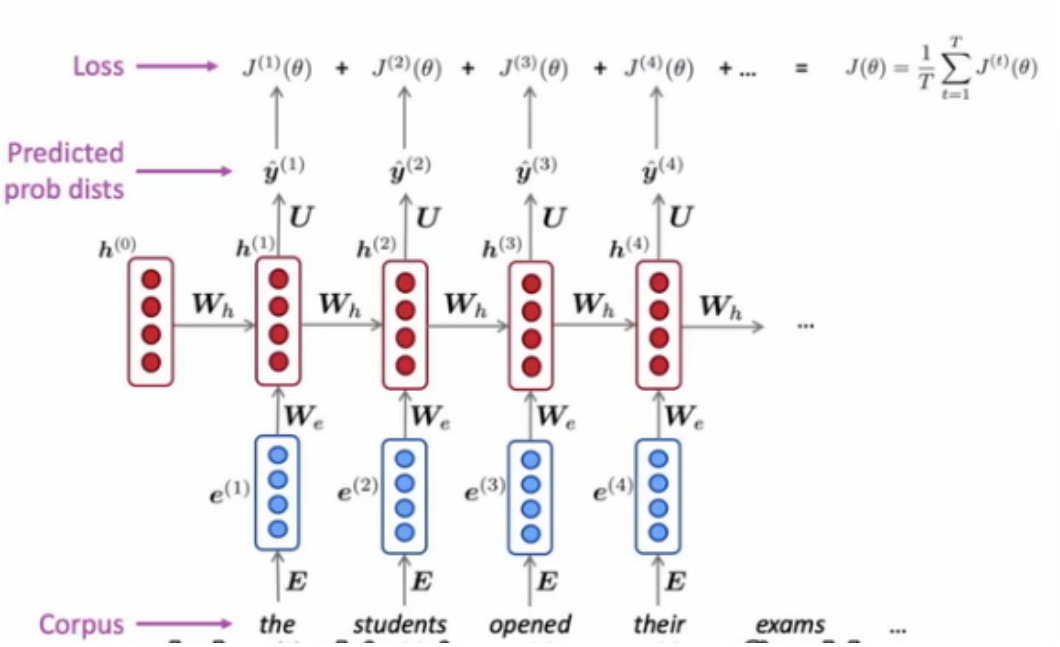
训练一个RNN语言模型

- 获取一个大的文本语料库，该语料库是x(1) ..x(T)的序列
- 输入到RNN-LM;计算每一步的输出分布。即预测每个单词的概率分布
- 步骤t上的损失函数为预测概率分布与下一个真实单词的交叉熵

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

- 求平均值，得到整个训练集的总体损失

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



CAVEAT：

- 计算消耗大
- 将次序列看作是句子或是文本

RNNs反向传播

重复权重的梯度是每次其出现时的梯度的总和

$$\frac{\partial J^{(t)}}{\partial \boldsymbol{W_h}} = \sum_{i=1}^{t} \frac{\partial J^{(t)}}{\partial \boldsymbol{W_h}}\bigg|_{(i)}$$

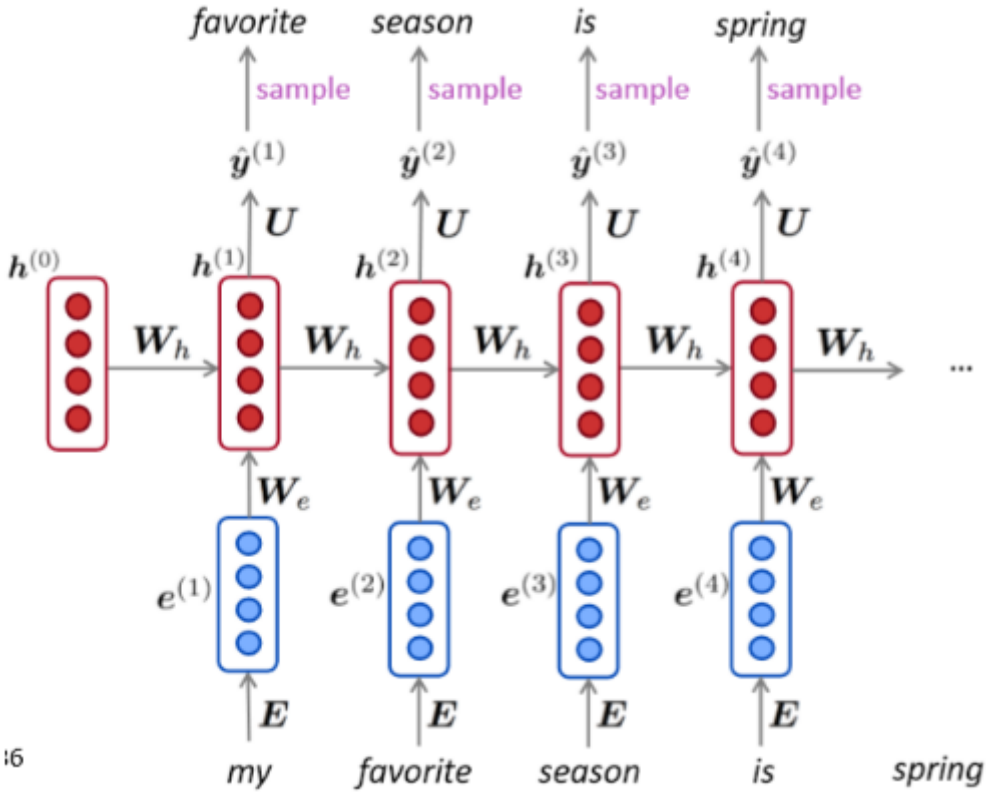
使用链式法则

$$\frac{\partial J^{(t)}}{\partial \boldsymbol{W_h}} = \sum_{i=1}^{t} \frac{\partial J^{(t)}}{\partial \boldsymbol{W_h}}\bigg|_{(i)} \boxed{\frac{\partial \boldsymbol{W_h}|_{(i)}}{\partial \boldsymbol{W_h}}} \quad (= 1)$$

$$= \sum_{i=1}^{t} \frac{\partial J^{(t)}}{\partial \boldsymbol{W_h}}\bigg|_{(i)}$$

使用RNN生成文本

与n-gram语言模型一样，可以使用RNN语言模型通过重复采样生成文本。采样输出是下一步的输入。



#课堂问题：前后引号匹配，当句子越长，后引号的概率增加，但也可能不出现后引号

#课堂问题：权重矩阵的维度：
Wh的维度为n* n；如果嵌入的维度是d，那么We的维度为n* d或d* n.

#课堂问题：RNN是否可以和手工规则结合？例如Beam Serach，但是可能很难做到

评估语言模型

语言模型的标准评价指标是perplexity

$$\text{perplexity} = \prod_{t=1}^{T} \left(\frac{1}{P_{\text{LM}}(\boldsymbol{x}^{(t+1)} \mid \boldsymbol{x}^{(t)}, \ldots, \boldsymbol{x}^{(1)})} \right)^{1/T}$$

Normalized by number of words

Inverse probability of corpus, according to Language Model

perplexity等于交叉熵损失J(theta)的指数

$$= \prod_{t=1}^{T} \left(\frac{1}{\hat{\boldsymbol{y}}^{(t)}_{\boldsymbol{x}_{t+1}}} \right)^{1/T} = \exp\left(\frac{1}{T} \sum_{t=1}^{T} -\log \hat{\boldsymbol{y}}^{(t)}_{\boldsymbol{x}_{t+1}} \right) = \exp(J(\theta))$$

perplexity越小越好

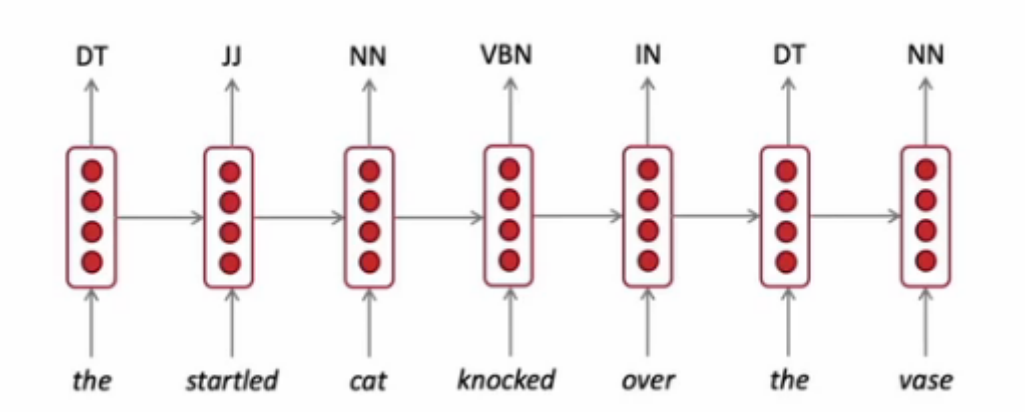
why Language Modelling is important

- 语言建模是一项基准任务，它可以帮助我们衡量在理解语言方面的进展。
- 语言建模是许多NLP任务的子组件，特别是那些涉及生成文本或估计文本概率的任务:
  - 预测性打字
  - 语音识别
  - 手写识别
  - 拼写/语法纠正
  - 作者识别
  - 机器翻译
  - 摘要
  - 对话

回顾

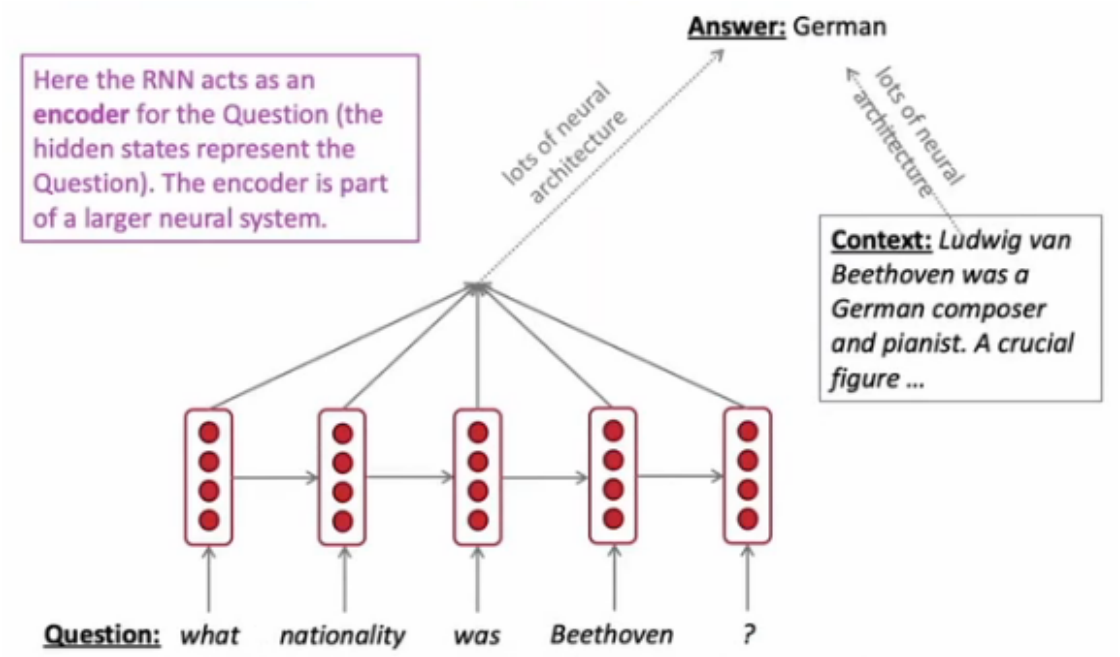
- 语言模型:预测下一个单词的系统
- 递归神经网络:
  - 取任意长度的顺序输入
  - 在每一步上应用相同的重量
  - 可以有选择地产生输出的每一步
- RNN 不等于 语言模型
  RNN是建立语言模型的一个很好的方式
- 但RNN能做到的有更多

标记或命名实体



句子分类

作为编码器模块



在这里，RNN充当问题的编码器，这个编码器是更大神经系统的一部分

生成文本

例如语音识别、机器翻译、摘要