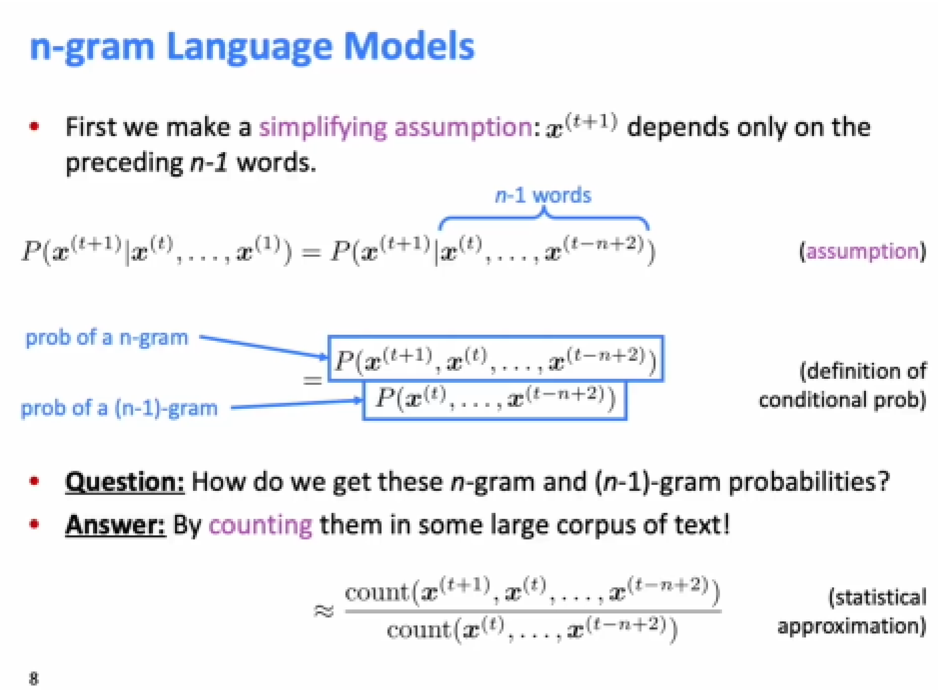
Language Modeling

Language Modeling是预测**序列的下一个元素**的任务。比如输入联想。

n-gram

n-gram指n个词一组，作为一个样本。idea是从数据库里找到大量n词一组的样本，直接通过计数计算条件概率。

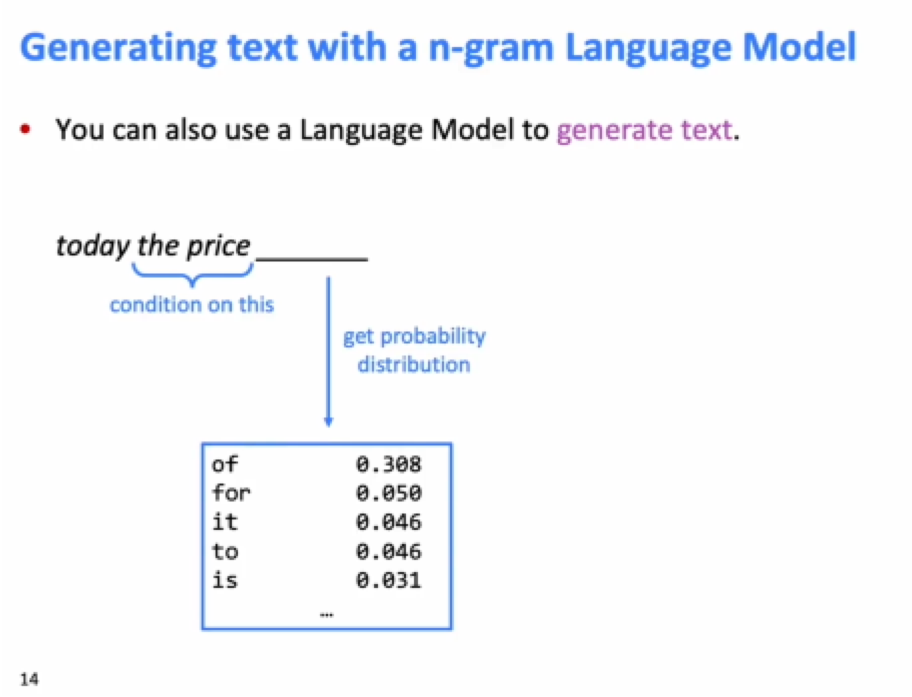
首先，我们简单假设第i个词出现的概率只取决于(i-n,...,i-1)这n个词(顺序必须也固定):

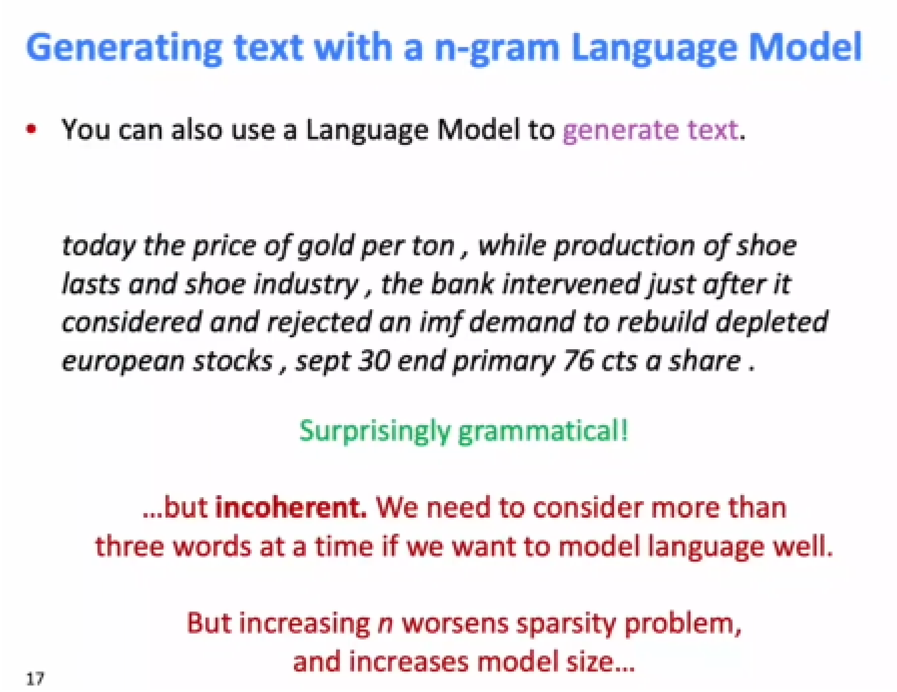


n-gram方法的缺点

- 可能取决于更之前的context，不在n-gram中
- sparsity problem：分子可能为0，需要增加假样本(数据平滑)。分母可能为0，需要从n-gram退化到(n-1)-gram
- storage problem：需要存储语料库中所有n-gram。

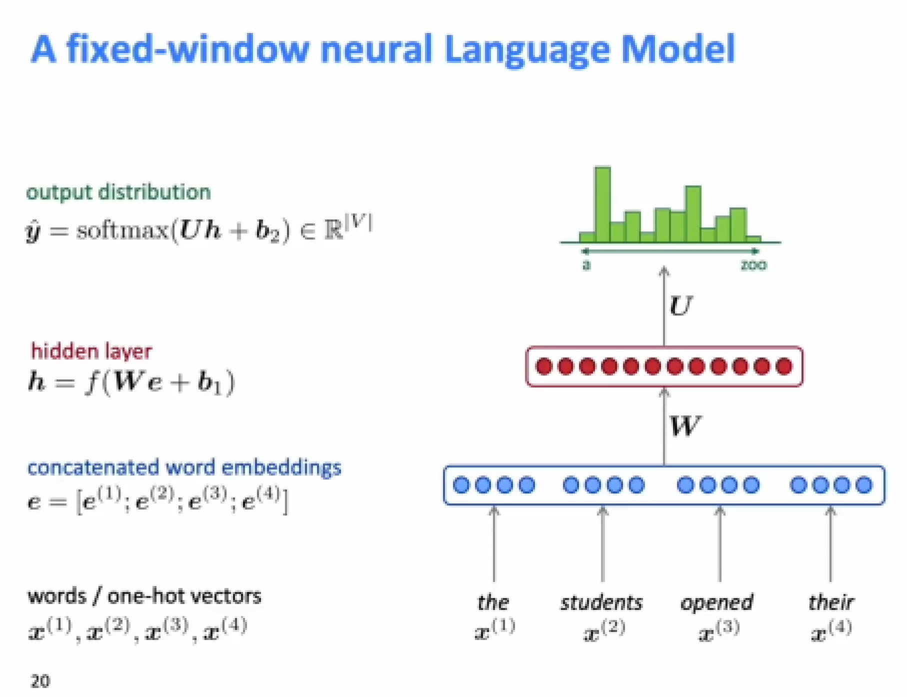
可以用language model来生成text，即逐一生成单词:





Build a neural language model

输入(50*windowsize,)维的向量，经过[50, 50*windowsize]的W，再乘以U，得到10000个单词各自生成的概率。



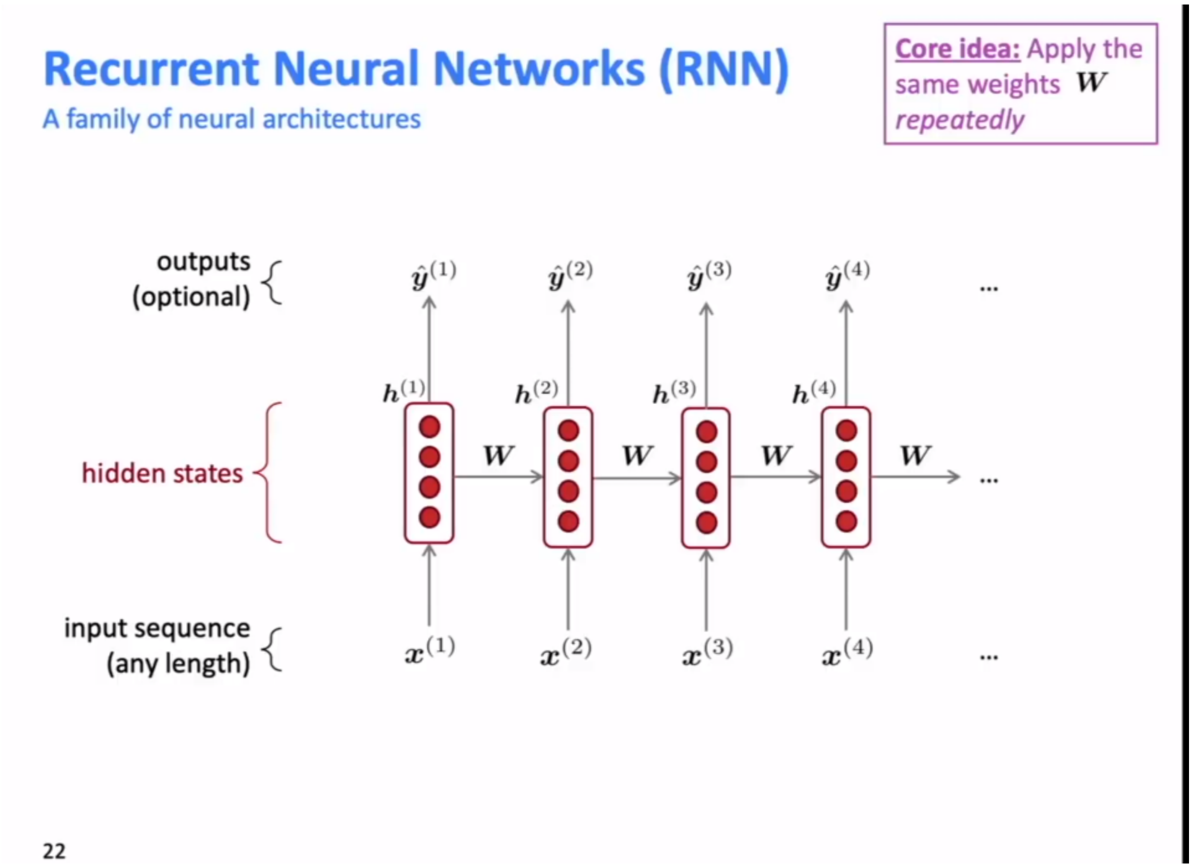
优点

- 没有sparsity problem
- 没有storage problem，不用存n-gram

缺点

- window太小，可能失去上下文
- window太大，W也要增大
- $x^{(1)}$和$x^{(2)}$在权重上没有联系

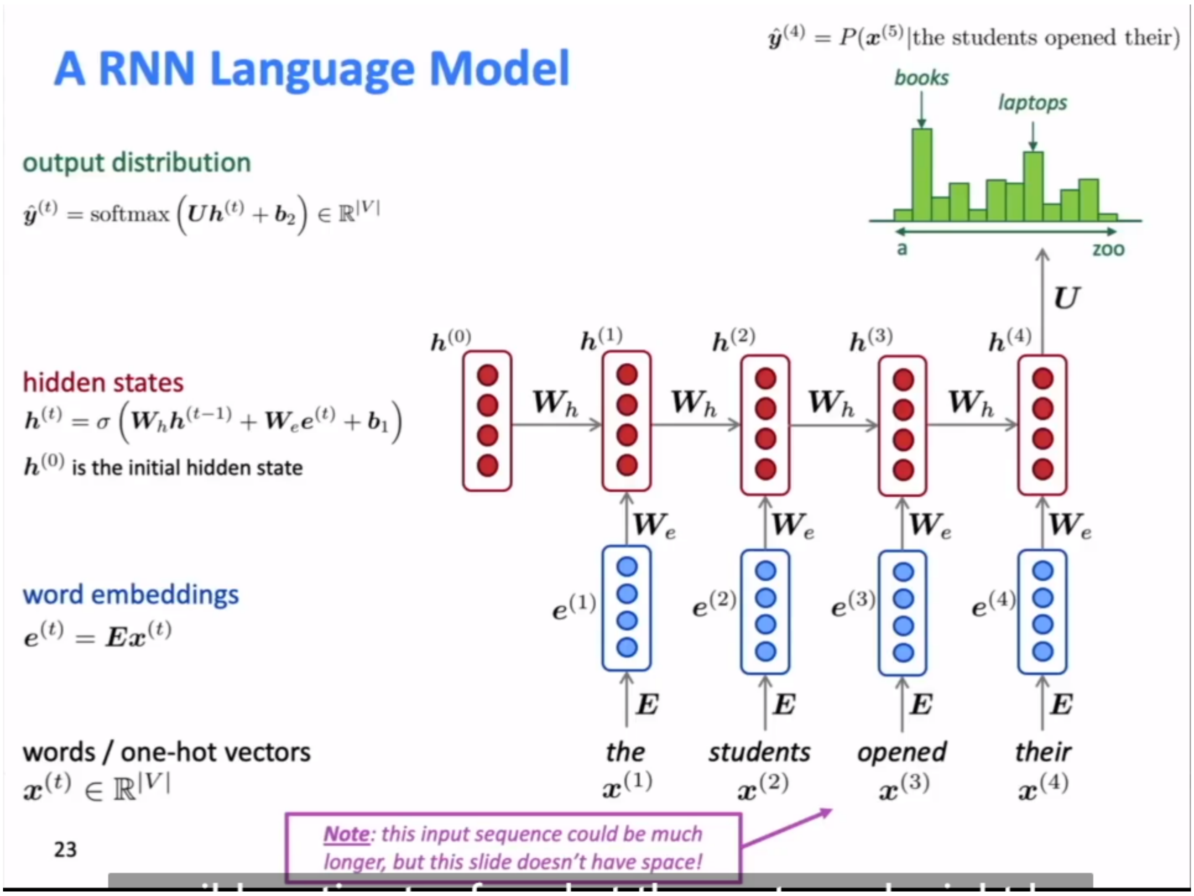
Recurrent Neural Networks (RNN)



上图是RNN的简图。

注：

- h是和时序有关的隐藏层
- y可以输出，也可以不输出
- W变换是每个隐藏层共享的



上图是RNN的细节图和推导。wordvec可以训练，也可以不训练。

优点

- 可以处理任意长度输入，不用固定windowsize
- 每一步都包含前面每一步的信息
- 因为共享权重，模型大小不会随输入变大
- 因为共享权重，不同输入会经过相同的权重变换
- 因为共享权重，学到的是更general的权重，而不是针对某些样本

缺点

- 计算很慢，必须串行计算
- 很难得到很多step之前的信息

注:

- 假设hidden unit有n个units，那么W_h维度是(n, n)，W_e维度是(n, d)

Train a RNN language model

需手推:

Training a RNN Language Model

- Get a big corpus of text which is a sequence of words $\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(T)}$
- Feed into RNN-LM; compute output distribution $\hat{\boldsymbol{y}}^{(t)}$ *for every step t.*
  - i.e. predict probability dist of *every word*, given words so far
- Loss function on step *t* is cross-entropy between predicted probability distribution $\hat{\boldsymbol{y}}^{(t)}$, and the true next word $\boldsymbol{y}^{(t)}$ (one-hot for $\boldsymbol{x}^{(t+1)}$):

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

- Average this to get overall loss for entire training set:

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

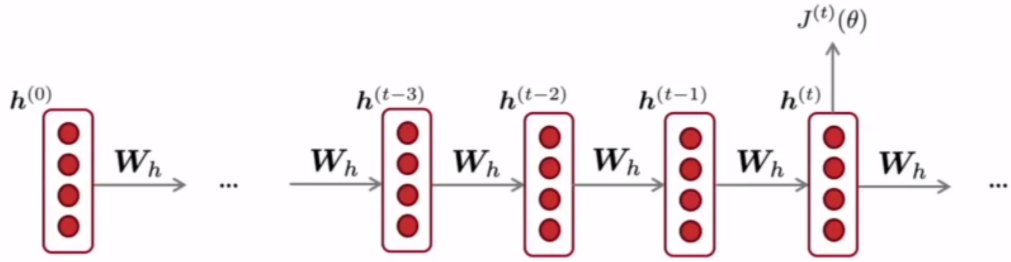
25

每个hidden layer都输出y计算loss消耗太大，一般训练时以sentence为单位，只计算最后一层的y的loss

BP for RNN

W_h的梯度是每一层loss产生梯度的和。#这里下标是i不是t，即$J^{(t)}$对每一层W_h的累加，具体到$\frac{\partial J^{(t)}}{\partial W_h}$还要展开用链式法则(下一章有讲，梯度消失问题)

Backpropagation for RNNs



Question: What's the derivative of $J^{(t)}(\theta)$ w.r.t. the repeated weight matrix $\boldsymbol{W}_h$?

Answer: $$\frac{\partial J^{(t)}}{\partial \boldsymbol{W_h}} = \sum_{i=1}^{t} \left. \frac{\partial J^{(t)}}{\partial \boldsymbol{W_h}} \right|_{(i)}$$

"The gradient w.r.t. a repeated weight is the sum of the gradient w.r.t. each time it appears"

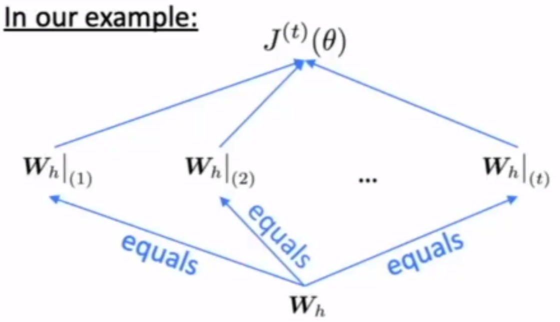
Why?

Backpropagation for RNNs: Proof sketch

- Given a multivariable function $f(x, y)$, and two single variable functions $x(t)$ and $y(t)$, here's what the multivariable chain rule says:

$$\underbrace{\frac{d}{dt} f(x(t), y(t))}_{\text{Derivative of composition function}} = \frac{\partial f}{\partial x}\frac{dx}{dt} + \frac{\partial f}{\partial y}\frac{dy}{dt}$$

In our example:

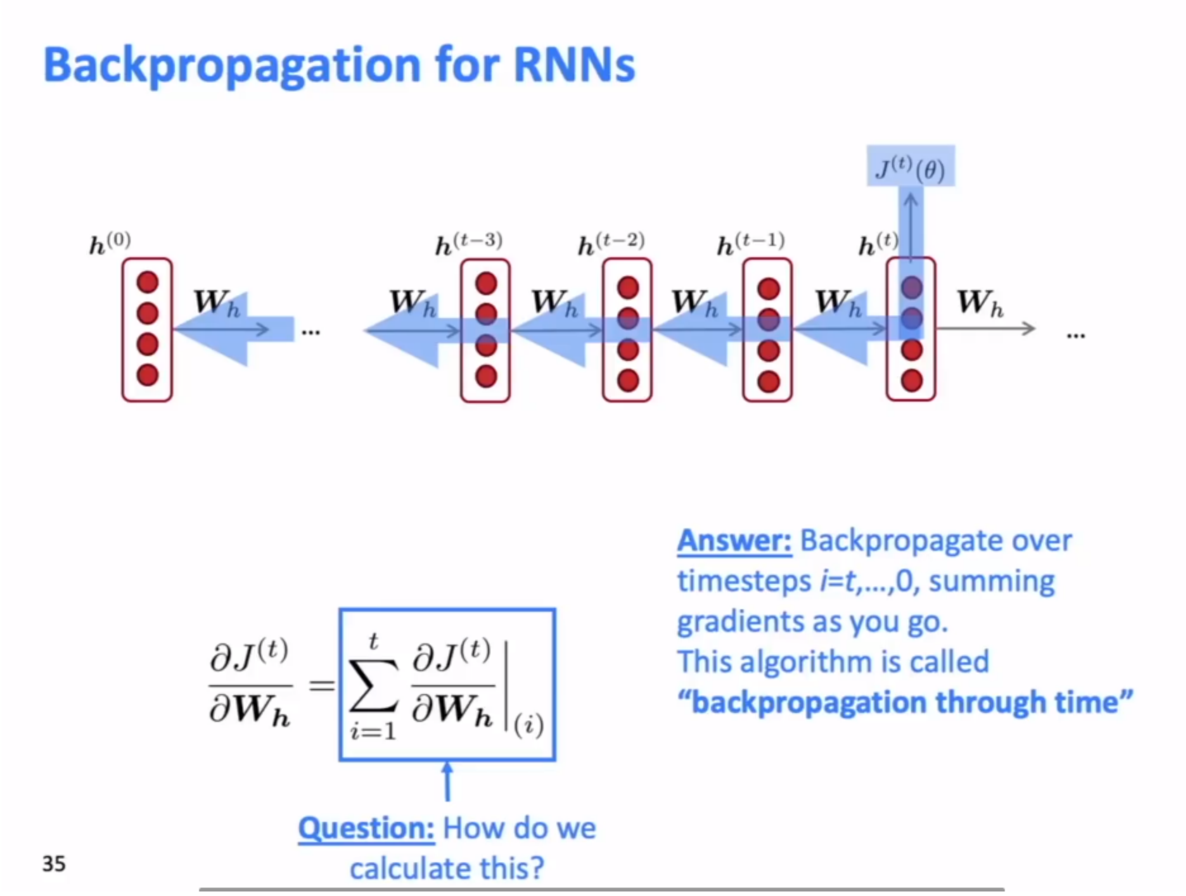


Apply the multivariable chain rule:

$$\frac{\partial J^{(t)}}{\partial \boldsymbol{W}_h} = \sum_{i=1}^{t} \left. \frac{\partial J^{(t)}}{\partial \boldsymbol{W}_h} \right|_{(i)} \underbrace{\boxed{\frac{\partial \boldsymbol{W}_h|_{(i)}}{\partial \boldsymbol{W}_h}}}_{= 1}$$

$$= \sum_{i=1}^{t} \left. \frac{\partial J^{(t)}}{\partial \boldsymbol{W}_h} \right|_{(i)}$$

Source:
https://www.khanacademy.org/math/multivariable-calculus/multivariable-derivatives/differentiating-vector-valued-functions/a/multivariable-chain-rule-simple-version



注：

- 前向传播全部结束后，会计算W_h的总梯度，最后只梯度下降一次

Evaluation Language Models

perplexity是预测概率的倒数的乘积，越低越好。

Evaluating Language Models

- The standard evaluation metric for Language Models is perplexity.

$$\text{perplexity} = \underbrace{\prod_{t=1}^{T} \left(\frac{1}{P_{\text{LM}}(\boldsymbol{x}^{(t+1)} | \, \boldsymbol{x}^{(t)}, \ldots, \boldsymbol{x}^{(1)})} \right)}_{\text{Inverse probability of corpus, according to Language Model}}^{1/T}$$

Normalized by number of words

- This is equal to the exponential of the cross-entropy loss $J(\theta)$:

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

Lower perplexity is better!

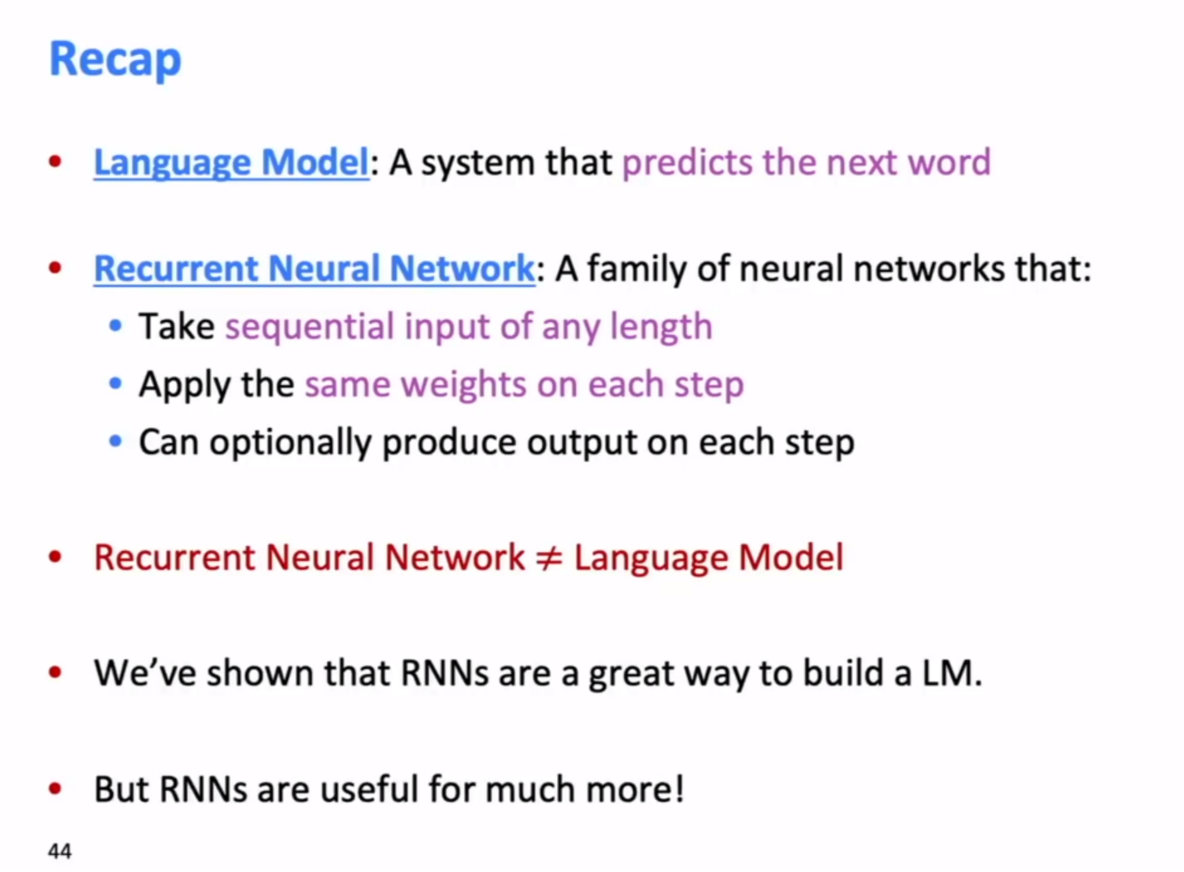
Why Language Modeling

Why should we care about Language Modeling?

- Language Modeling is a benchmark task that helps us measure our progress on understanding language
- Language Modeling is a subcomponent of many NLP tasks, especially those involving generating text or estimating the probability of text:
    - Predictive typing
    - Speech recognition
    - Handwriting recognition
    - Spelling/grammar correction
    - Authorship identification
    - Machine translation
    - Summarization
    - Dialogue
    - etc.

often it's kind of noisy and hard to make out what they're say

总结



注：RNN不是language model，但是可以用来构建一个language model。RNN还可以做很多别的事情。

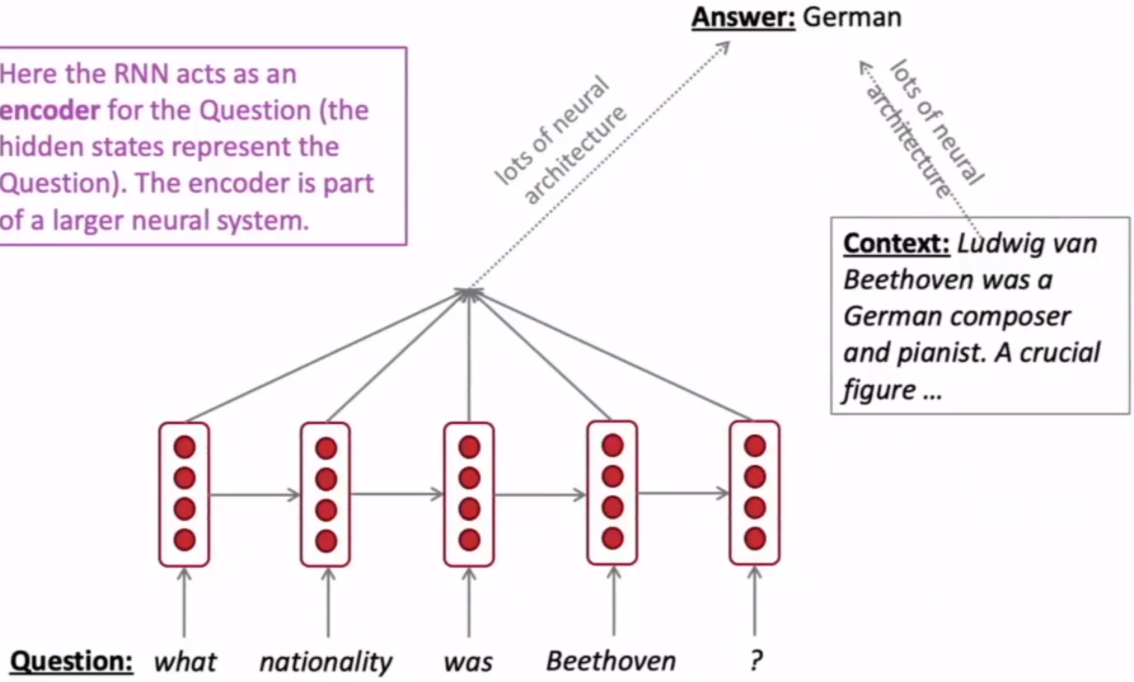
RNN还可以做：

- part of speech tagging
- sentence classification(like sentiment classification)
- general purpose encoder module(like question answering, machine translation)
- generate text(like speech recognition)

RNNs can be used as an encoder module

e.g. question answering, machine translation, *many other tasks!*

Here the RNN acts as an **encoder** for the Question (the hidden states represent the Question). The encoder is part of a larger neural system.



A note on terminology

RNN described in this lecture = "vanilla RNN" 

Next lecture: You will learn about other RNN flavors

like GRU  and LSTM  and multi-layer RNNs 

By the end of the course: You will understand phrases like *"stacked bidirectional LSTM with residual connections and self-attention"*