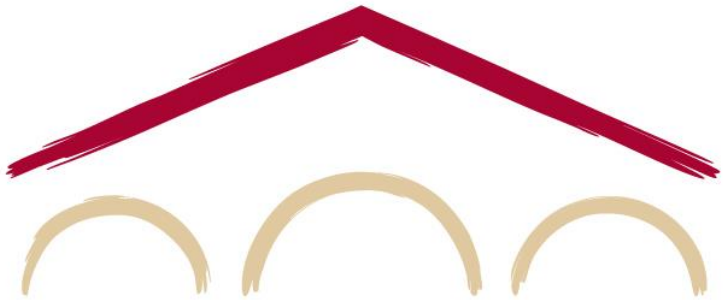
Natural Language P
与 Deep Learning
CS224N/Ling284



Diyi Yang

Lecture 4：语言模型与循环神经网络

课程计划

Ｌｅｃｔｕｒｅ　４：语言建模　＋　ＲＮＮ

1. A new NLP task: **Language Modeling** （２０　分钟）

This is the most important concept in the 课程！引出了大部分现代　ＮＬＰ

↓ 推动了

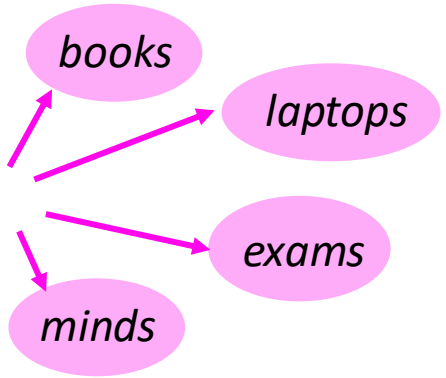
2. Language models with neural nets: **Recurrent Neural Networks (RNNs)** （２５　分钟）
3. Problems with RNNs: exploding and vanishing gradients (20 mins)
4. 机器翻译（１０　分钟）

Reminder: 作业　２　—　截止　１　月　２２　日，周四

1． 语言建模

- 语言建模　　是预测下一个词是什么的任务

学生们打开了他们的 _ _ _ _ _ _



- 更正式地：给定一个词序列，
  计算下一个词的概率分布：

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

其中……可以是词表中的任意词　　$V=\{\boldsymbol{w}_1,\ldots,\boldsymbol{w}_{|V|}\}$

- A system that does this is called a 语言模型

语言建模

- You can also think of a Language Model as a system that
  给一段文本分配概率

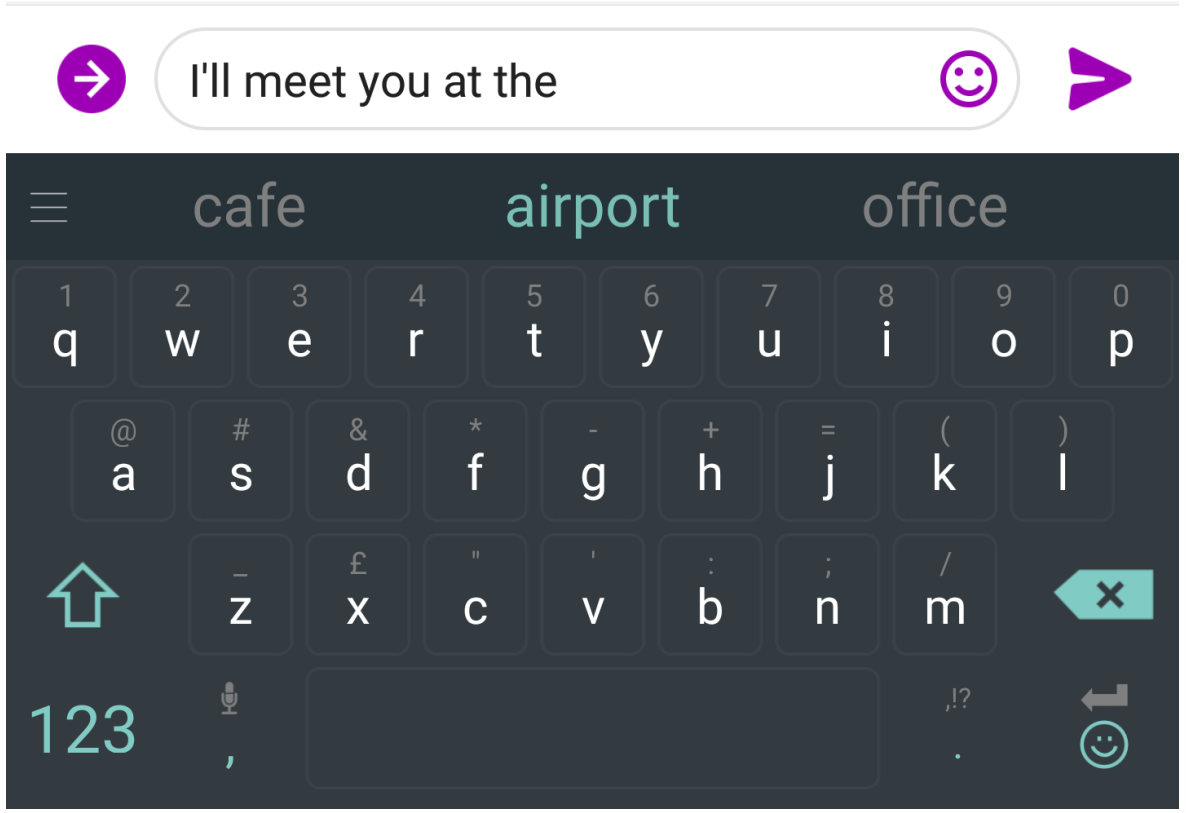
- For example, if we have some text $\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(T)}$, then the
  这段文本的概率（根据语言模型）是：

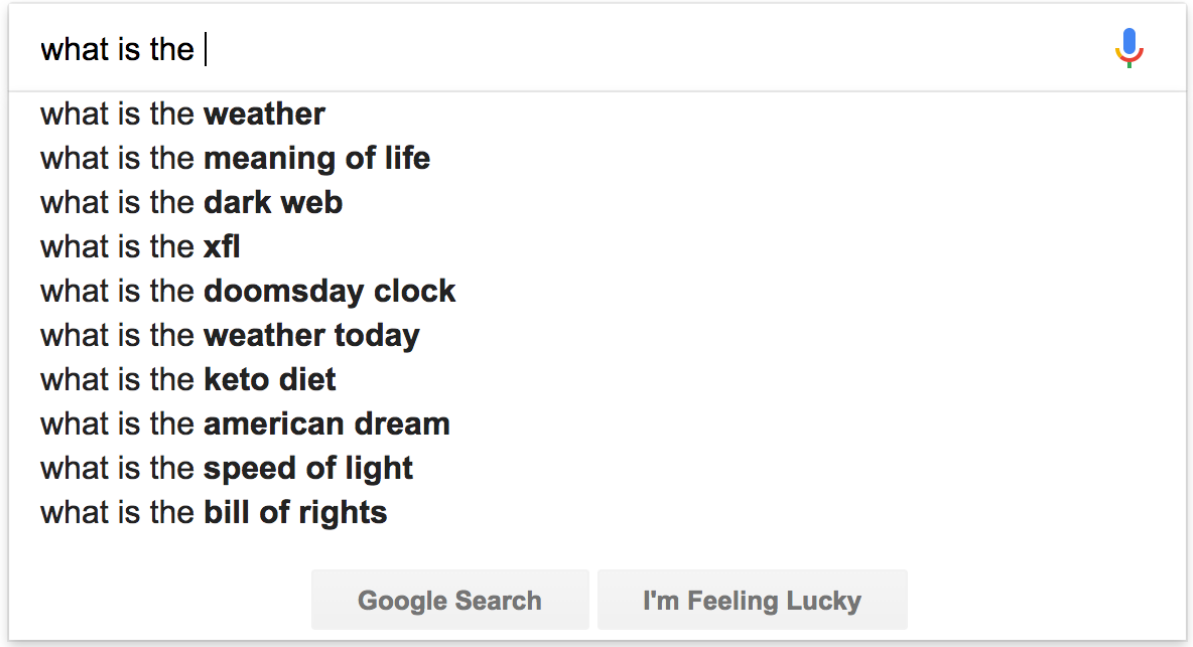
$$P(\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(T)}) = P(\boldsymbol{x}^{(1)}) \times P(\boldsymbol{x}^{(2)} | \boldsymbol{x}^{(1)}) \times \cdots \times P(\boldsymbol{x}^{(T)} | \boldsymbol{x}^{(T-1)}, \ldots, \boldsymbol{x}^{(1)})$$

$$= \prod_{t=1}^{T} P(\boldsymbol{x}^{(t)} | \boldsymbol{x}^{(t-1)}, \ldots, \boldsymbol{x}^{(1)})$$

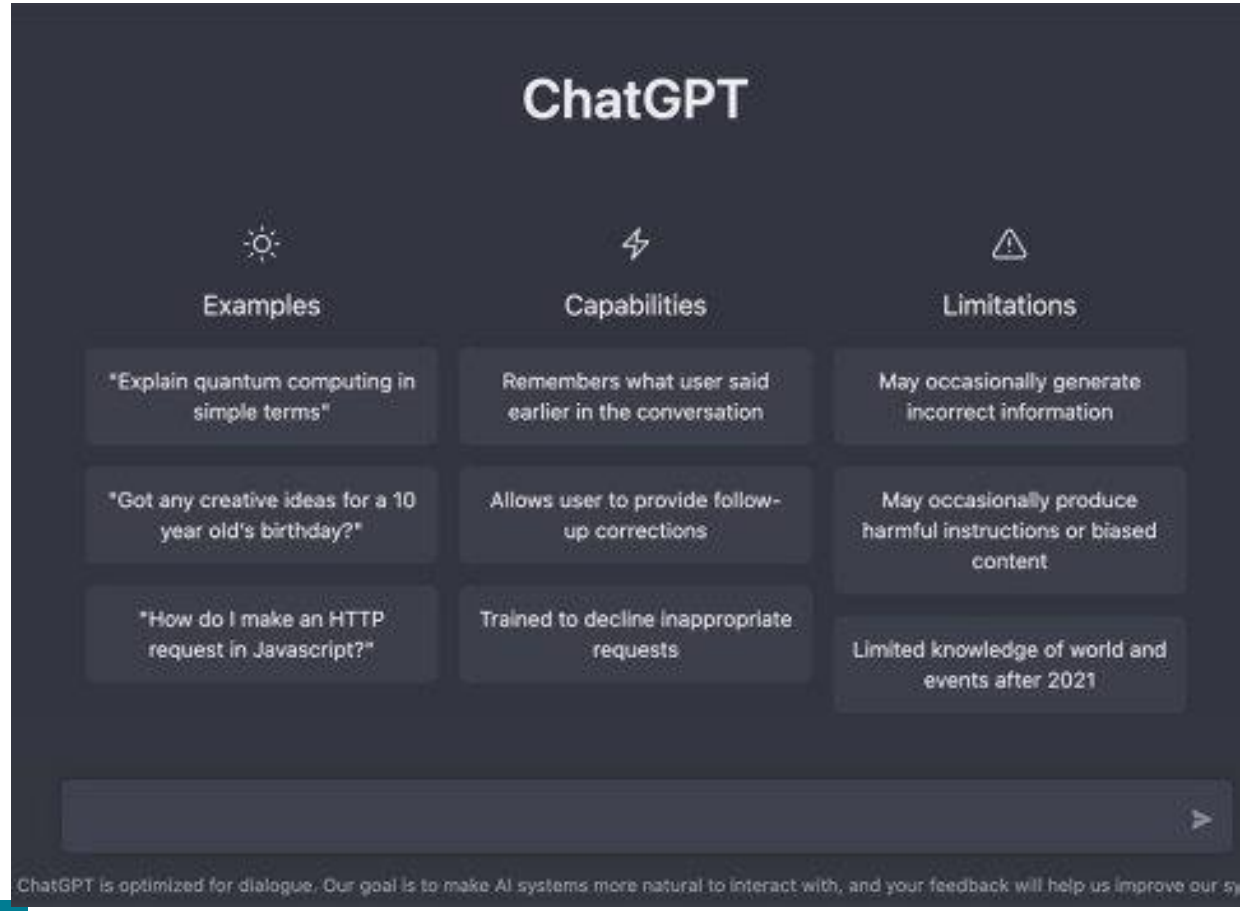
这就是我们的语言模型提供的

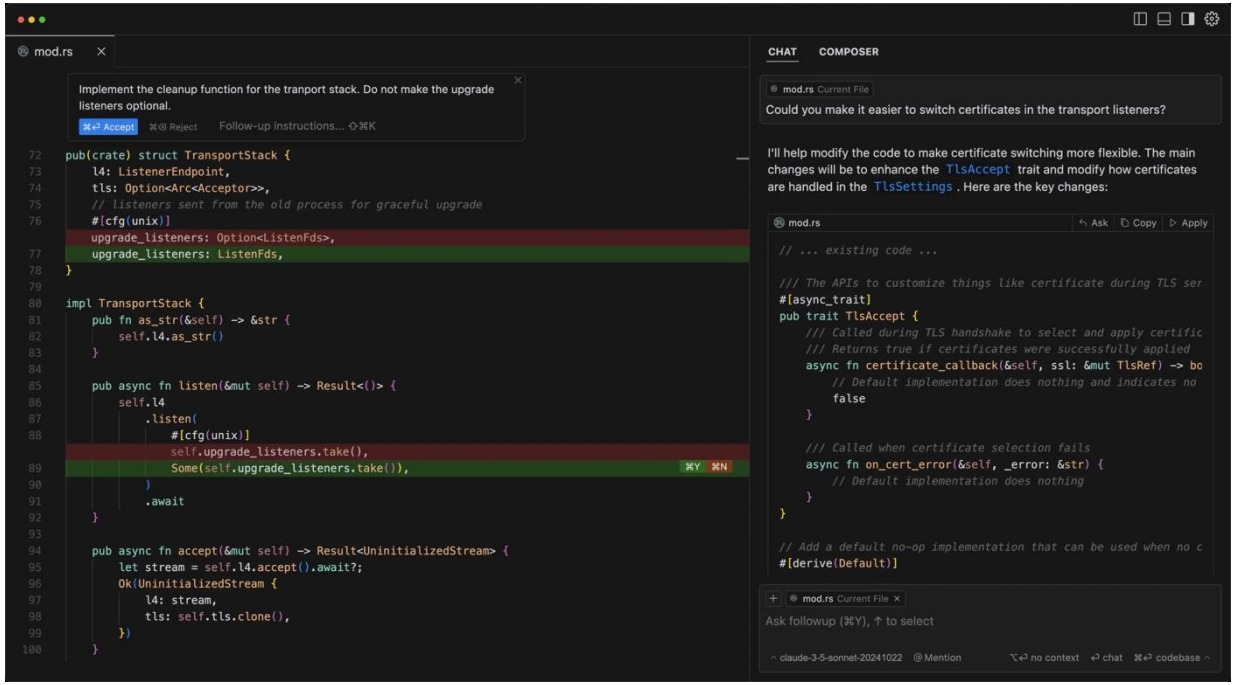
你每天都在使用语言模型！





你每天都在使用语言模型！





为什么我们应该关心语言建模？

- Language Modeling is a benchmark task that helps us measure our progress on 预测语言使用
- Language Modeling is a 子组件 of many NLP tasks, especially those involving 生成文本 or 估计文本的概率 :
  - 预测输入
  - 语音识别
  - 手写识别
  - 拼写 / 语法纠正
  - 作者身份识别
  - 机器翻译
  - 摘要
  - 对话
  - etc.
- Everything else in NLP has been rebuilt upon Language Modeling: ＣｈａｔＧＰＴ 就是一个

你能用下一词预测做什么？

- 一个足够强的（！）语言模型可以做很多很多事情

Stanford University 位于加利福尼亚州_ _ _ _ _ _ _ _ _ _。 [Trivia]

我把_ _ _叉子放在桌上。 [syntax]

那个女人穿过街道，回头越过_ _ _肩膀查看交通状况。 [coreference]

I went to the ocean to see the fish, turtles, seals, and _____. [lexical semantics/topic]

Overall, the value I got from the two hours watching it was the sum total of the popcorn and the drink. The movie was ___. [sentiment]

Iroh went into the kitchen to make some tea. Standing next to Iroh, Zuko pondered his destiny. Zuko left the _____. [some reasoning – this is harder]

I was thinking about the sequence that goes 1, 1, 2, 3, 5, 8, 13, 21, ____ [some basic arithmetic]

n - g r a m 语言模型

学生们打开了他们的 _ _ _ _ _ _

- 问题 ：如何学习语言模型？
- 回答 (pre- Deep Learning): learn an *n* - g r a m 语言模型 !

- 定义： An *n*-gram is a chunk of *n* consecutive words.
  - unigrams: “the”, “students”, “opened”, ”their”
  - bigrams: “the students”, “students opened”, “opened their”
  - trigrams: “the students opened”, “students opened their”
  - four-grams: “the students opened their”

- 想法 :Collect statistics about how frequent different n-grams are and use these to 预测下一个词。

n-gram 语言模型

- First we make a 马尔可夫假设 : $x^{(t+1)}$ depends only on the preceding *n*-1 words

$$P(\boldsymbol{x}^{(t+1)}|\boldsymbol{x}^{(t)},\ldots,\boldsymbol{x}^{(1)}) = P(\boldsymbol{x}^{(t+1)}|\boldsymbol{x}^{(t)},\ldots,\boldsymbol{x}^{(t-n+2)})$$

n-1 个词

（假设）

n-gram 的概率

$$= \frac{P(\boldsymbol{x}^{(t+1)},\boldsymbol{x}^{(t)},\ldots,\boldsymbol{x}^{(t-n+2)})}{P(\boldsymbol{x}^{(t)},\ldots,\boldsymbol{x}^{(t-n+2)})}$$

(n-1)-gram 的概率

(definition of conditional prob)

- 问题： How do we get these *n*-gram and (*n*-1)-gram 概率？
- 回答： By 计数 them in some large corpus of text!

$$\approx \frac{\text{count}(\boldsymbol{x}^{(t+1)},\boldsymbol{x}^{(t)},\ldots,\boldsymbol{x}^{(t-n+2)})}{\text{count}(\boldsymbol{x}^{(t)},\ldots,\boldsymbol{x}^{(t-n+2)})}$$

(statistical 近似)

n-gram 语言模型：示例

Suppose we are learning a 4-gram Language Model.

~~当监考老师启动计时器时，学生们打~~开了他们的 _ _ _ _ _

~~当监考老师启动计时器时，学生们打~~ discard

开了他们的 _ _ _ _ _ 以此为条件

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

例如，假设在语料库中：

- "students opened their" occurred 1000 times
- "students opened their books" occurred 400 times
  - → P(books | students opened their) = 0.4
- "students opened their exams" occurred 100 times
  - → P(exams | students opened their) = 0.1

Should we have discarded the "proctor" context?

稀疏性问题 with n-gram Language Models

稀疏性问题 1

问题： What if *"students opened their w"* never occurred in data? Then *w* has 概率为 0 ！

（部分）解决方案：Add small δ to the count for every $w \in V$. This is called 平滑 .

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

稀疏性问题 2

问题： What if *"students opened their"* never occurred in data? Then we can't calculate probability for *any w*!

(Partial) Solution: Just condition on *"opened their"* instead. This is called *backoff*.

Note: Increasing *n* makes sparsity problems 更糟。
Typically, we can't have *n* bigger than 5.

存储问题

with n-gram Language Models

存储: Need to store count for all *n*-grams you saw in the corpus.

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

Increasing *n* or increasing

语料库增大模型大小！

n-gram 语言模型的实践

- You can build a simple trigram Language Model over a
  １７０ 万词的语料库（Ｒｅｕｔｅｒｓ），在你的笔记本上几秒内完成 *

商业和财经新闻

today the _____

get probability distribution

company	0.153
bank	0.153
price	0.077
italian	0.039
emirate	0.039
…	

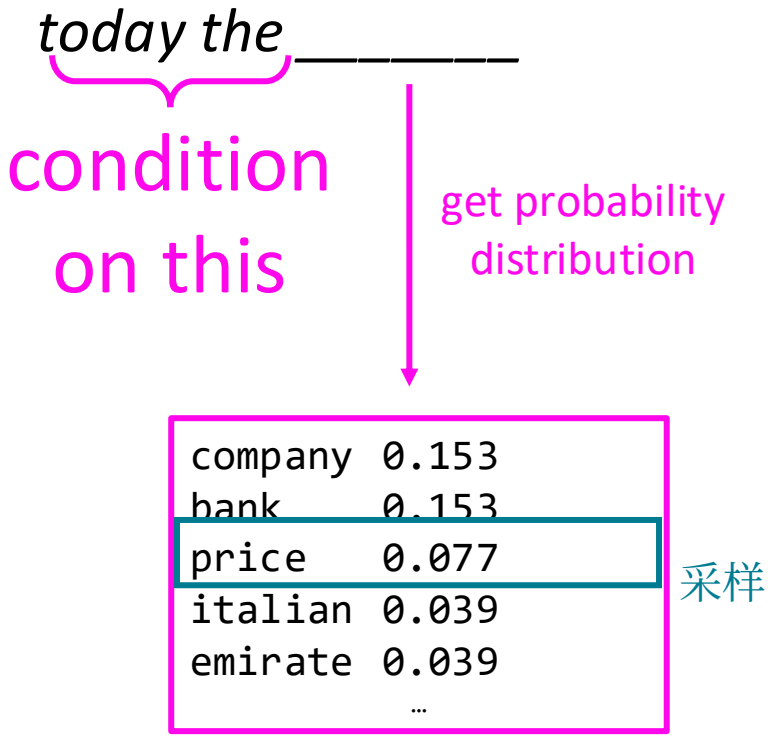
稀疏性问题：
not much granularity in the probability distribution

否则看起来合理！

* 自己试试： https://nlpforhackers.io/language-models/

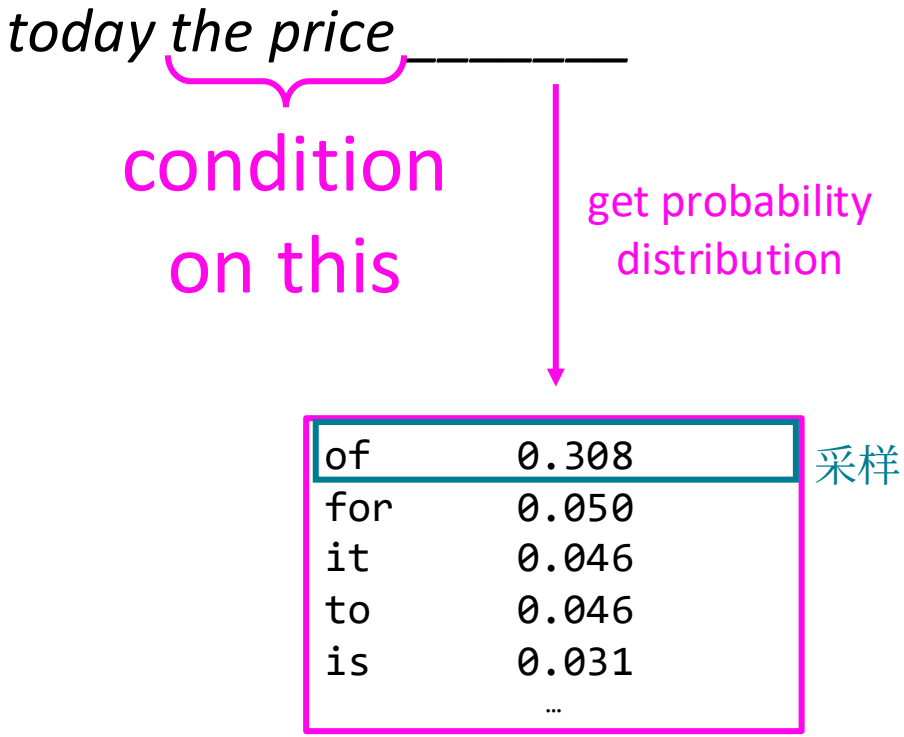
使用 n-gram 语言模型生成文本

You can also use a Language Model to 生成文本



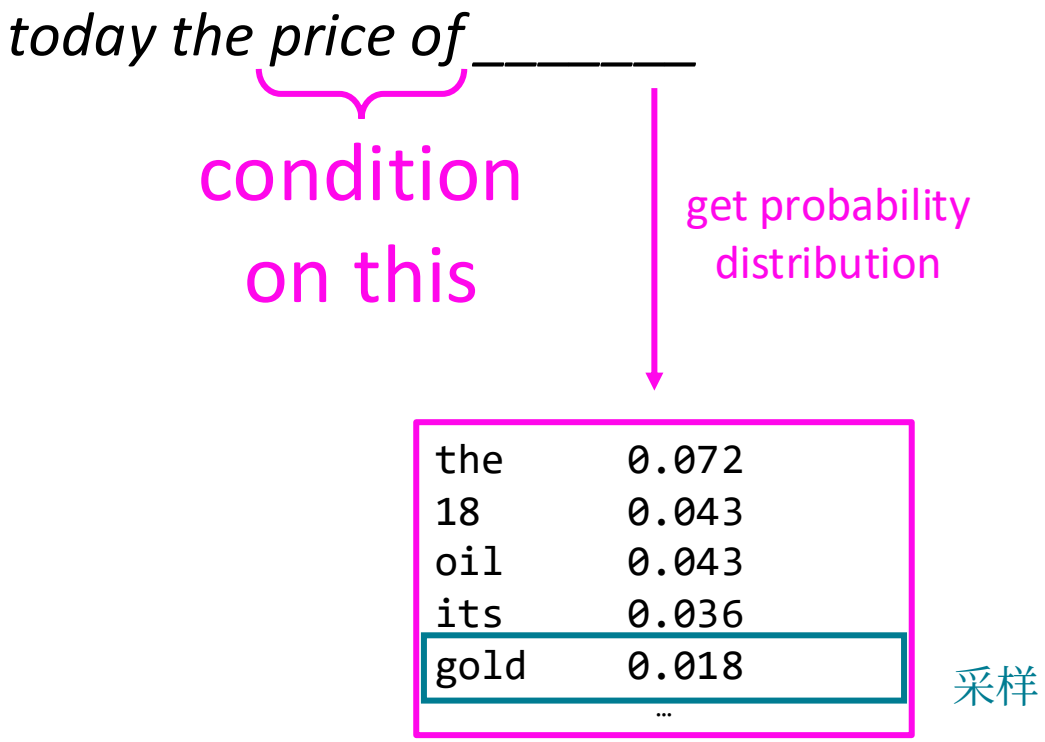
使用 n-gram 语言模型生成文本

You can also use a Language Model to 生成文本



使用 n-gram 语言模型生成文本

You can also use a Language Model to 生成文本



使用 n-gram 语言模型生成文本

You can also use a Language Model to 生成文本

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 .

语法出奇地正确！

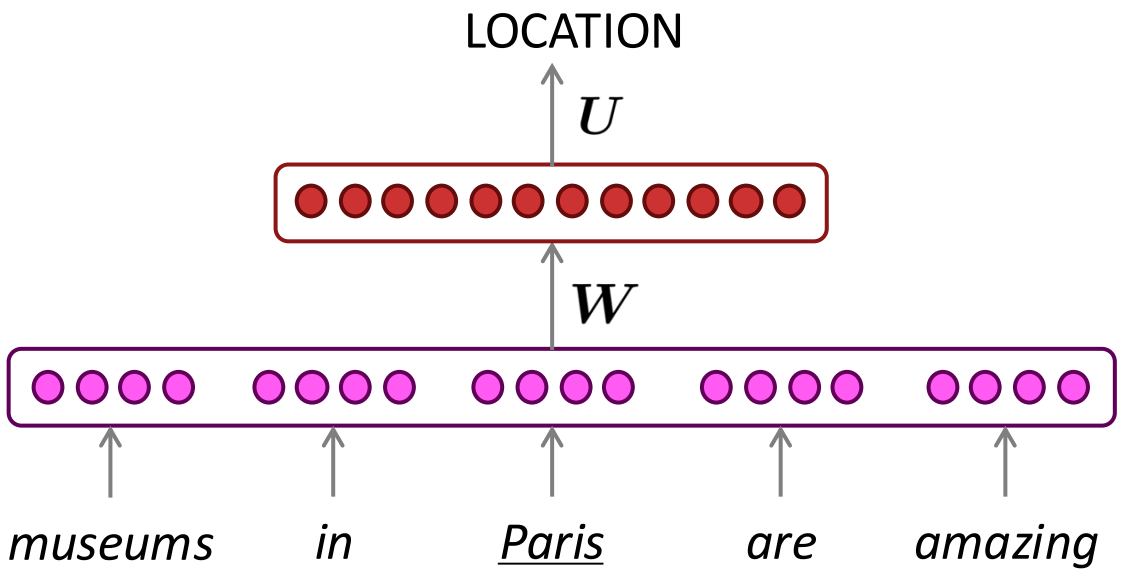
…but **incoherent.** We need to consider more than 如果我们想很好地建模语言，一次三个词是不够的。

But increasing *n* worsens sparsity problem, 并增加模型大小……

How to build a *neural* language model?

- 回顾语言建模任务：
  - 输入：词序列 $\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, \ldots, \boldsymbol{x}^{(t)}$
  - Output: prob. dist. of the next word $P(\boldsymbol{x}^{(t+1)} | \boldsymbol{x}^{(t)}, \ldots, \boldsymbol{x}^{(1)})$

- How about a 基于窗口的神经模型 ?
  - 我们看到这应用于命名实体识别：



固定窗口神经语言模型


as the proctor started the clock
the students opened their _____
discard
固定窗口

固定窗口神经语言模型

output distribution

$$\hat{\boldsymbol{y}} = \text{softmax}(\boldsymbol{U}\boldsymbol{h} + \boldsymbol{b}_2) \in \mathbb{R}^{|V|}$$

隐藏层

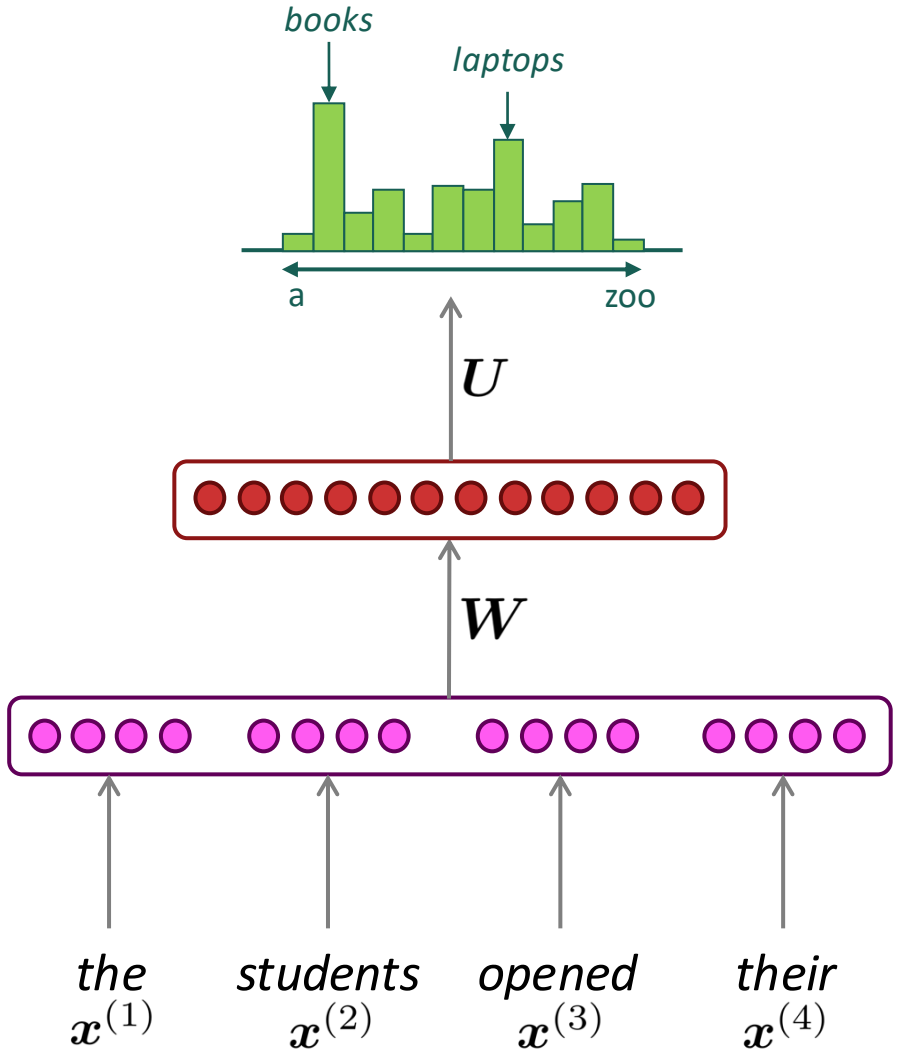
$$\boldsymbol{h} = f(\boldsymbol{W}\boldsymbol{e} + \boldsymbol{b}_1)$$

拼接的 word embeddings

$$\boldsymbol{e} = [\boldsymbol{e}^{(1)}; \boldsymbol{e}^{(2)}; \boldsymbol{e}^{(3)}; \boldsymbol{e}^{(4)}]$$

words / one-hot vectors

$$\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, \boldsymbol{x}^{(3)}, \boldsymbol{x}^{(4)}$$



固定窗口神经语言模型

大约：Y． Bengio 等人（2000/2003）：一个神经概率语言模型

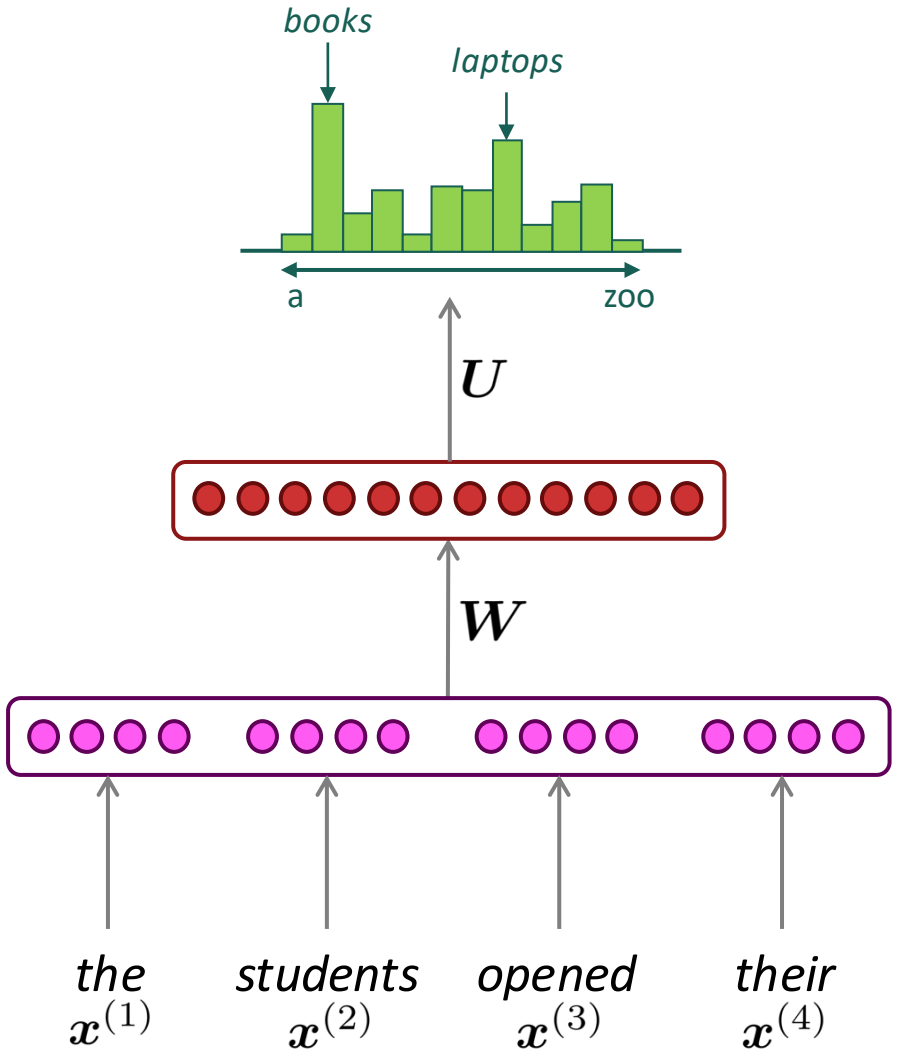
改进 over *n*-gram LM:

- 无稀疏性问题
- Don't need to store all observed *n*-grams

Remaining 问题 :

- Fixed window is 太小
- Enlarging window enlarges W
- 窗口永远不够大！
- $x^{(1)}$ and $x^{(2)}$ are multiplied by completely different weights in W. No symmetry in how the inputs are processed.

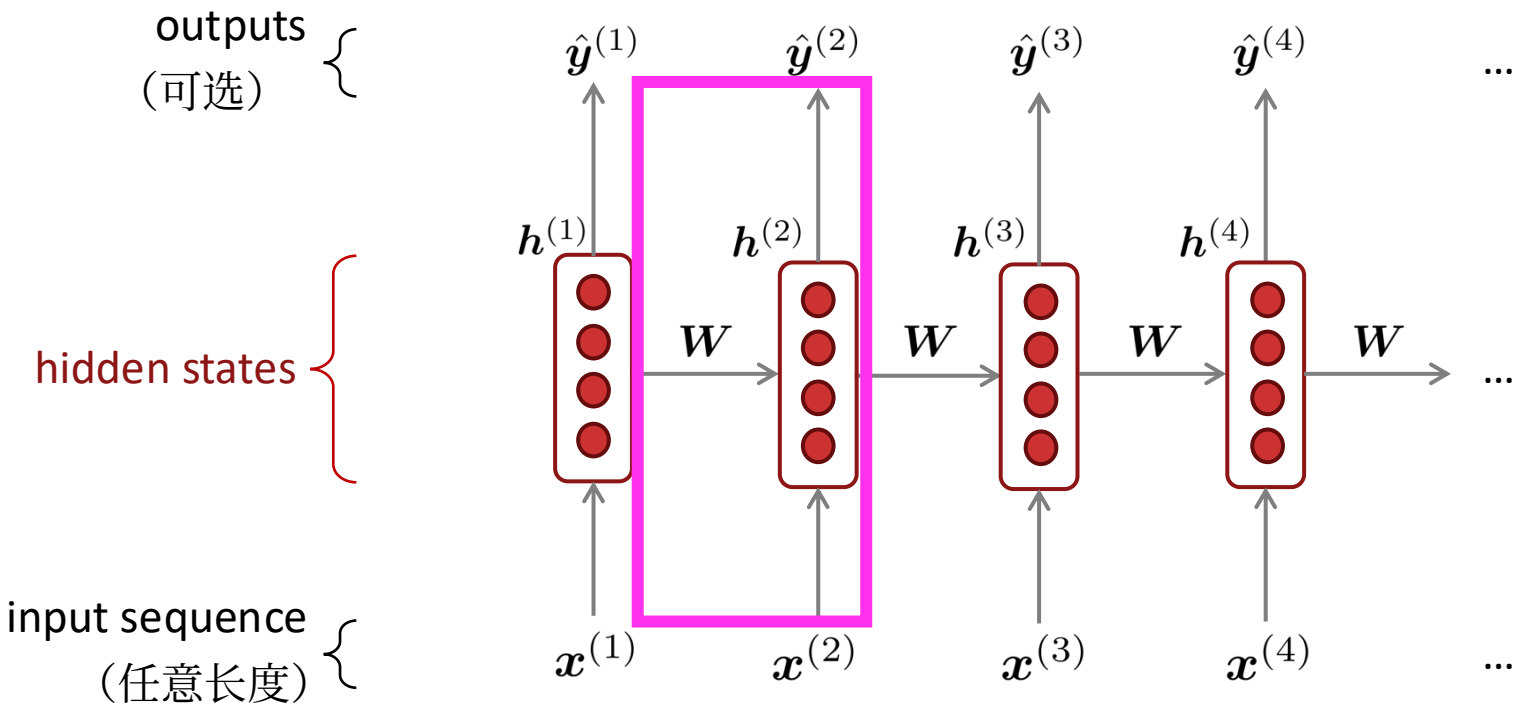
We need a neural architecture that can process 任意长度输入



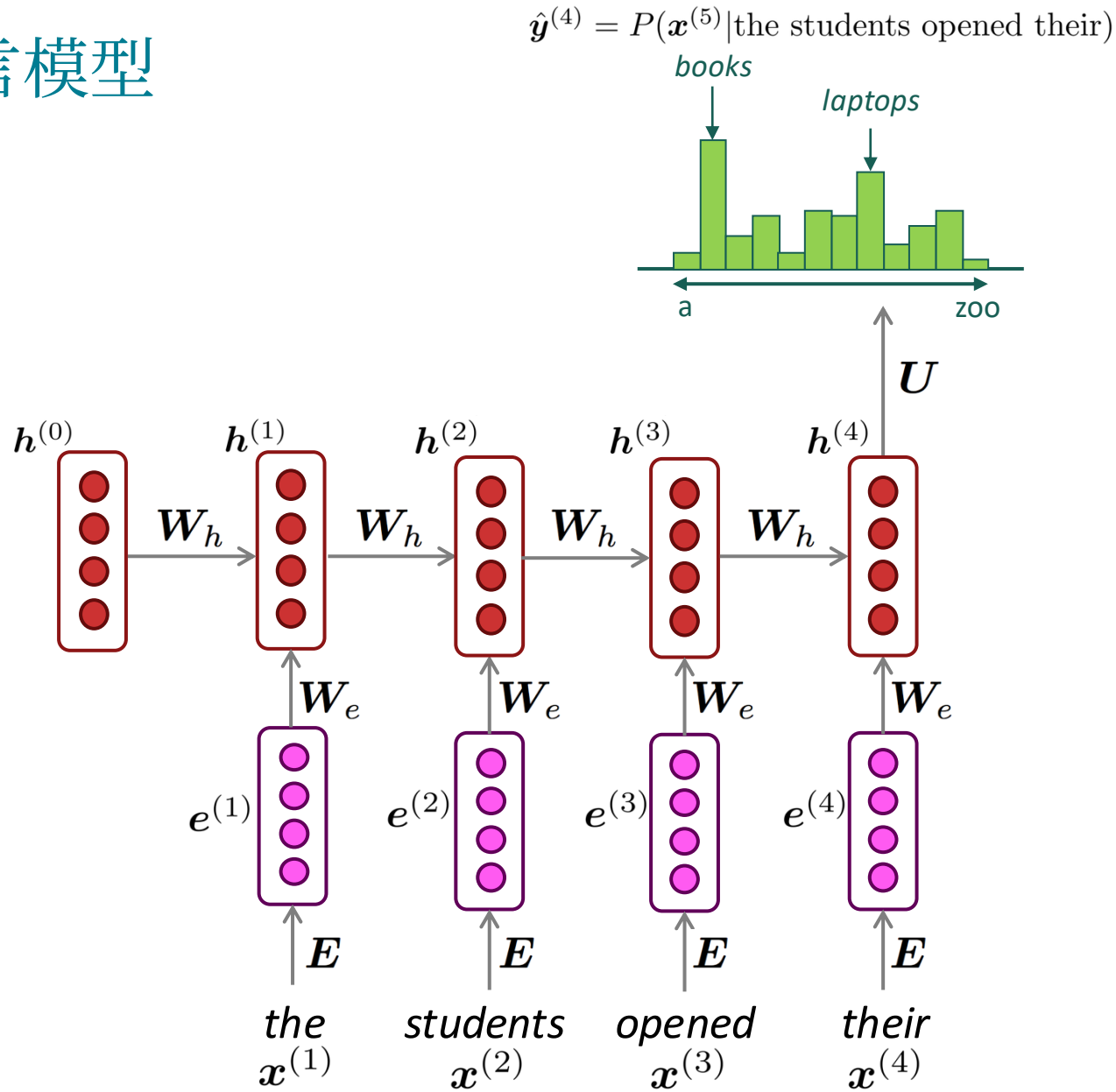
2． 循环神经网络（RNN）

一族神经架构

核心思想：Apply the same weights *W repeatedly*



一个简单的 RNN 语言模型



output distribution

$$\hat{\boldsymbol{y}}^{(t)} = \text{softmax}\left(\boldsymbol{U}\boldsymbol{h}^{(t)} + \boldsymbol{b}_2\right) \in \mathbb{R}^{|V|}$$

hidden states

$$\boldsymbol{h}^{(t)} = \sigma\left(\boldsymbol{W}_h\boldsymbol{h}^{(t-1)} + \boldsymbol{W}_e\boldsymbol{e}^{(t)} + \boldsymbol{b}_1\right)$$

$\boldsymbol{h}^{(0)}$ 是初始隐藏状态

word embeddings

$$\boldsymbol{e}^{(t)} = \boldsymbol{E}\boldsymbol{x}^{(t)}$$

words / one-hot vectors

$$\boldsymbol{x}^{(t)} \in \mathbb{R}^{|V|}$$

Note: *this input sequence could be much*

现在更长了！

RNN 语言模型

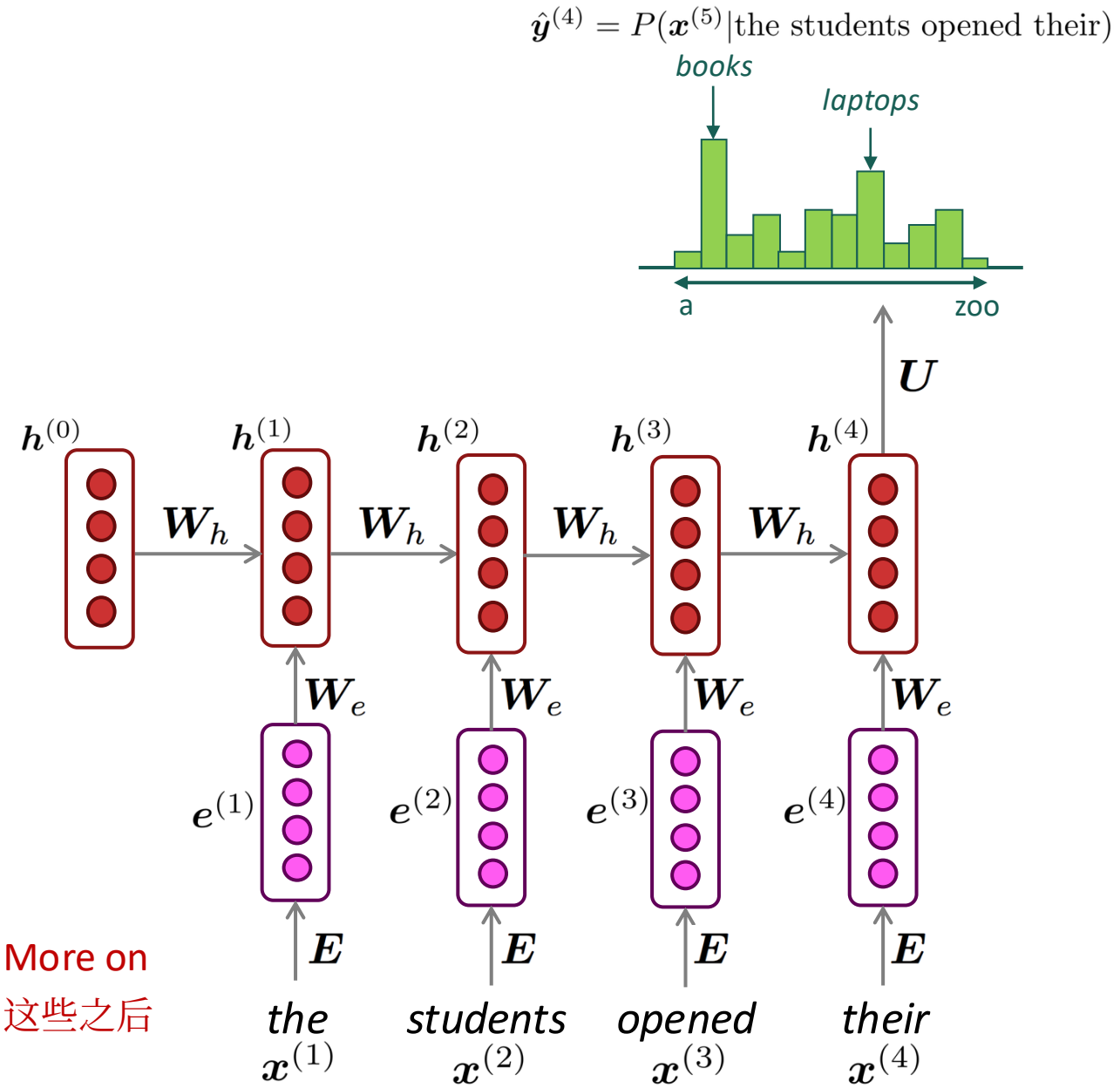
RNN 优点 :

- Can process any length input
- Computation for step *t* can (in 理论）使用来自……的信息 很多步之前
- Model size doesn't increase for 更长的输入上下文
- Same weights applied on every timestep, so there is symmetry 在输入的处理方式上。

RNN 缺点 :

- Recurrent computation is slow
- In practice, difficult to access info from many steps back

More on 这些之后



训练 RNN 语言模型

- Get a big corpus of text 这是一个词序列 $\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(T)}$
- Feed into RNN-LM; compute output distribution $\hat{\boldsymbol{y}}^{(t)}$ for 每个时间步 . t
  - i.e., predict probability dist of 每个词 , 给定到目前为止的词

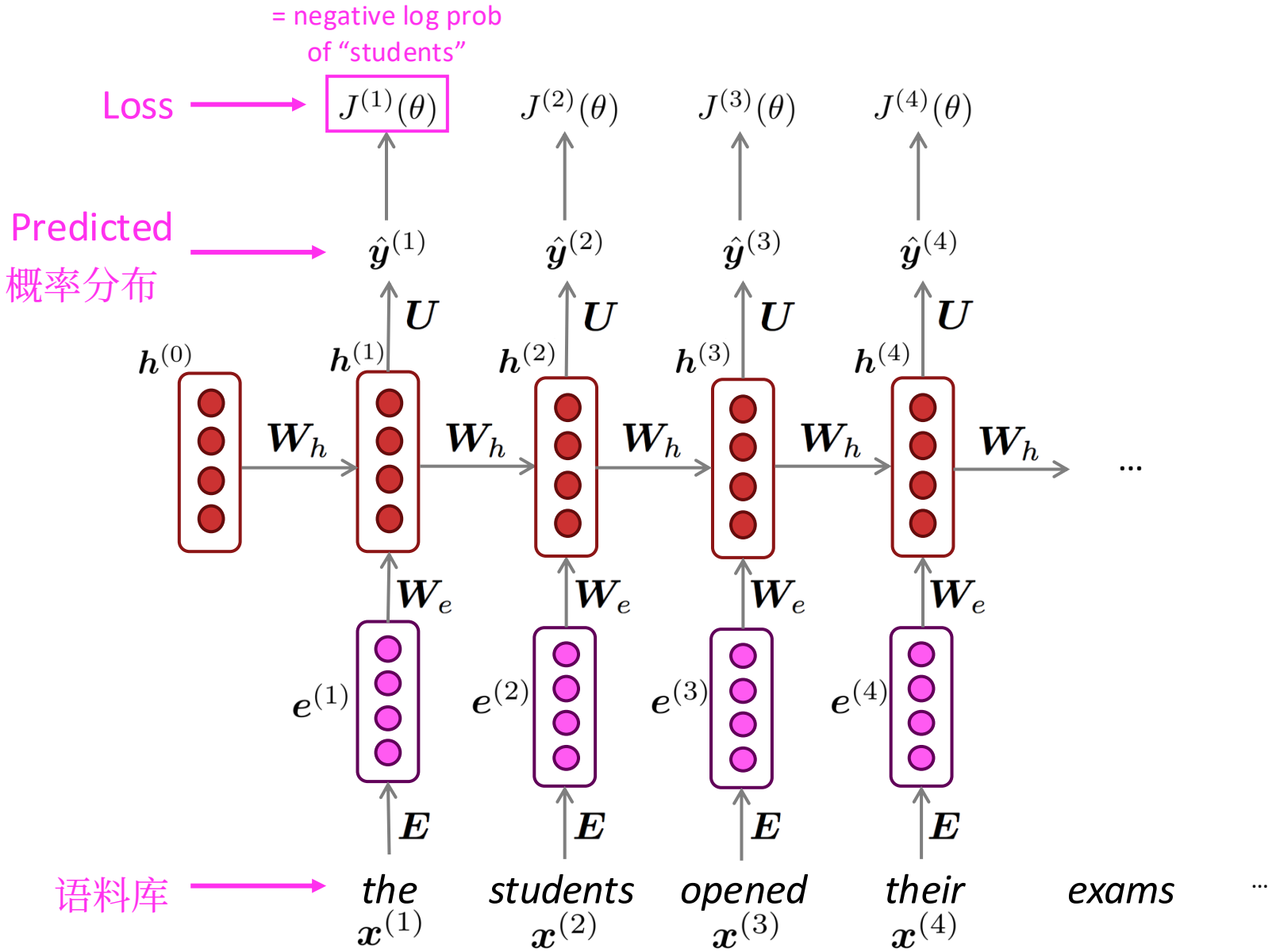
- Loss function on step *t* is 交叉熵 between predicted probability 分布，以及真实的下一个词（one-hot 表示）：

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

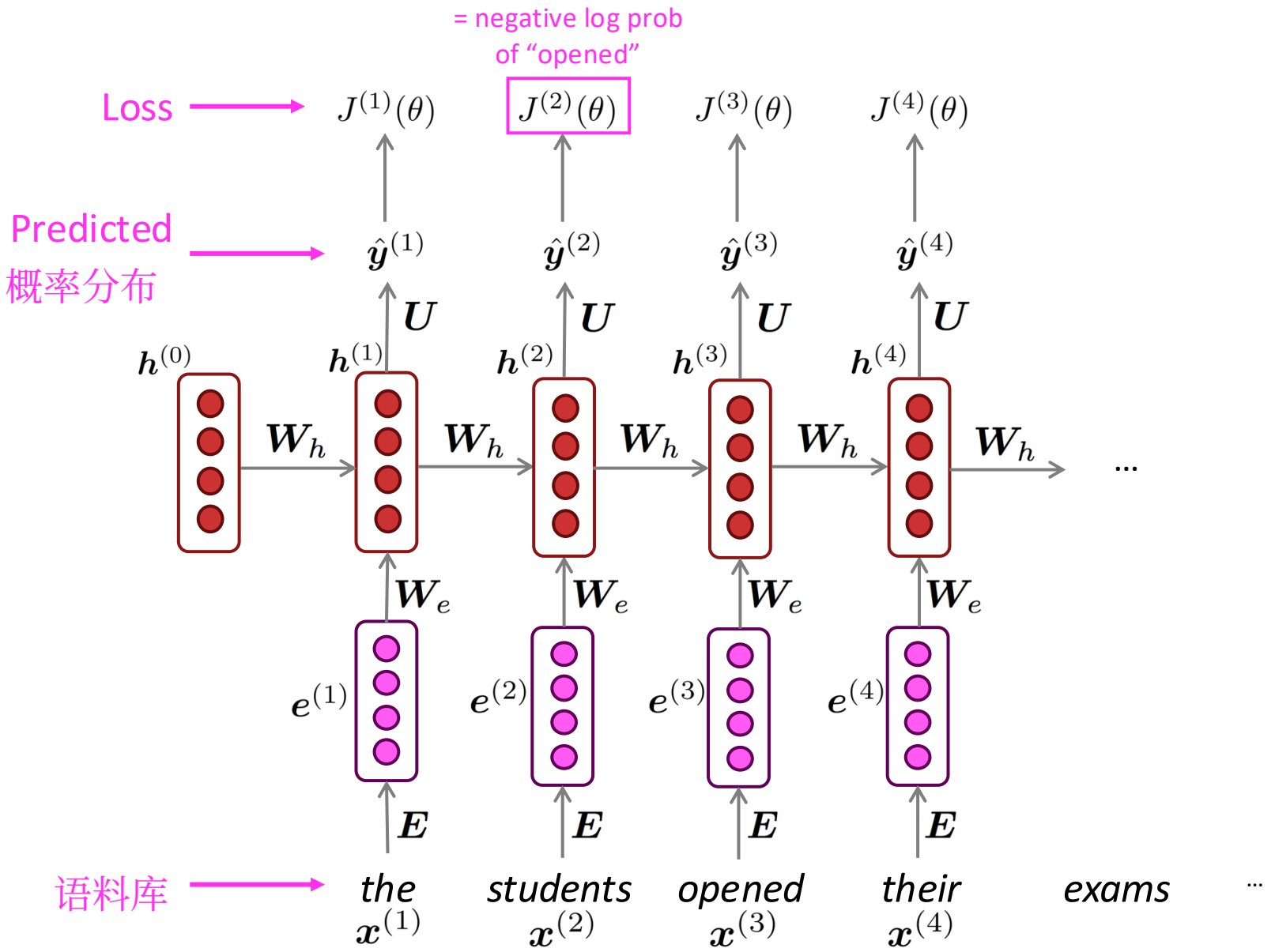
- Average this to get overall loss 对整个训练集：

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

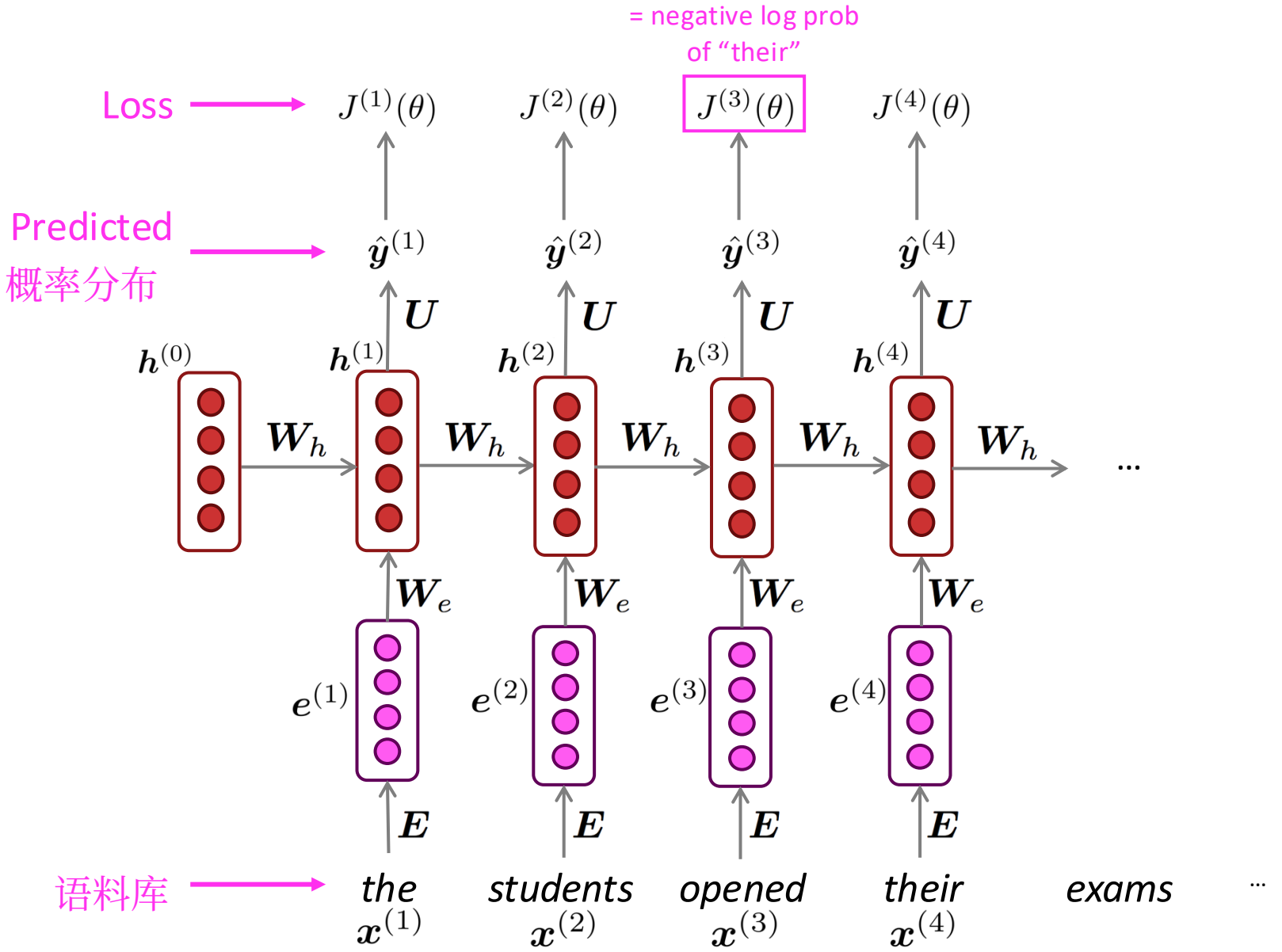
训练 RNN 语言模型


= negative log prob of "students"
Loss
$J^{(1)}(\theta)$
$J^{(2)}(\theta)$
$J^{(3)}(\theta)$
$J^{(4)}(\theta)$
Predicted
概率分布
$\hat{y}^{(1)}$
$\hat{y}^{(2)}$
$\hat{y}^{(3)}$
$\hat{y}^{(4)}$
U
$h^{(0)}$
$h^{(1)}$
$h^{(2)}$
$h^{(3)}$
$h^{(4)}$
W_h
...
W_e
$e^{(1)}$
$e^{(2)}$
$e^{(3)}$
$e^{(4)}$
E
语料库
the
$x^{(1)}$
students
$x^{(2)}$
opened
$x^{(3)}$
their
$x^{(4)}$
exams
...

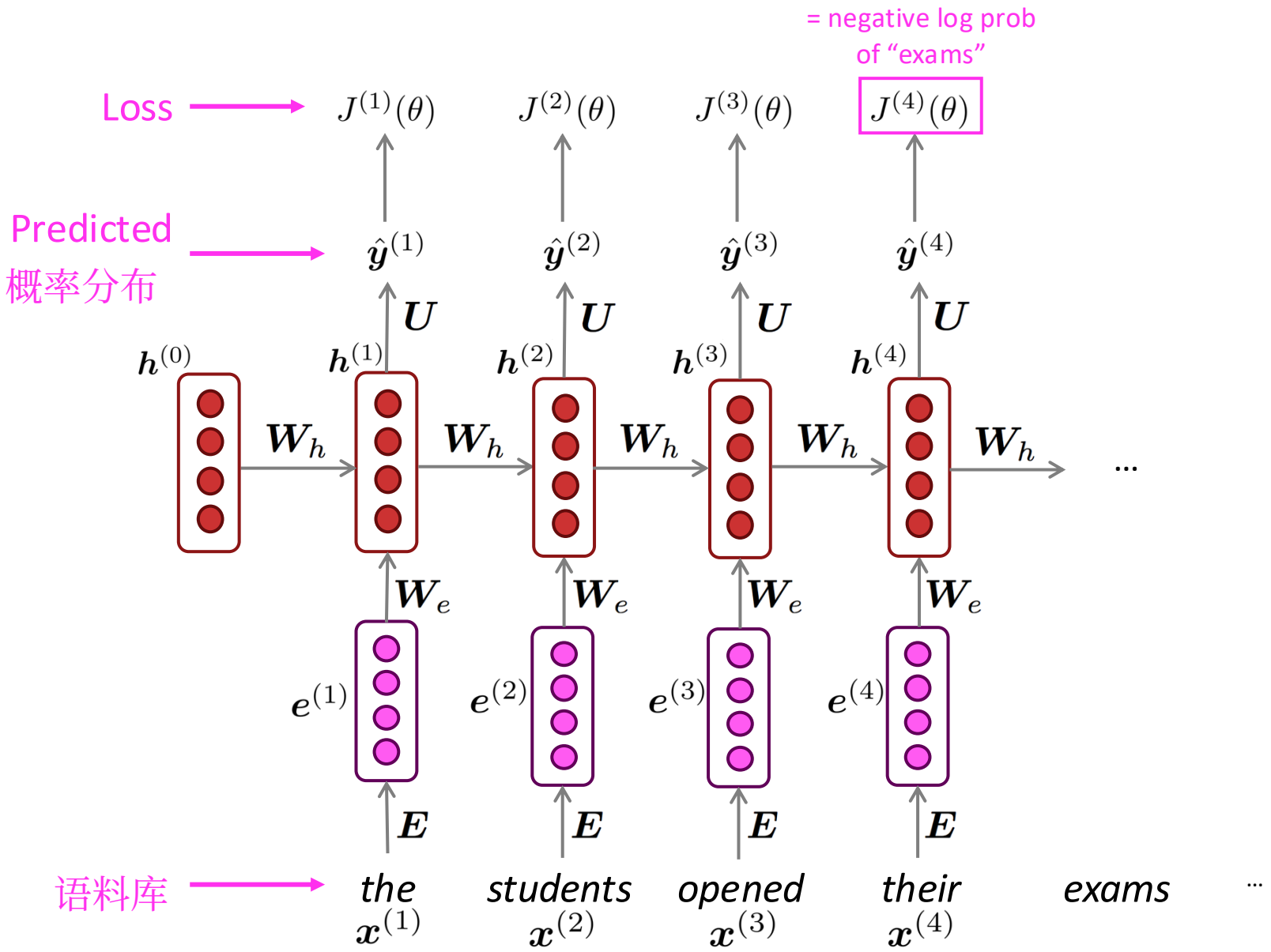
训练 RNN 语言模型


= negative log prob of "opened"
Loss
Predicted
概率分布
语料库
$J^{(1)}(\theta)$
$J^{(2)}(\theta)$
$J^{(3)}(\theta)$
$J^{(4)}(\theta)$
$\hat{y}^{(1)}$
$\hat{y}^{(2)}$
$\hat{y}^{(3)}$
$\hat{y}^{(4)}$
$h^{(0)}$
$h^{(1)}$
$h^{(2)}$
$h^{(3)}$
$h^{(4)}$
U
W_h
W_e
E
$e^{(1)}$
$e^{(2)}$
$e^{(3)}$
$e^{(4)}$
the
$x^{(1)}$
students
$x^{(2)}$
opened
$x^{(3)}$
their
$x^{(4)}$
exams
...

训练 RNN 语言模型

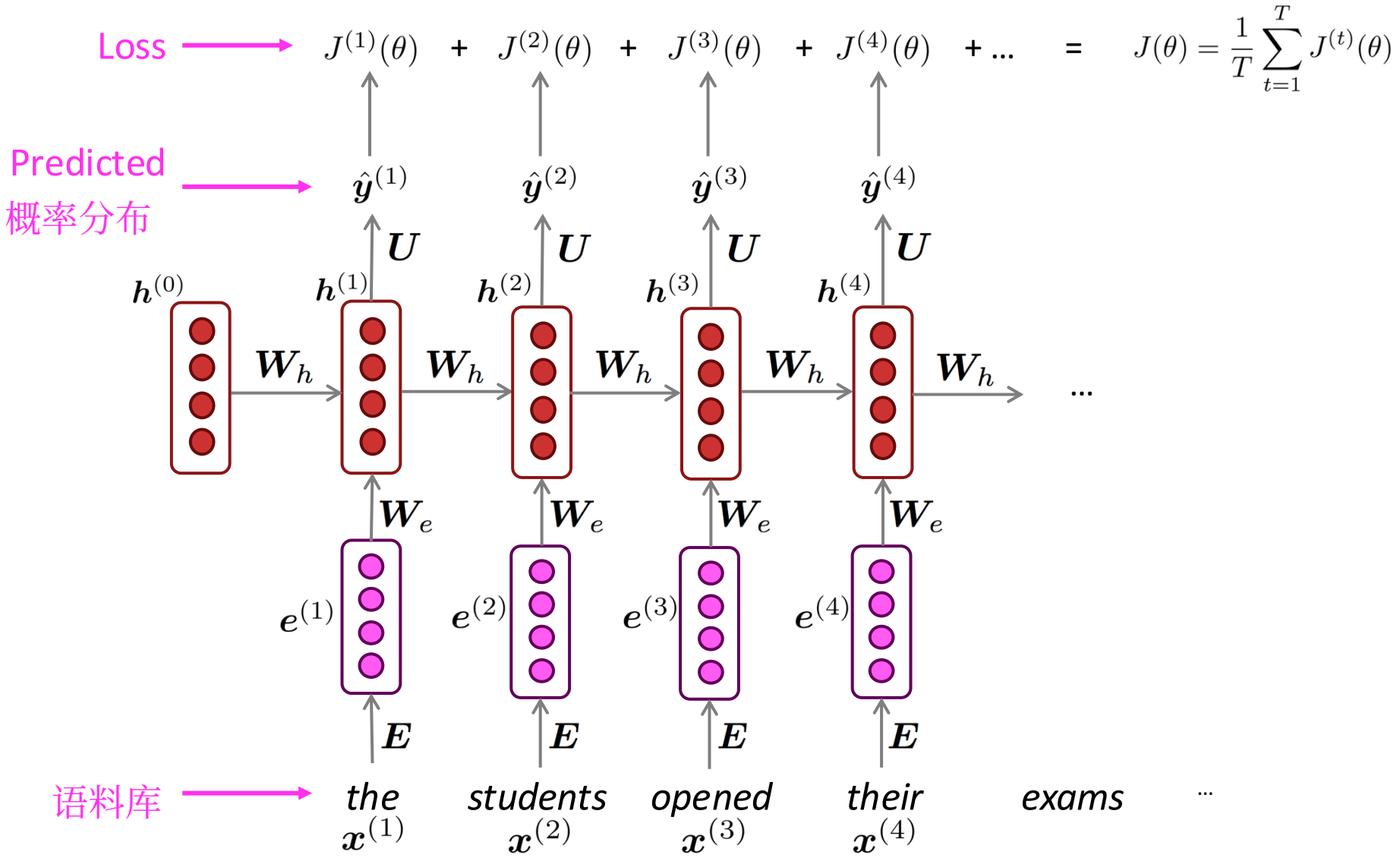

= negative log prob of "their"
Loss
Predicted
概率分布
语料库
$J^{(1)}(\theta)$
$J^{(2)}(\theta)$
$J^{(3)}(\theta)$
$J^{(4)}(\theta)$
$\hat{y}^{(1)}$
$\hat{y}^{(2)}$
$\hat{y}^{(3)}$
$\hat{y}^{(4)}$
$h^{(0)}$
$h^{(1)}$
$h^{(2)}$
$h^{(3)}$
$h^{(4)}$
U
W_h
W_e
E
$e^{(1)}$
$e^{(2)}$
$e^{(3)}$
$e^{(4)}$
the
$x^{(1)}$
students
$x^{(2)}$
opened
$x^{(3)}$
their
$x^{(4)}$
exams
...

训练 RNN 语言模型


= negative log prob of "exams"
Loss
Predicted
概率分布
语料库
$J^{(1)}(\theta)$
$J^{(2)}(\theta)$
$J^{(3)}(\theta)$
$J^{(4)}(\theta)$
$\hat{y}^{(1)}$
$\hat{y}^{(2)}$
$\hat{y}^{(3)}$
$\hat{y}^{(4)}$
$h^{(0)}$
$h^{(1)}$
$h^{(2)}$
$h^{(3)}$
$h^{(4)}$
U
W_h
W_e
E
$e^{(1)}$
$e^{(2)}$
$e^{(3)}$
$e^{(4)}$
the
$x^{(1)}$
students
$x^{(2)}$
opened
$x^{(3)}$
their
$x^{(4)}$
exams
...

训练 RNN 语言模型

"Teacher forcing"



训练 RNN 语言模型

- However: Computing loss and gradients across entire corpus $\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(T)}$ at once is too expensive （内存方面）！

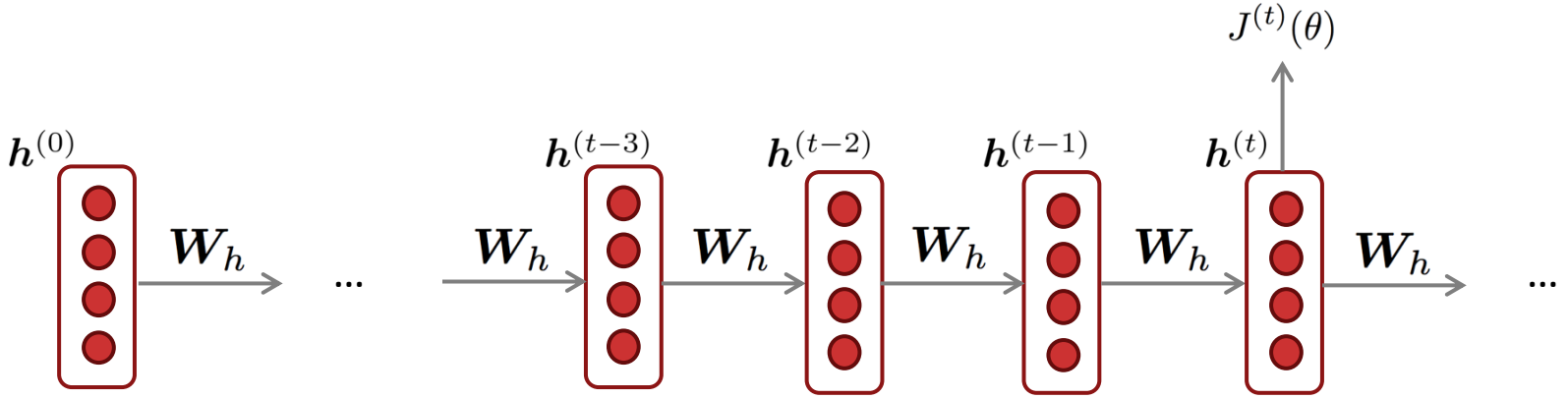
$$J(\theta) = \frac{1}{T} \sum_{t=1}^{T} J^{(t)}(\theta)$$

- In practice, consider $\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(T)}$ as a 句子 (or a document)

- Recall: Stochastic Gradient Descent allows us to compute loss and gradients for small 数据块，然后更新。

- Compute loss $J(\theta)$ for a sentence (actually, a batch of sentences), compute gradients 并更新权重。在新的一批句子上重复。

RNN 的反向传播



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

回答：$$\frac{\partial J^{(t)}}{\partial \boldsymbol{W}_h} = \sum_{i=1}^{t} \frac{\partial J^{(t)}}{\partial \boldsymbol{W}_h}\bigg|_{(i)}$$

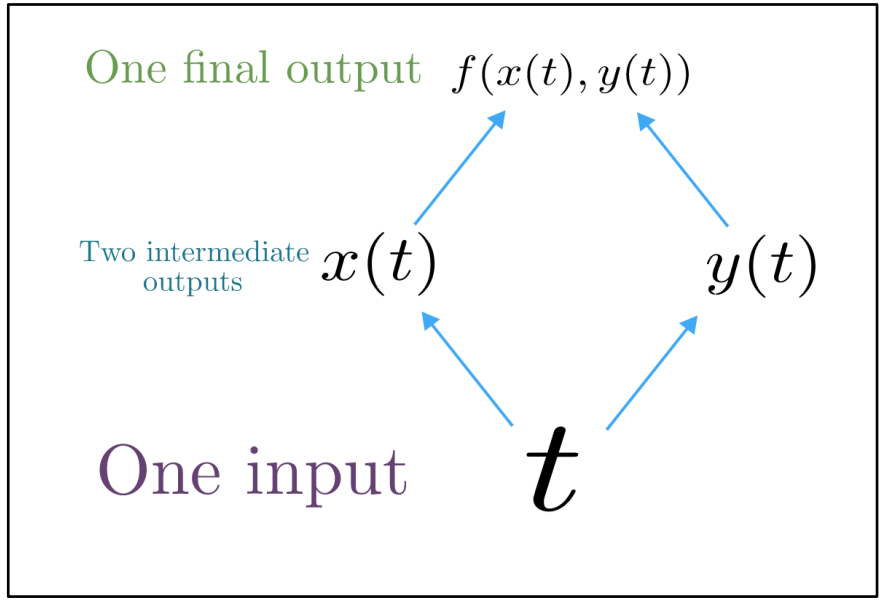
"The gradient w.r.t. a repeated weight is the sum of the gradient w.r.t. each time it appears"

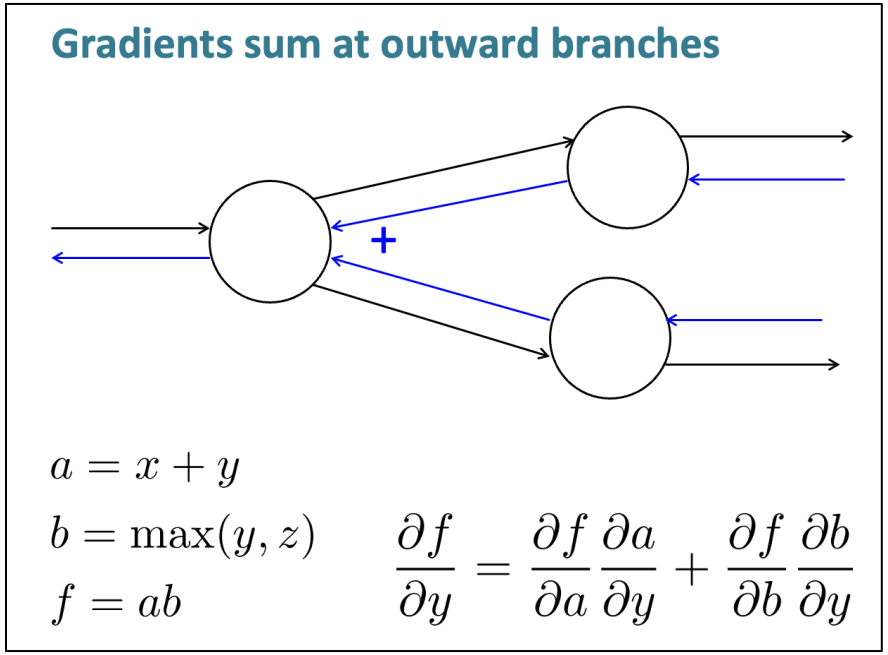
为什么？

多元链式法则

- 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:

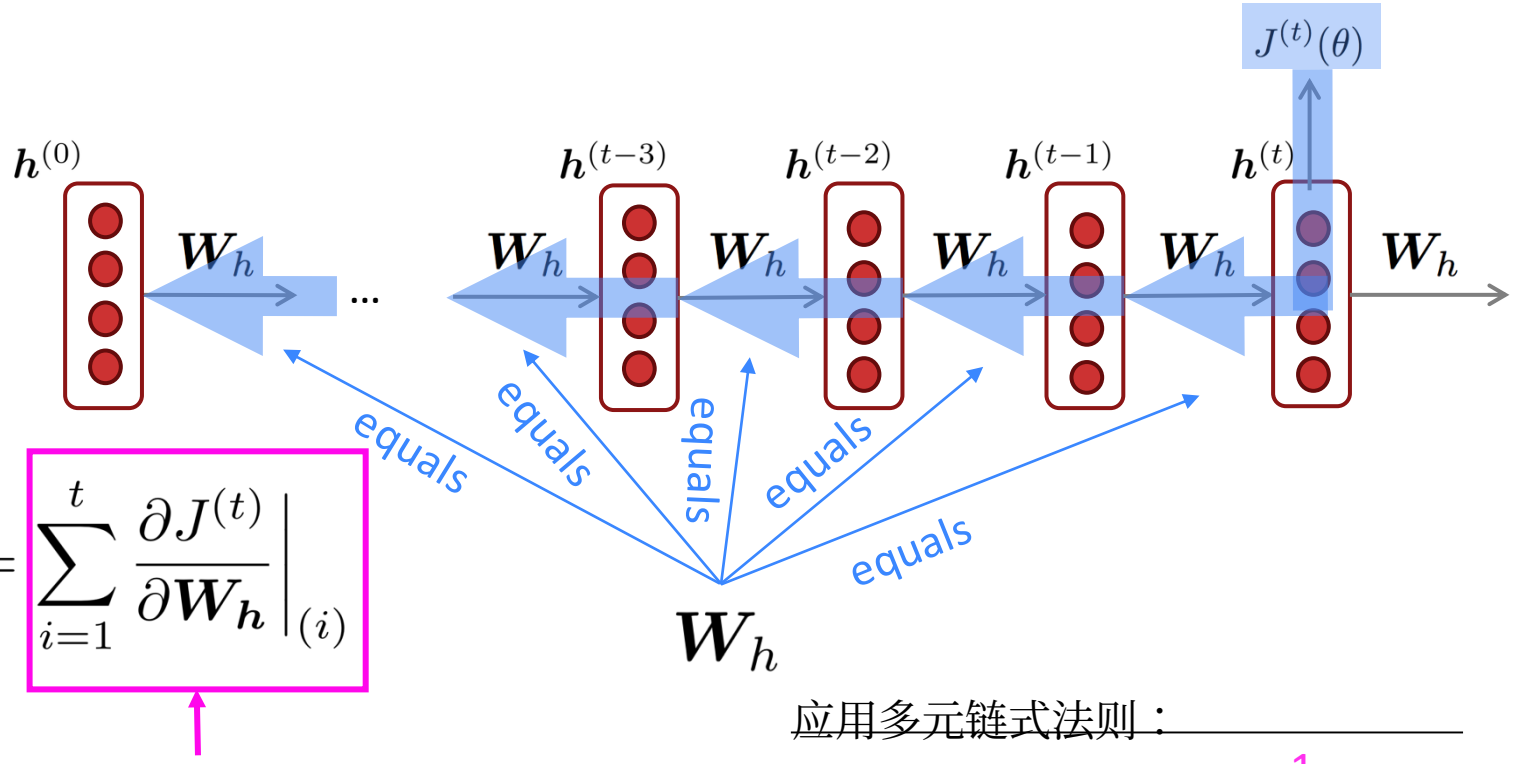
$$\underbrace{\frac{d}{dt} f(x(t), y(t))}_{\text{Derivative of composition function}} = \frac{\partial f}{\partial x}\frac{dx}{dt} + \frac{\partial f}{\partial y}\frac{dy}{dt}$$





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

训练 RNN 的参数：RNN 的反向传播



In practice, often "truncated" after ~20 timesteps for training

效率原因

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

问题： How do we calculate this?

回答： Backpropagate over timesteps $i = t, \cdots, 0$，沿途累加梯度。

This algorithm is called **"backpropagation through time"** [Werbos, P.G., 1988, *Neural Networks* **1**, 等人]

应用多元链式法则：

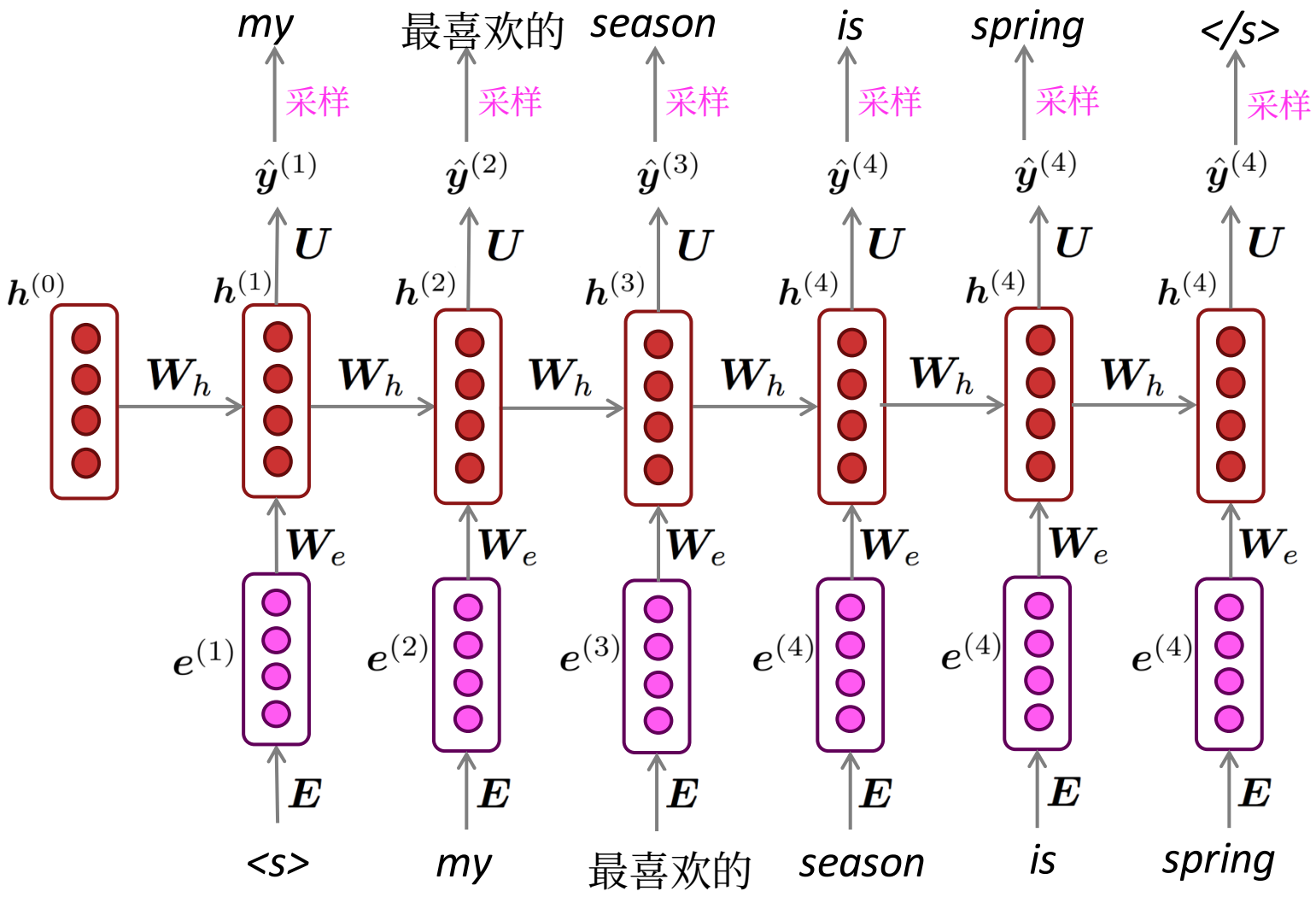
$$\frac{\partial J^{(t)}}{\partial \boldsymbol{W}_h} = \sum_{i=1}^{t} \frac{\partial J^{(t)}}{\partial \boldsymbol{W}_h}\bigg|_{(i)} \frac{\partial \boldsymbol{W}_h|_{(i)}}{\partial \boldsymbol{W}_h}$$

= 1

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

Generating with an RNN Language Model ("Generating roll outs")

Just like an n-gram Language Model, you can use a RNN Language Model to generate text by 重复采样. Sampled output becomes next step's input.



使用　ＲＮＮ　语言模型生成文本

Let’s have some fun!

- 你可以在任何类型的文本上训练　ＲＮＮ－ＬＭ，然后以该风格生成文本。
- RNN-LM trained on *Harry Potter*:

“Sorry,” Harry shouted, panicking—“I’ll leave those brooms in London, are they?”

“No idea,” said Nearly Headless Nick, casting low close by Cedric, carrying the last bit of treacle Charms, from Harry’s shoulder, and to answer him the common room perched upon it, four arms held a shining knob from when the spider hadn’t felt it seemed. He reached the teams too.

来源：https://medium.com/deep-writing/harry-potter-written-by-artificial-intelligence-8a9431803da6

使用 RNN 语言模型生成文本

Let's have some fun!

- 你可以在任何类型的文本上训练 RNN-LM，然后以该风格生成文本。
- RNN-LM trained on 菜谱 ：

```
      Title: CHOCOLATE RANCH BARBECUE
Categories: Game, Casseroles, Cookies, Cookies
      Yield: 6 Servings

      2 tb Parmesan cheese -- chopped
      1 c  Coconut milk
      3    Eggs, beaten

Place each pasta over layers of lumps. Shape mixture into the moderate oven and simmer
until firm. Serve hot in bodied fresh, mustard, orange and cheese.

Combine the cheese and salt together the dough in a large skillet; add the ingredients
and stir in the chocolate and pepper.
```

来源： https://gist.github.com/nylki/1efbaa36635956d35bcc

评估语言模型

- The standard 评估指标 for Language Models is 困惑度 .

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

Normalized by 词的数量

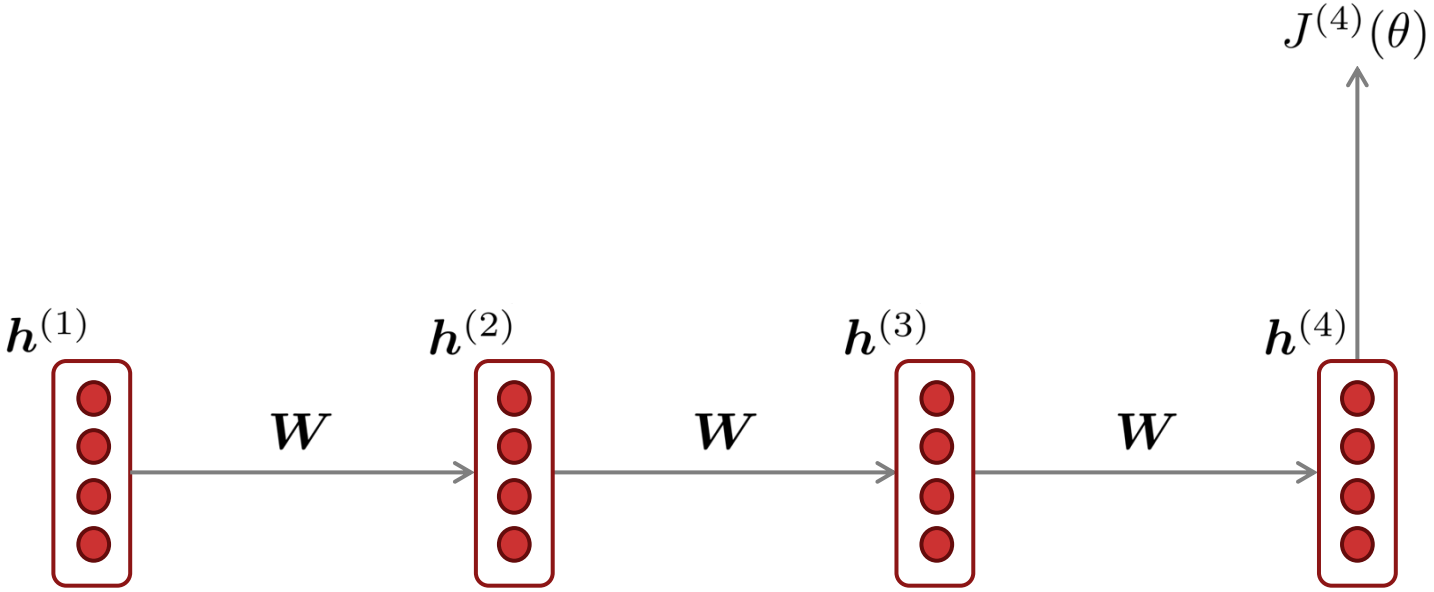
根据语言模型的语料库的逆概率

- 这等于交叉熵损失的指数：

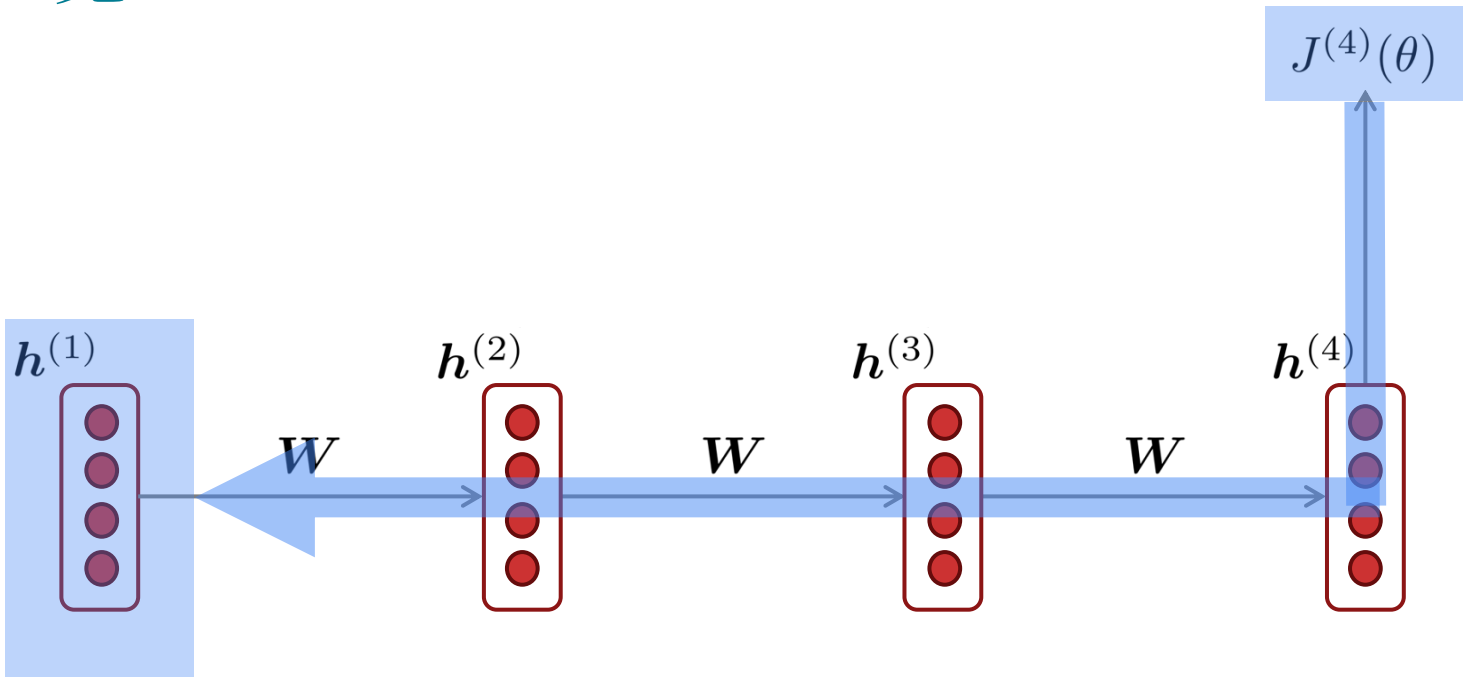
$$= \prod_{t=1}^{T} \left(\frac{1}{\hat{\boldsymbol{y}}^{(t)}_{\boldsymbol{x}_{t+1}}} \right)^{1/T} = \exp\left(\frac{1}{T} \sum_{t=1}^{T} -\log \hat{\boldsymbol{y}}^{(t)}_{\boldsymbol{x}_{t+1}} \right) = \exp(J(\theta))$$

较低的perplexity is better!

3． RNN 的问题：梯度消失和梯度爆炸

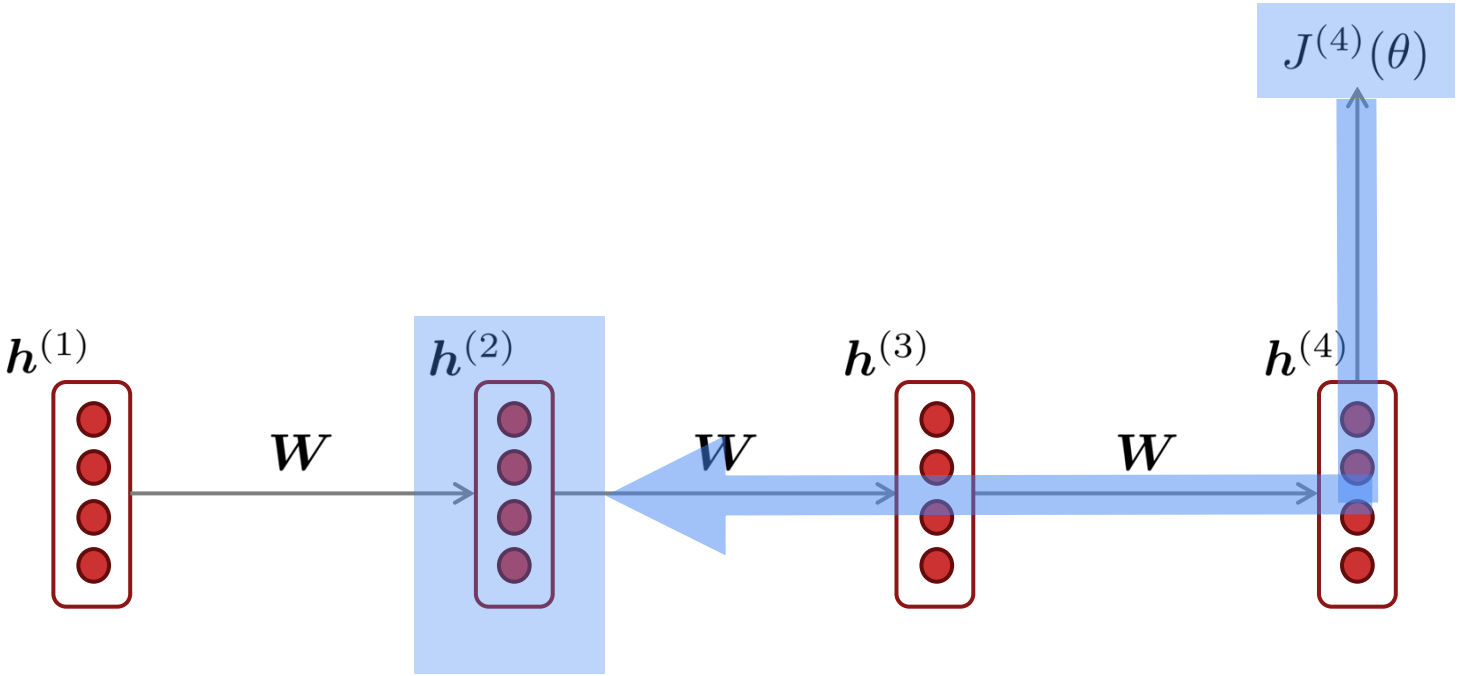


梯度消失的直觉



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

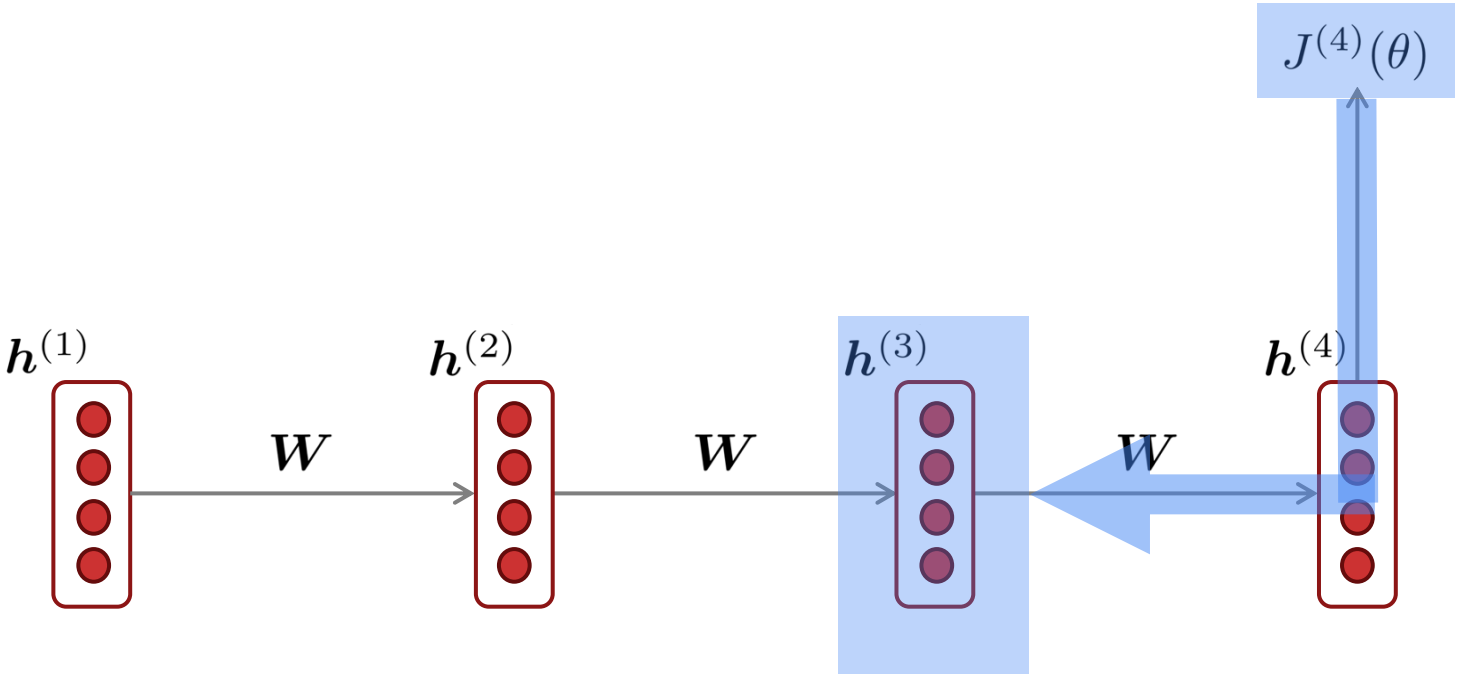
梯度消失的直觉



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

链式法则！

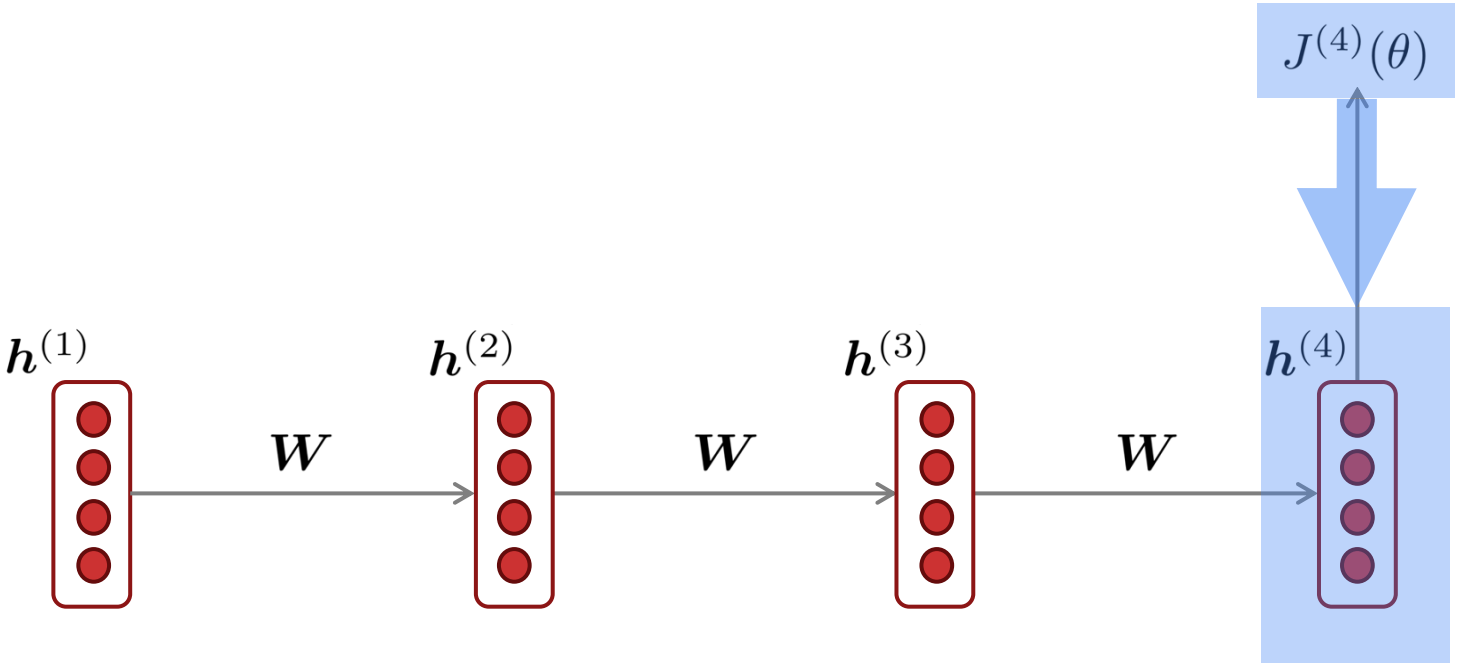
梯度消失的直觉



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

链式法则！

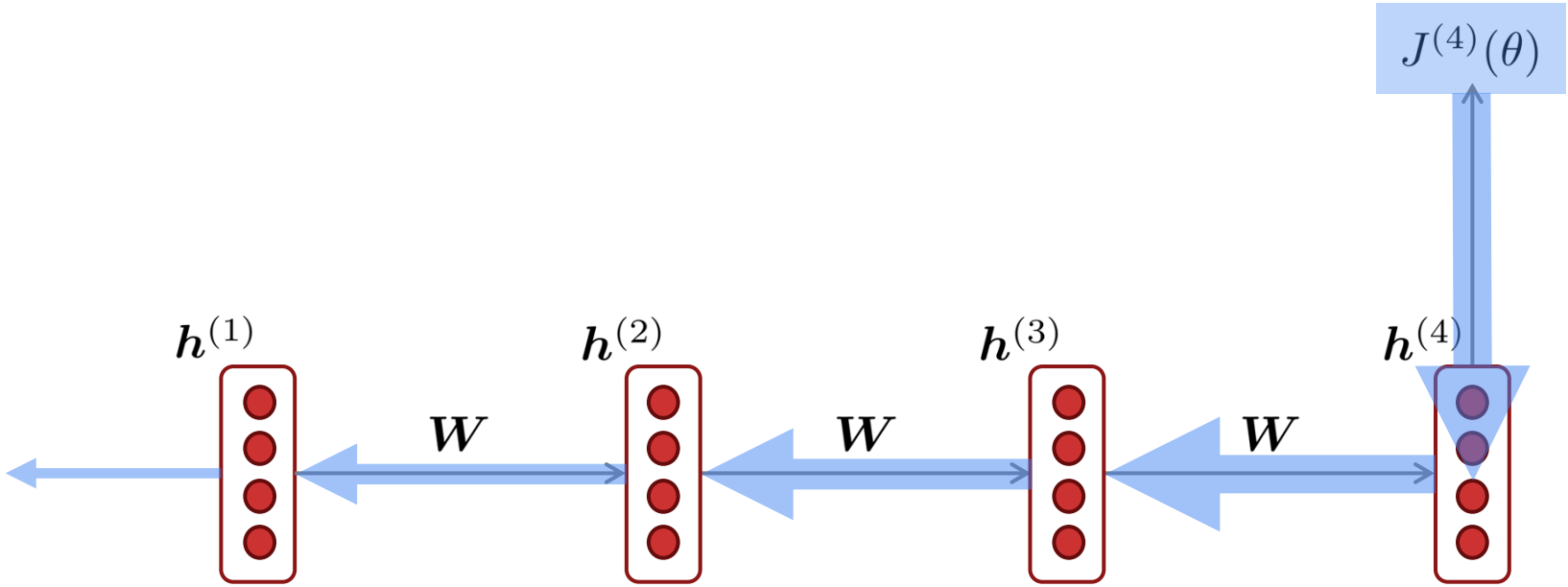
梯度消失的直觉



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

链式法则！

梯度消失的直觉



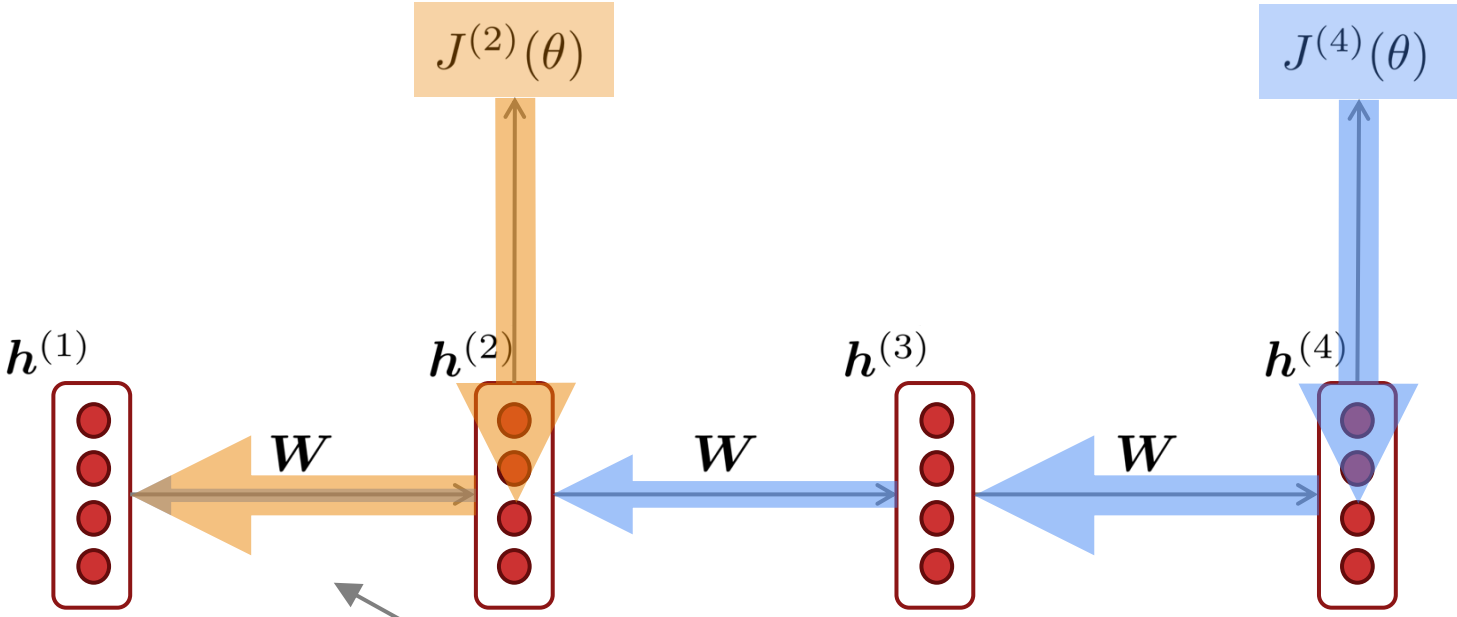
$$\frac{\partial J^{(4)}}{\partial h^{(1)}} = \frac{\partial h^{(2)}}{\partial h^{(1)}} \times \frac{\partial h^{(3)}}{\partial h^{(2)}} \times \frac{\partial h^{(4)}}{\partial h^{(3)}} \times \frac{\partial J^{(4)}}{\partial h^{(4)}}$$

如果这些很小会怎样？

Vanishing gradient problem:
When these are small, the gradient signal gets smaller and smaller as it 进一步反向传播

Source: "On the difficulty of training recurrent neural networks", Pascanu et al, 2013. http://proceedings.mlr.press/v28/pascanu13.pdf (and supplemental materials), at http://proceedings.mlr.press/v28/pascanu13-supp.pdf

为什么梯度消失是一个问题？



Gradient signal from far away is lost because it's much smaller than 来自近处的梯度信号

So, model weights are updated only with respect to 近处效果 , not 长期影响 .

梯度消失对 RNN-LM 的影响

- 语言模型任务：*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*

  将墨粉装入打印机后，她终于打印了她的 _ _ _ _ _ _ _ _

- To learn from this training example, the RNN-LM needs to model the dependency between *"tickets"* on the 7th step and the target word *"tickets"* 在最后。

- But if the gradient is small, the model can't learn this dependency
  - So, the model is unable to predict similar long-distance dependencies 在测试时

为什么梯度爆炸是一个问题？

- 如果梯度变得太大，那么 ＳＧＤ 更新步骤就变得太大：

$$\theta^{new} = \theta^{old} - \overbrace{\alpha}^{\text{学习率}} \underbrace{\nabla_\theta J(\theta)}_{\text{gradient}}$$

- This can cause 错误的更新 : we take too large a step and reach a weird and bad 参数配置（损失很大）
  - You think you've found a hill to climb, but suddenly you're in Iowa

- In the worst case, this will result in Inf or NaN in your network （然后你必须从更早的 ｃｈｅｃｋｐｏｉｎｔ 重新开始训练）

梯度裁剪：梯度爆炸的解决方案

- 梯度裁剪 : if the norm of the gradient is greater than some threshold, scale it down before applying SGD update

Algorithm 1 Pseudo-code for norm clipping

$\hat{\mathbf{g}} \leftarrow \frac{\partial \mathcal{E}}{\partial \theta}$
if $\|\hat{\mathbf{g}}\| \geq threshold$ **then**
  $\hat{\mathbf{g}} \leftarrow \frac{threshold}{\|\hat{\mathbf{g}}\|}\hat{\mathbf{g}}$
end if

- 直觉 ：沿同一方向走一步，但步幅更小

- In practice, 记住裁剪梯度很重要 , but exploding gradients are an 容易解决的问题

Source: “On the difficulty of training recurrent neural networks”, Pascanu et al, 2013. http://proceedings.mlr.press/v28/pascanu13.pdf

如何解决梯度消失问题？

- The main problem is that *it's too difficult for the RNN to learn to preserve information* 跨很多时间步 .

- In a vanilla RNN, the hidden state is constantly being 被重写

$$\boldsymbol{h}^{(t)} = \sigma\left(\boldsymbol{W}_h \boldsymbol{h}^{(t-1)} + \boldsymbol{W}_x \boldsymbol{x}^{(t)} + \boldsymbol{b}\right)$$

- First: How about an RNN with separate memory 被加到了哪里？
  - 长短期记忆（LSTM）[link]

- 然后：在模型中创建更直接和线性的直通连接
  - Attention、residual connections 等

5． 机器翻译

Machine Translation (MT) is the task of translating a sentence *x* from one language (the 源语言) to a sentence *y* in another language (the 目标语言).

x: *I like deep learning*

y: *我喜欢深度学习*

NMT：NLP Deep Learning 的第一个重大成功案例

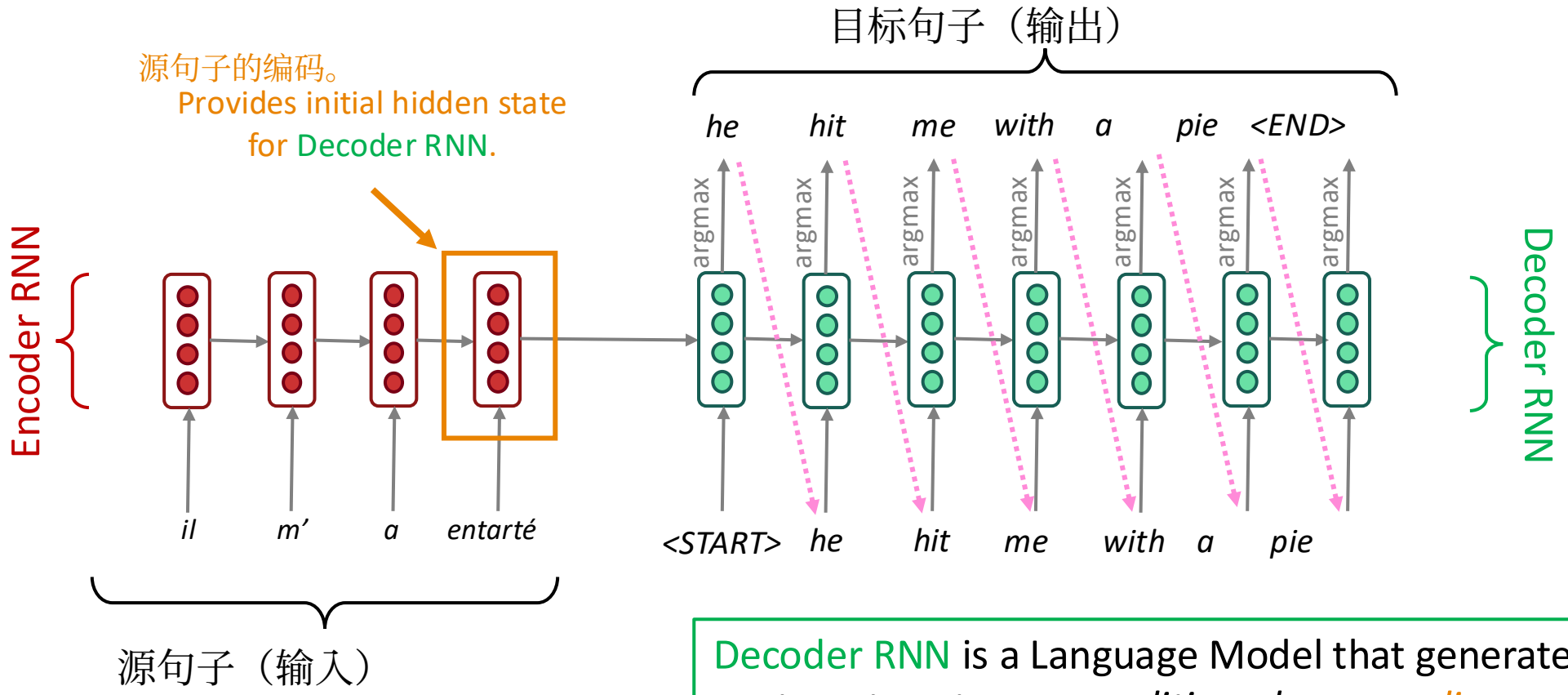
Neural Machine Translation went from a fringe research attempt in **2014** to the leading 标准方法 in **2016**

- **2014**: First seq2seq paper published [Sutskever et al. 2014]
- **2016**：Google 翻译从 SMT 切换到 NMT — 到 2018 年所有人都

- 这太棒了！
  - **SMT** systems, built by 数百 of engineers over many years, outperformed by NMT systems trained by 小组 of engineers in a few months

神经机器翻译（NMT）

Sequence-to-sequence 模型



Encoder RNN produces an encoding of the 源句子。

Decoder RNN is a Language Model that generates target sentence, *conditioned on encoding*.

Note: This diagram shows 测试时 behavior: decoder output is fed in ⋯→ as next step's input

Sequence-to-sequence 用途广泛！

- The general notion here is an encoder-decoder model
  - 一个神经网络接收输入并产生一个神经表示
  - 另一个网络基于该神经表示产生输出
  - 如果输入和输出都是序列，我们称之为 seq2seq 模型

- Sequence-to-sequence is useful for 不仅仅是机器翻译
- 许多 NLP 任务可以表述为 sequence-to-sequence：
  - 摘要 (long text → short text)
  - 对话 (previous utterances → next utterance)
  - Parsing (input text → output parse as sequence)
  - Code generation （自然语言 → Python 代码）

神经机器翻译（NMT）

- The 序列到序列 model is an example of a 条件语言模型
  - **Language Model** because the decoder is predicting the next word of the target sentence *y*
  - **Conditional** because its predictions are *also* conditioned on the source sentence *x*

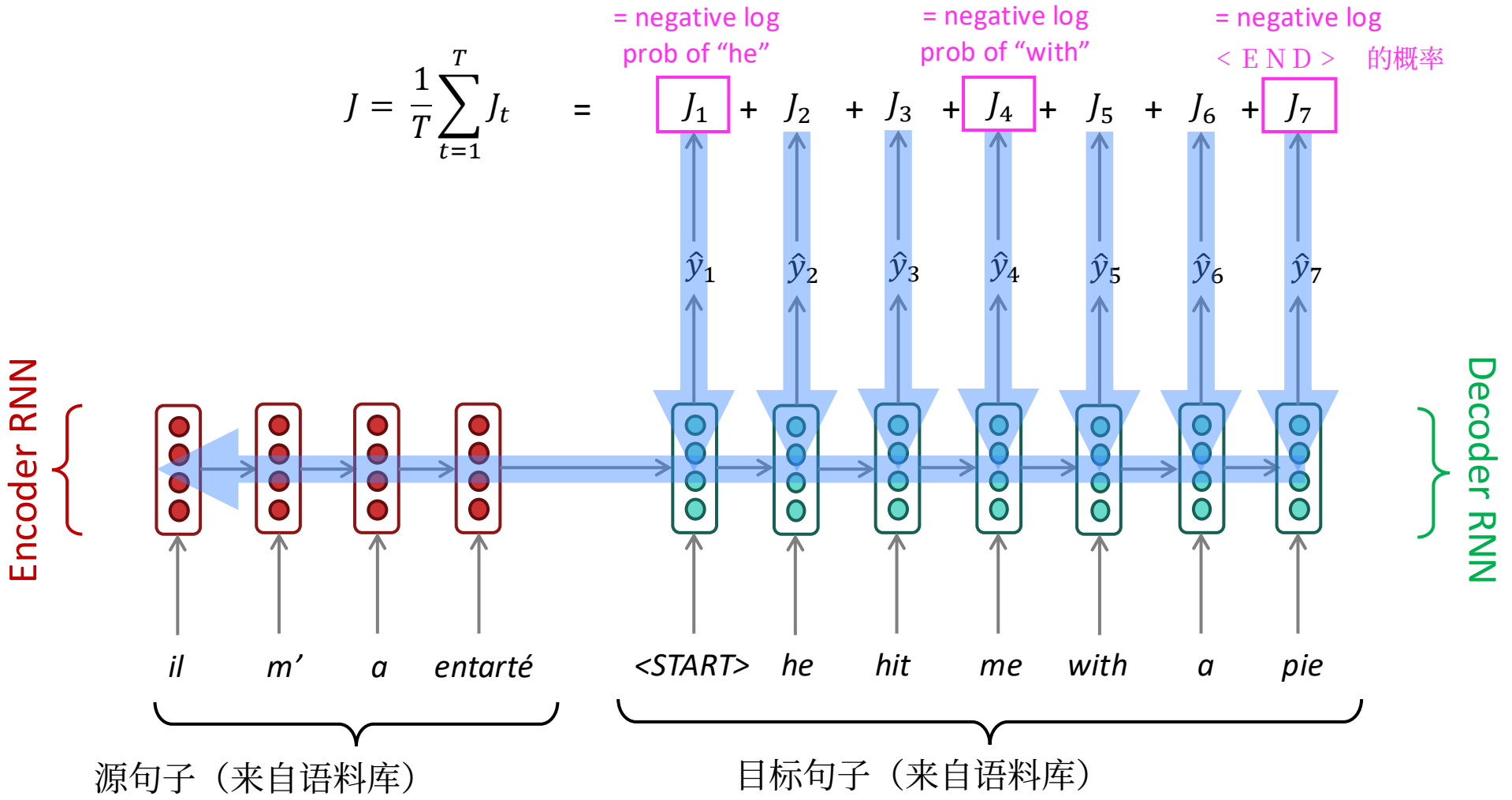
- NMT 直接计算：

$$P(y|x) = P(y_1|x)\,P(y_2|y_1,x)\,P(y_3|y_1,y_2,x)\ldots\underbrace{P(y_T|y_1,\ldots,y_{T-1},x)}$$

Probability of next target word, given target words so far and source sentence *x*

- **Question:** 如何训练一个 NMT 系统？
- **(Easy) Answer:** 获取一个大型平行语料库……
  - But there is now exciting work on “unsupervised NMT”, data augmentation, etc.

训练神经机器翻译系统

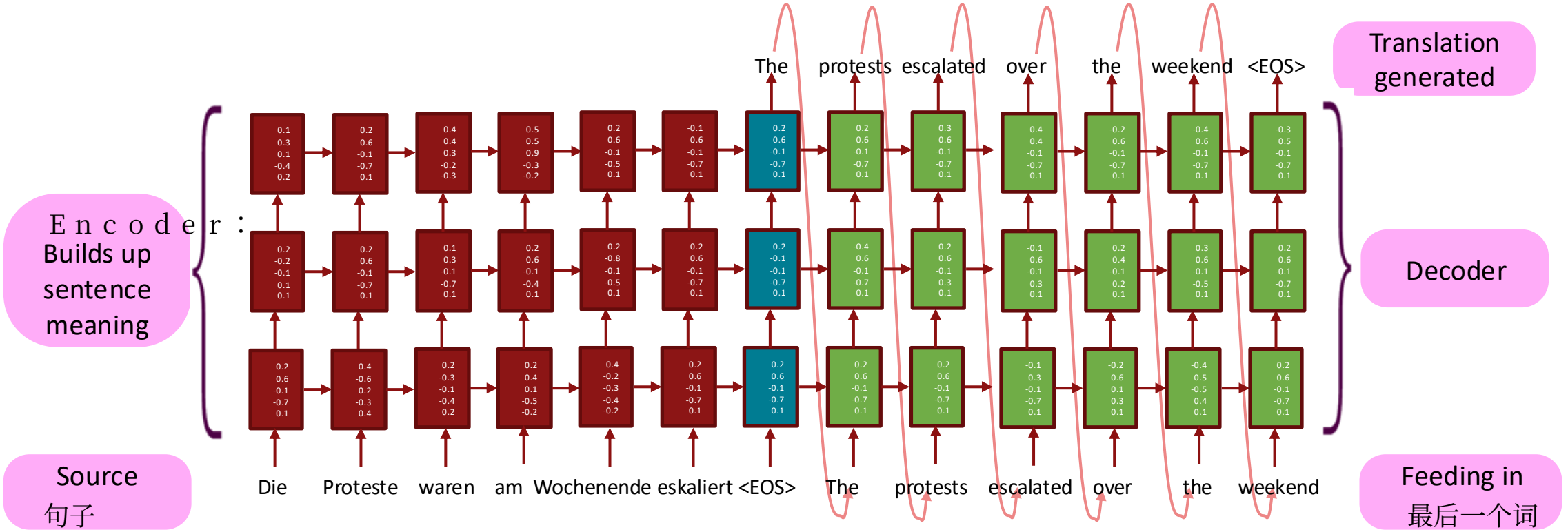


Seq2seq is optimized as a 单一系统。 Backpropagation operates “*end-to-end*”.

多层深度 encoder-decoder 机器翻译网络

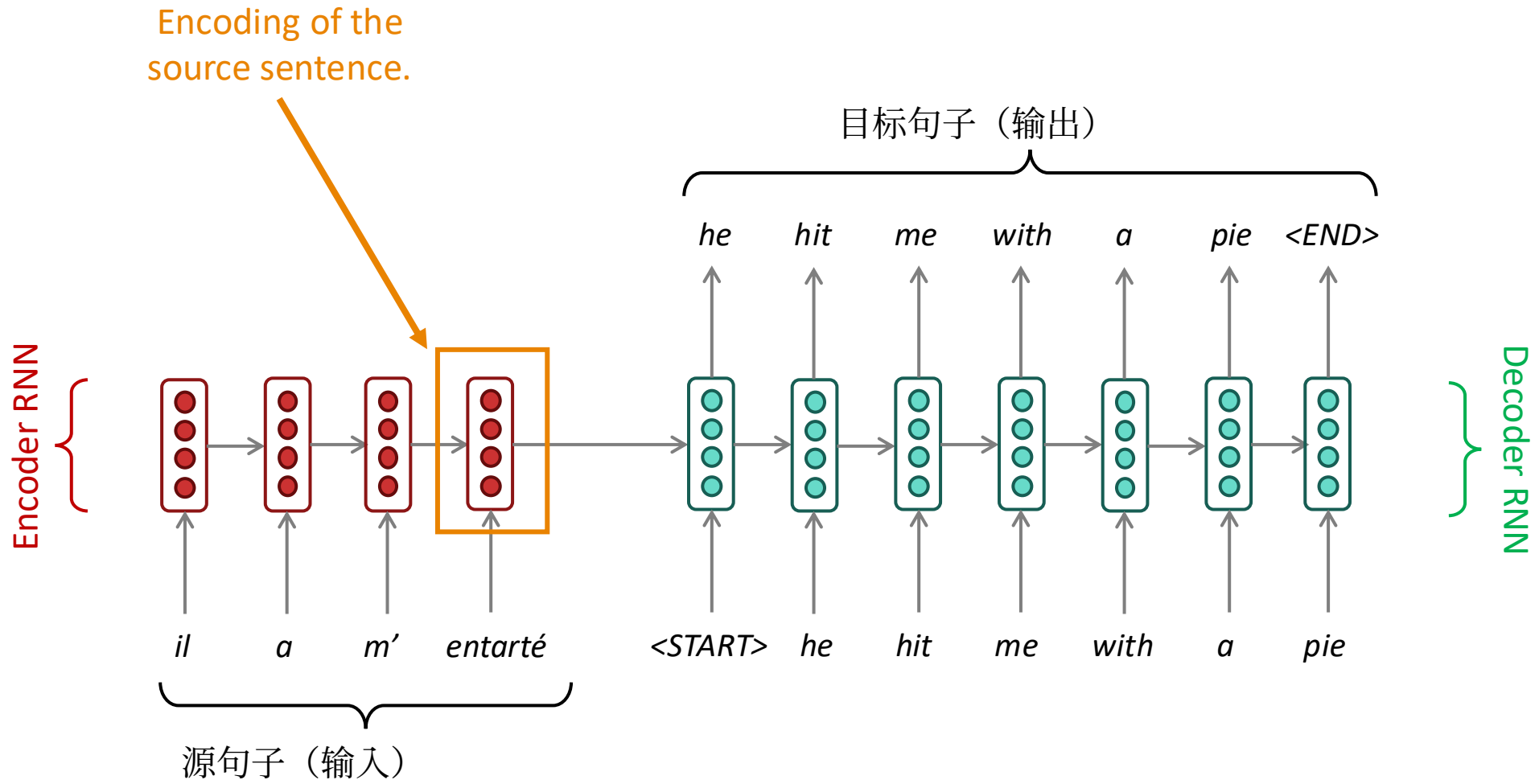
[Sutskever et al. 2014; Luong et al. 2015]

The hidden states from RNN layer *i* are the inputs to RNN layer *i*+1



条件 = Bottleneck

最后一部分：ＲＮＮ　中的瓶颈问题



这种架构的问题？

课程计划

Ｌｅｃｔｕｒｅ　４：语言建模　＋　ＲＮＮ

1. A new NLP task: **Language Modeling** （２０　分钟）

This is the most important concept in the 课程！引出了大部分现代　ＮＬＰ

↓ 推动了

2. Language models with neural nets: **Recurrent Neural Networks (RNNs)** （２５　分钟）
3. Problems with RNNs: exploding and vanishing gradients (20 mins)
4. 机器翻译（１０　分钟）