Natural Language Processing with Deep Learning

CS224N/Ling284

Lecture 6:
Language Models and
Recurrent Neural Networks

Abigail See

Overview

Today we will:

- Introduce a new NLP task
  - **Language Modeling**

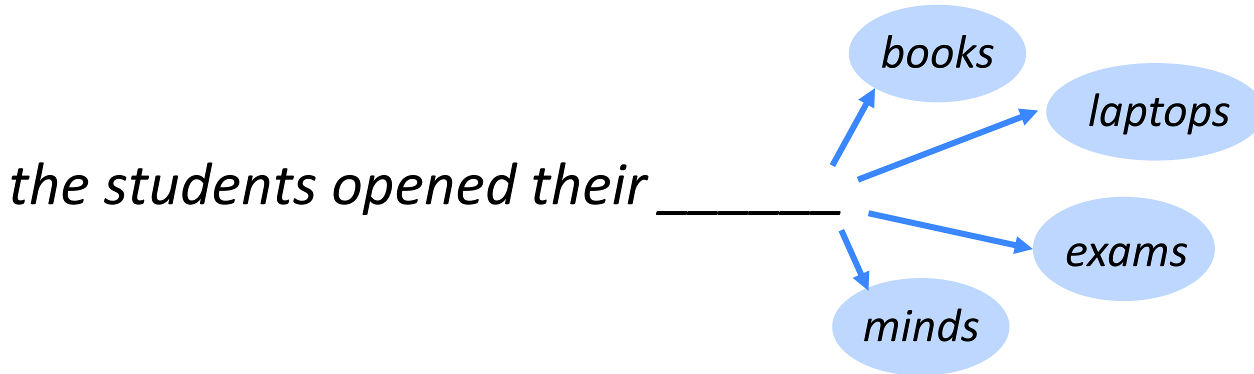
motivates

- Introduce a new family of neural networks
  - **Recurrent Neural Networks (RNNs)**

These are two of the most important ideas for the rest of the class!

Language Modeling

- **Language Modeling** is the task of predicting what word comes next.



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

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

where $\boldsymbol{x}^{(t+1)}$ can be any word in the vocabulary $V = \{\boldsymbol{w}_1, ..., \boldsymbol{w}_{|V|}\}$

所以可以视作分类模型，因为总类数就是|V|

- A system that does this is called a **Language Model**.

Language Modeling

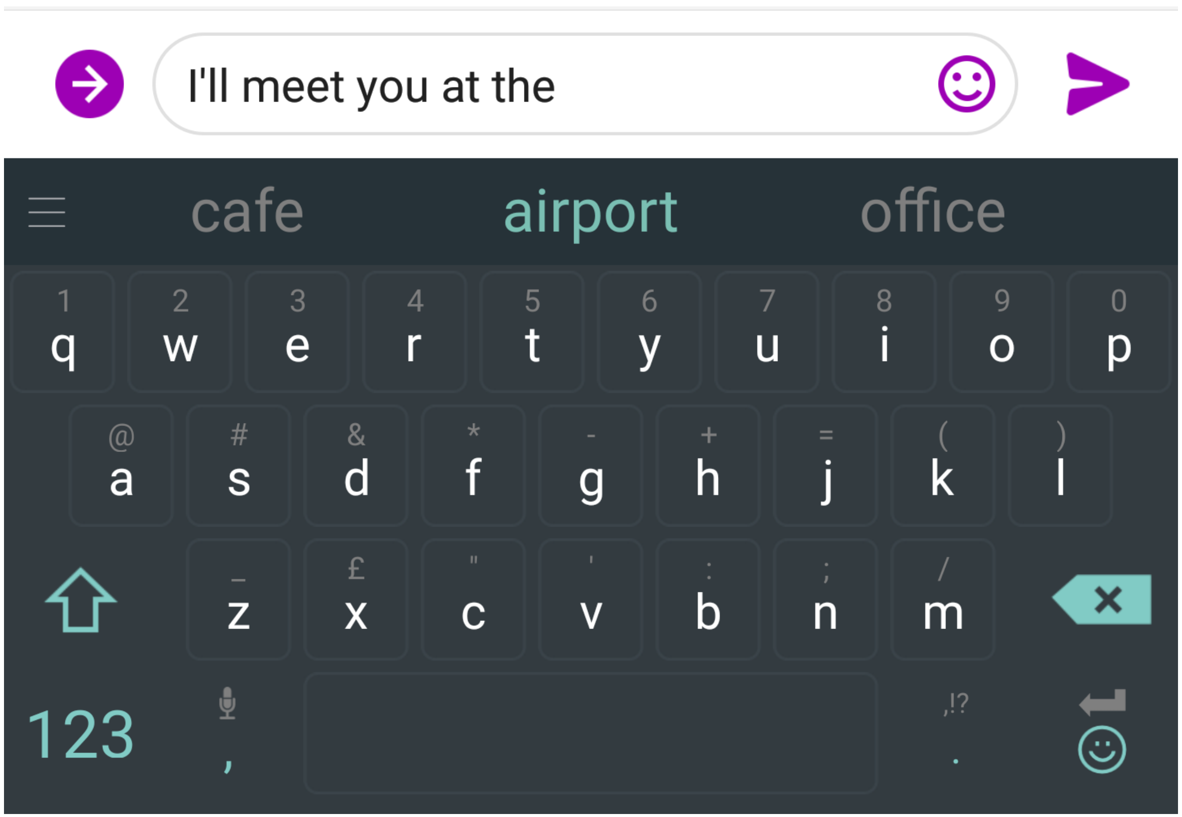
- You can also think of a Language Model as a system that assigns probability to a piece of text.

- For example, if we have some text $\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(T)}$, then the probability of this text (according to the Language Model) is:

$$
\begin{aligned}
P(\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(T)}) &= P(\boldsymbol{x}^{(1)}) \times P(\boldsymbol{x}^{(2)} | \, \boldsymbol{x}^{(1)}) \times \cdots \times P(\boldsymbol{x}^{(T)} | \, \boldsymbol{x}^{(T-1)}, \ldots, \boldsymbol{x}^{(1)}) \\
&= \prod_{t=1}^{T} \underbrace{P(\boldsymbol{x}^{(t)} | \, \boldsymbol{x}^{(t-1)}, \ldots, \boldsymbol{x}^{(1)})}_{\text{This is what our LM provides}}
\end{aligned}
$$

You use Language Models every day!

输入法、搜索框等



You use Language Models every day!

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

n-gram Language Models

the students opened their ______

- **Question**: How to learn a Language Model?
- **Answer** (pre- Deep Learning): learn a *n*-gram Language Model!

t-n+1

- Definition: A *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"
  - 4-grams: "the students opened their"

- Idea: Collect statistics about how frequent different n-grams are, and use these to predict next word.

n-gram Language Models

- First we make a simplifying assumption: $\boldsymbol{x}^{(t+1)}$ depends only on the preceding *n-1* words.

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

(assumption)

prob of a n-gram

prob of a (n-1)-gram

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

(definition of conditional prob)

- **Question:** How do we get these *n*-gram and (*n*-1)-gram probabilities?
- **Answer:** By counting 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 approximation)

n-gram Language Models: Example

Suppose we are learning a 4-gram Language Model.

~~as the proctor started the clock, the~~ $\underbrace{\textit{students opened their}}_{\text{condition on this}}$ ______

discard

condition on this

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

<u>For example, suppose that in the corpus:</u>

- "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?

监考人

Sparsity Problems with n-gram Language Models

很多词都不会是下一个词，分子为0

Sparsity Problem 1

Problem: What if *"students opened their $\boldsymbol{w}$"* never occurred in data? Then $\boldsymbol{w}$ has probability 0!

防止训练集无的数据全为0

(Partial) Solution: Add small δ to the count for every $\boldsymbol{w} \in V$. This is called *smoothing*.

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

分母为0【！】

Sparsity Problem 2

Problem: What if *"students opened their"* never occurred in data? Then we can't calculate probability for *any* $\boldsymbol{w}$!

“退化”成n-1 gram

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

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

Storage Problems with n-gram Language Models

<u>Storage</u>: Need to store count for all *n*-grams you saw in the corpus.

组合数: |V|^n

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

Increasing *n* or increasing corpus increases model size!

n-gram Language Models in practice

- You can build a simple trigram Language Model over a 1.7 million word corpus (Reuters) in a few seconds on your laptop*

Business and financial news

today the _______

get probability distribution

company	$\boxed{0.153}$
bank	$\boxed{0.153}$
price	0.077
italian	0.039
emirate	0.039
…	

Sparsity problem: not much granularity in the probability distribution **粒度, 跨度**

Otherwise, seems reasonable!

*** Try for yourself:** https://nlpforhackers.io/language-models/

Generating text with a n-gram Language Model

- You can also use a Language Model to generate text.

today the ________

condition on this

get probability distribution

company	0.153
bank	0.153
price	0.077
italian	0.039
emirate	0.039
…	

sample

Generating text with a n-gram Language Model

- You can also use a Language Model to generate text.

today the price ________

condition on this

get probability distribution

of	0.308	sample
for	0.050	
it	0.046	
to	0.046	
is	0.031	
…		

Generating text with a n-gram Language Model

- You can also use a Language Model to generate text.

today the $\underbrace{\textit{price of}}_{\text{condition on this}}$ ________

get probability distribution

the	0.072
18	0.043
oil	0.043
its	0.036
gold	0.018
…	

sample

Generating text with a n-gram Language Model

- You can also use a Language Model to generate text.

today the price of gold _______

Generating text with a n-gram Language Model

- You can also use a Language Model to generate text.

把标点符号也作为token kind

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 .

Surprisingly grammatical!

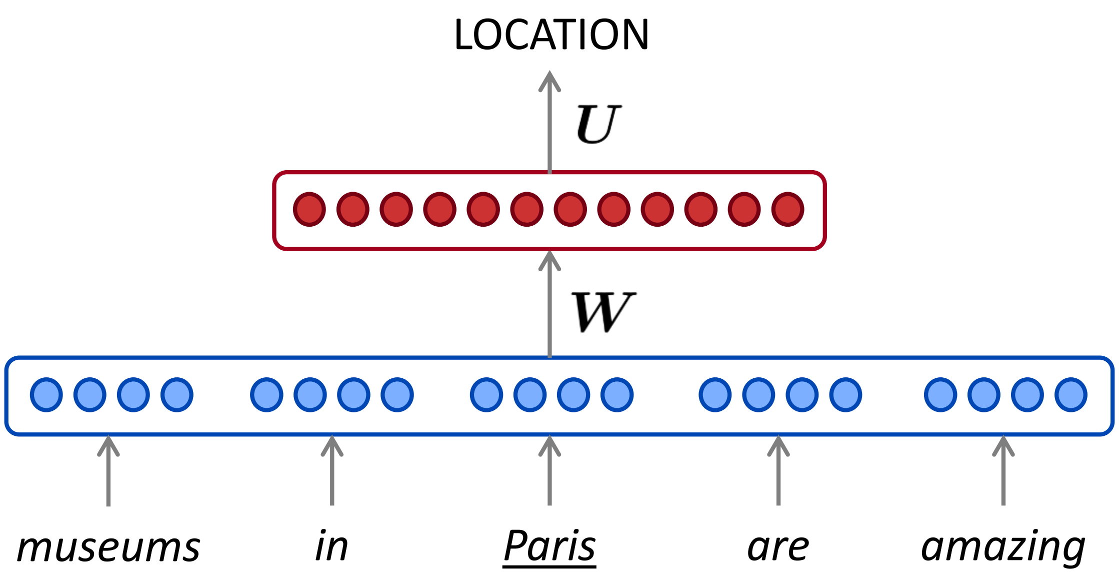
…but **incoherent.** We need to consider more than three words at a time if we want to model language well.

But increasing *n* worsens sparsity problem, and increases model size…

How to build a *neural* Language Model?

- Recall the Language Modeling task:
  - Input: sequence of words $\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, \ldots, \boldsymbol{x}^{(t)}$
  - Output: prob dist of the next word $P(\boldsymbol{x}^{(t+1)} | \ \boldsymbol{x}^{(t)}, \ldots, \boldsymbol{x}^{(1)})$

- How about a window-based neural model?
  - We saw this applied to Named Entity Recognition in Lecture 3:



A fixed-window neural Language Model

大小固定，相当于也抛弃了前面的context

~~as the proctor started the clock~~ the students opened their ______

discard

fixed window

A fixed-window neural Language Model

output distribution

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

hidden layer

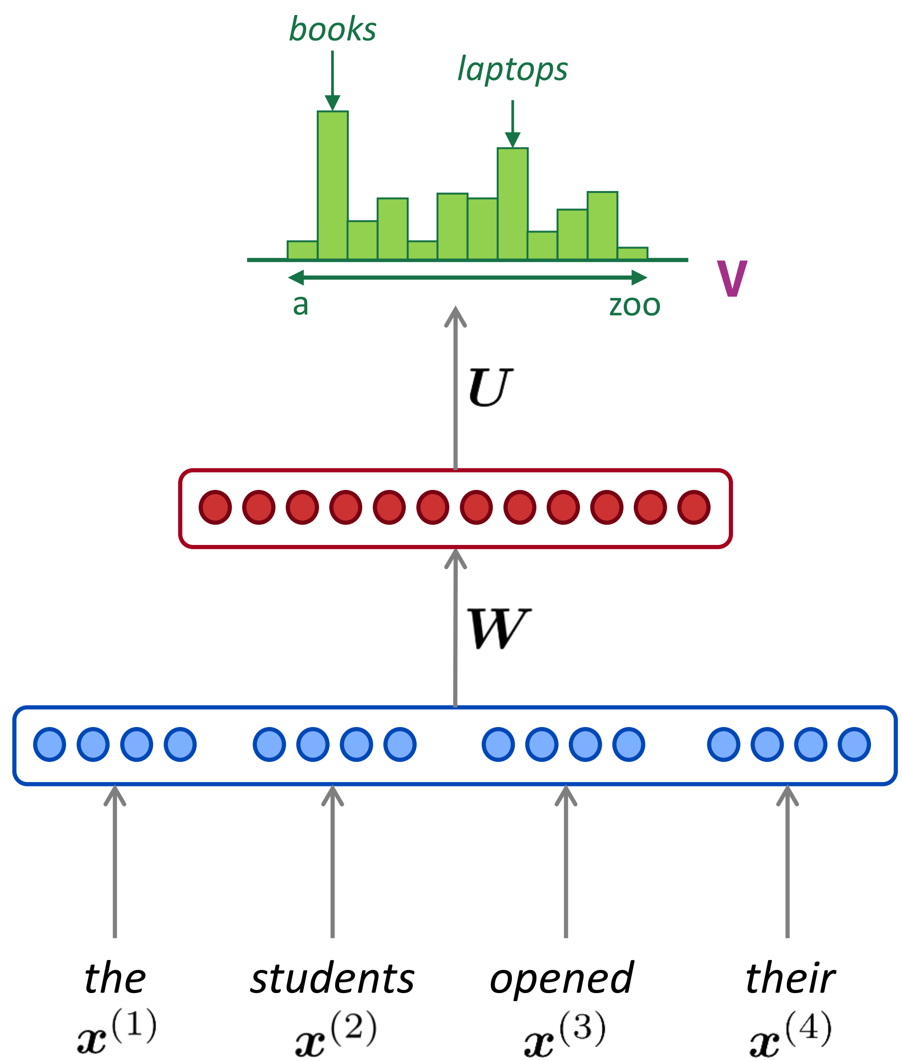
$$\boldsymbol{h} = f(\boldsymbol{W}\boldsymbol{e} + \boldsymbol{b}_1)$$

concatenated word embeddings

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

words / one-hot vectors

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



A fixed-window neural Language Model

sparsity, storage

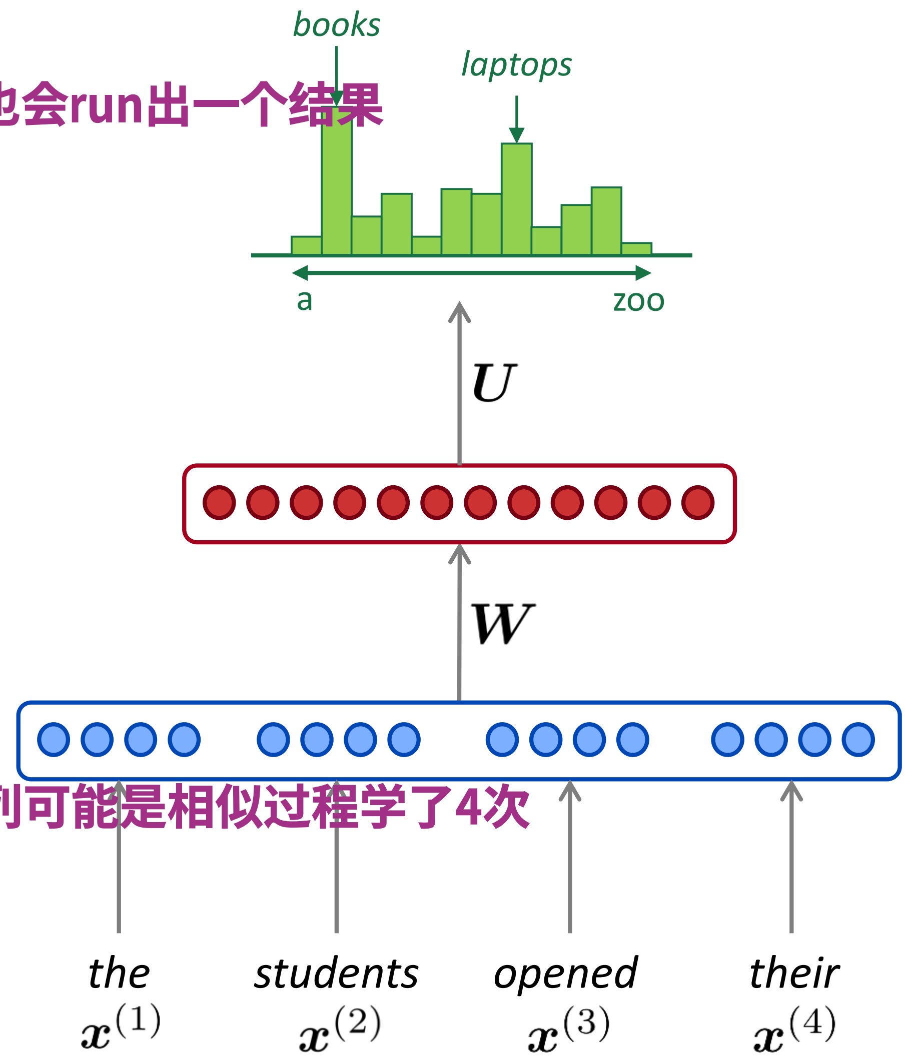
Improvements over *n*-gram LM:

- No sparsity problem 训练集没见过也会run出一个结果
- Don't need to store all observed *n*-grams

Remaining **problems**:

- Fixed window is too small
- Enlarging window enlarges $\boldsymbol{W}$
- Window can never be large enough!
- $\boldsymbol{x}^{(1)}$ and $\boldsymbol{x}^{(2)}$ are multiplied by completely different weights in $\boldsymbol{W}$. No symmetry in how the inputs are processed.

e1只和W col1有关，4列可能是相似过程学了4次



We need a neural architecture that can process *any length input*

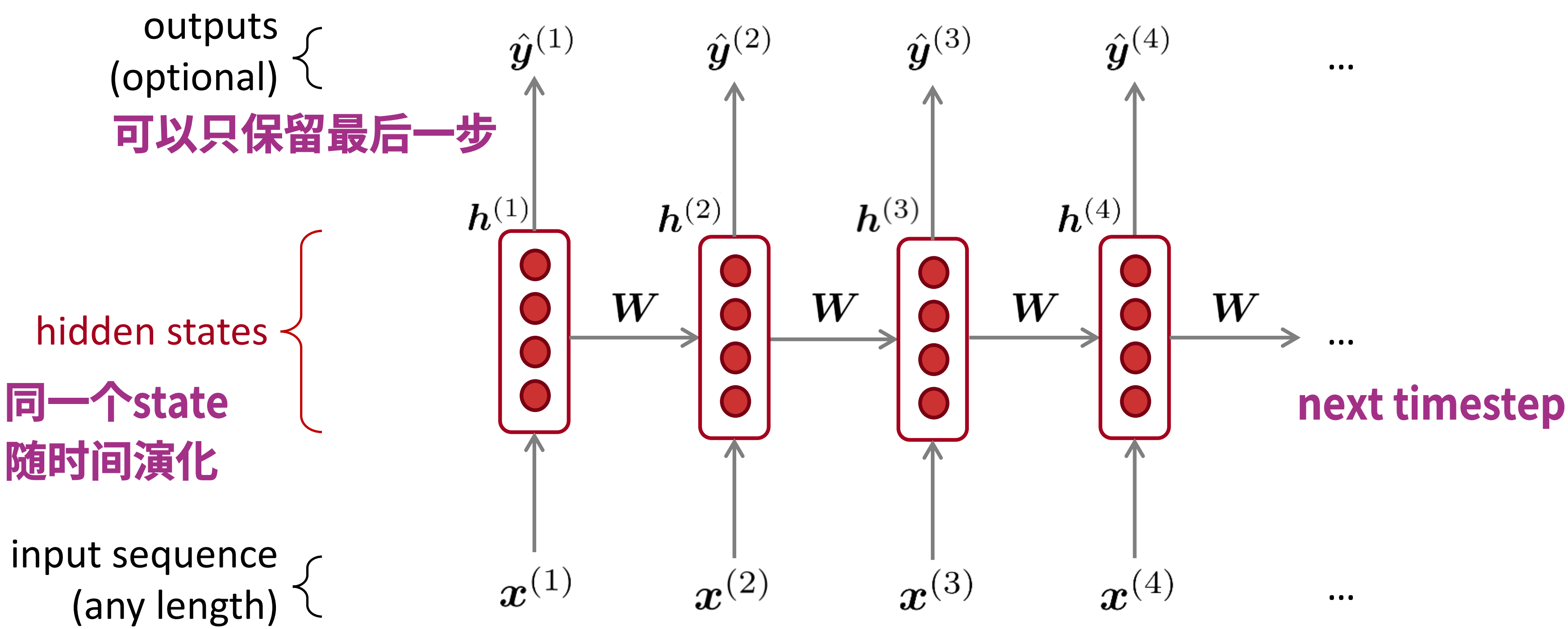
任意长度（窗的大小感觉应该随text len变化，所以不是好选择）

Recurrent Neural Networks (RNN)

A family of neural architectures

<u>Core idea:</u> Apply the same weights $\boldsymbol{W}$ *repeatedly*

每步的计算取决于该步输入&上个hidden state



A RNN Language Model

$$\hat{\boldsymbol{y}}^{(4)} = P(\boldsymbol{x}^{(5)}|\text{the students opened their})$$

output distribution

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

hidden states

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

$\boldsymbol{h}^{(0)}$ is the initial hidden state

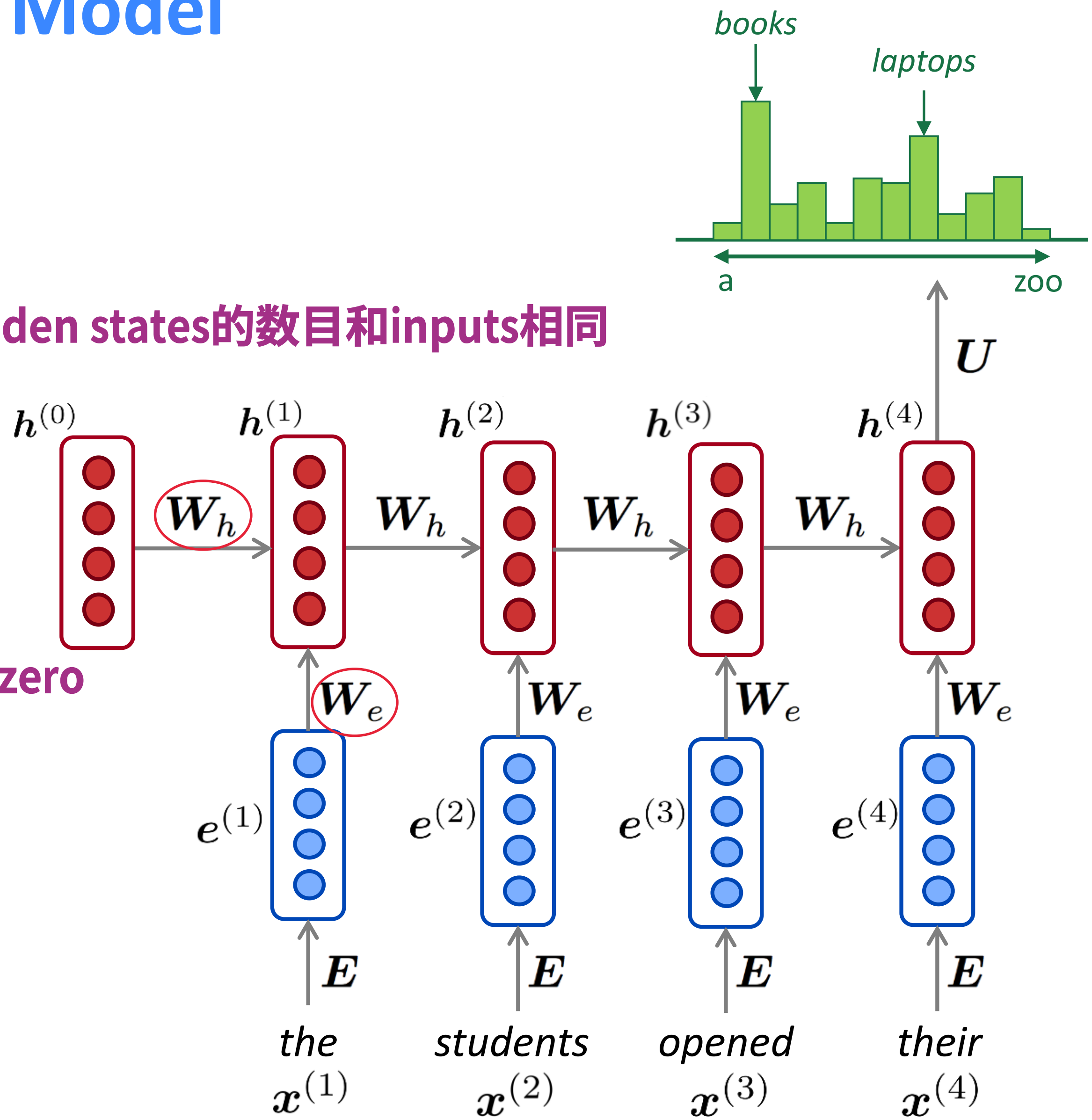
可以是学到的，也可以假定是zero

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 longer, but this slide doesn't have space!

A RNN Language Model

RNN **Advantages**:

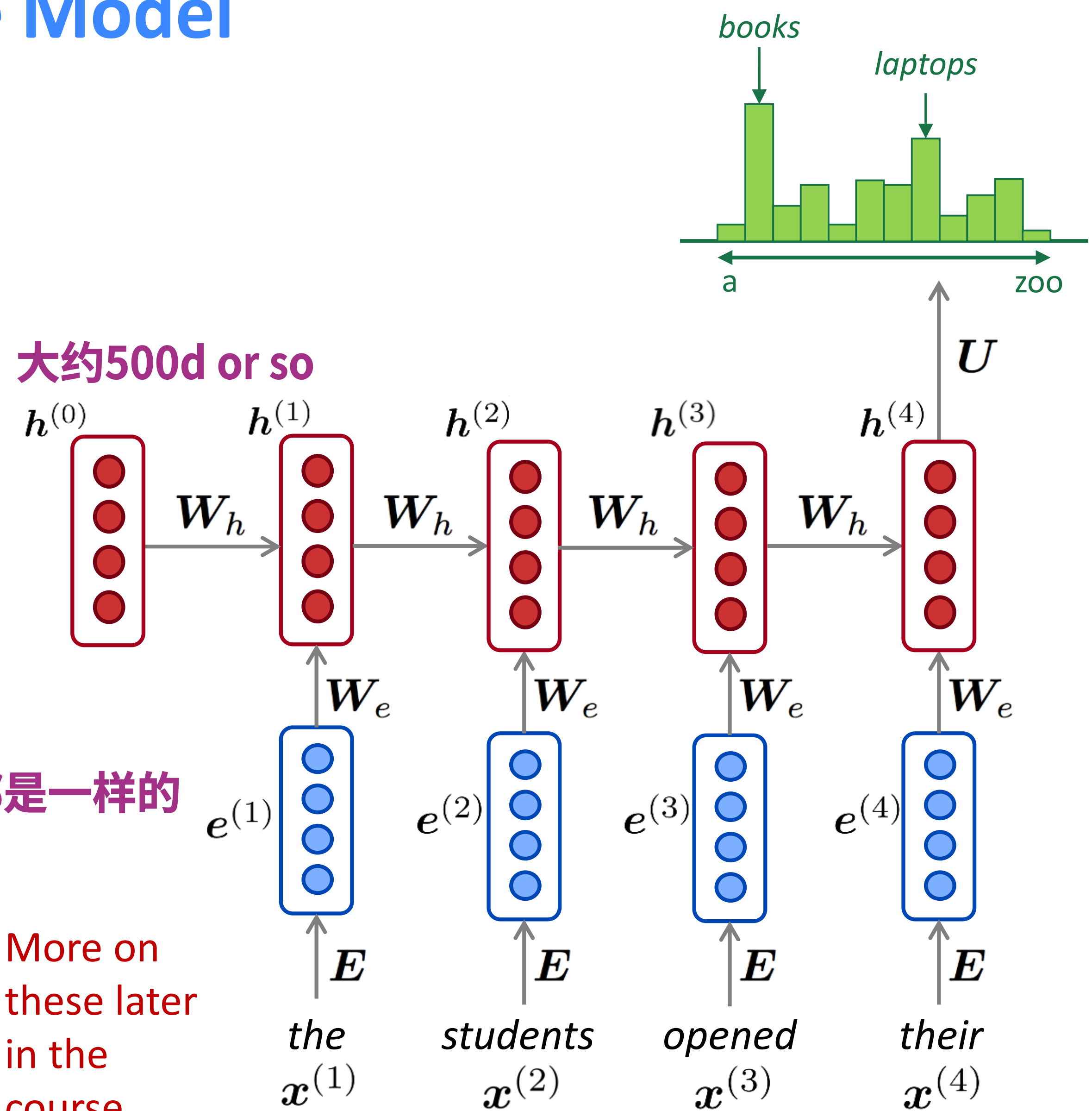
- Can process any length input
- Computation for step *t* can (in theory) use information from many steps back
- Model size doesn't increase for longer input
- Same weights applied on every timestep, so there is symmetry in how inputs are processed. 每次变换都是一样的

RNN **Disadvantages**:

- Recurrent computation is slow 【无法并行】
- In practice, difficult to access information from many steps back 遗忘问题

More on these later in the course

$$\hat{\boldsymbol{y}}^{(4)} = P(\boldsymbol{x}^{(5)}|\text{the students opened their})$$



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

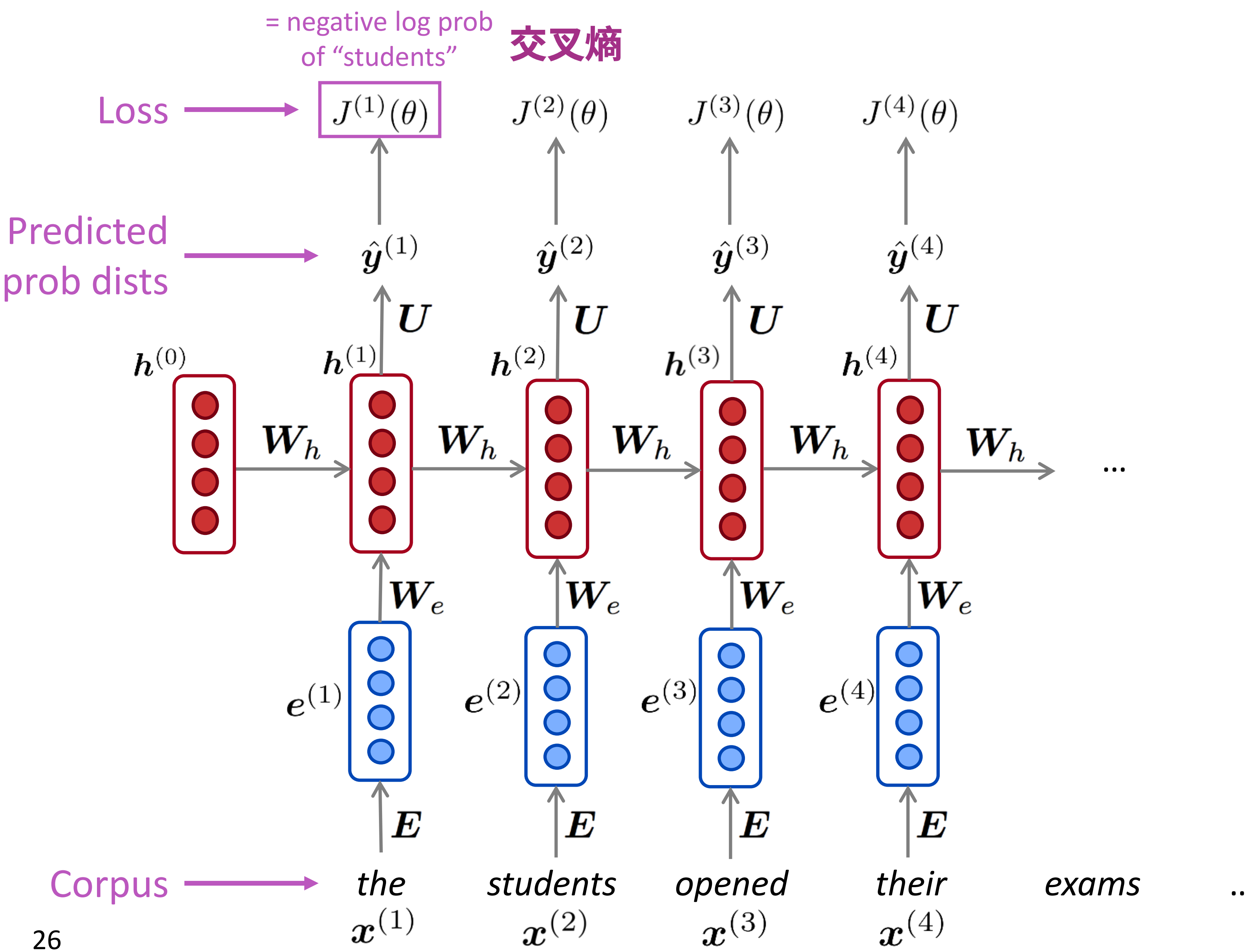
- Loss function on step *t* is cross-entropy between predicted probability distribution $\hat{\boldsymbol{y}}^{(t)}$, and the true next word $\boldsymbol{y}^{(t)}$ (one-hot for $\boldsymbol{x}^{(t+1)}$):

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

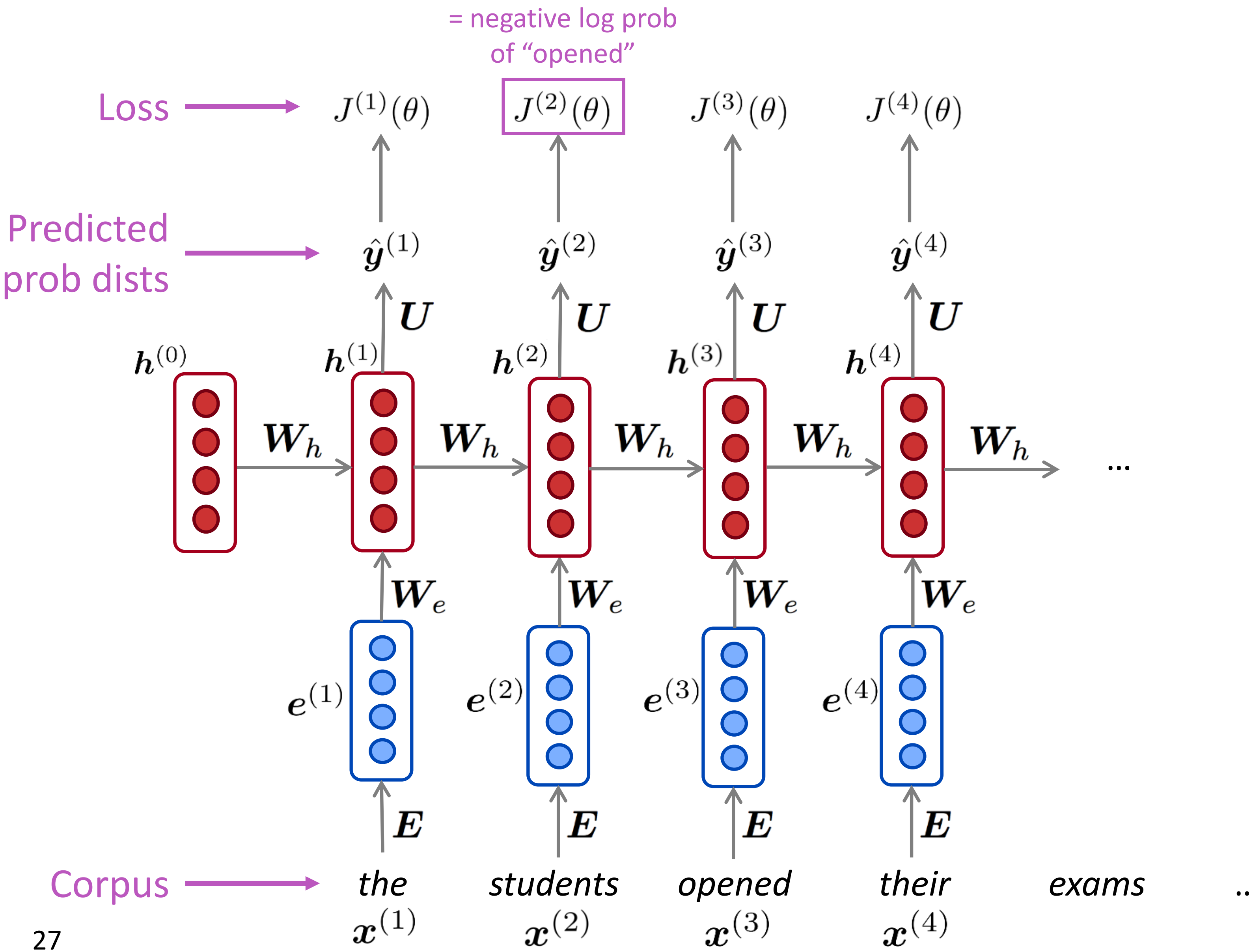
- Average this to get overall loss for entire training set:

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

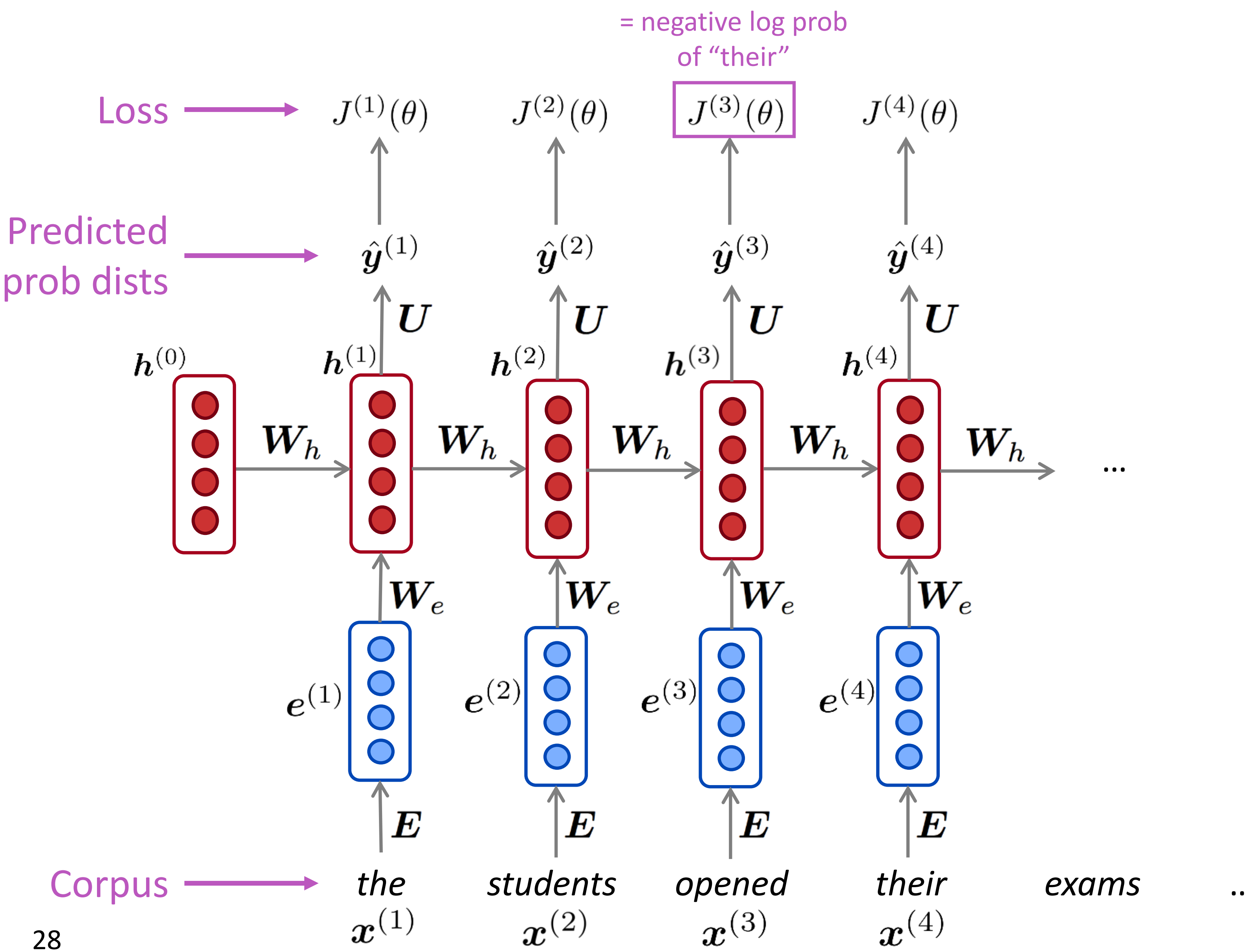
Training a RNN Language Model



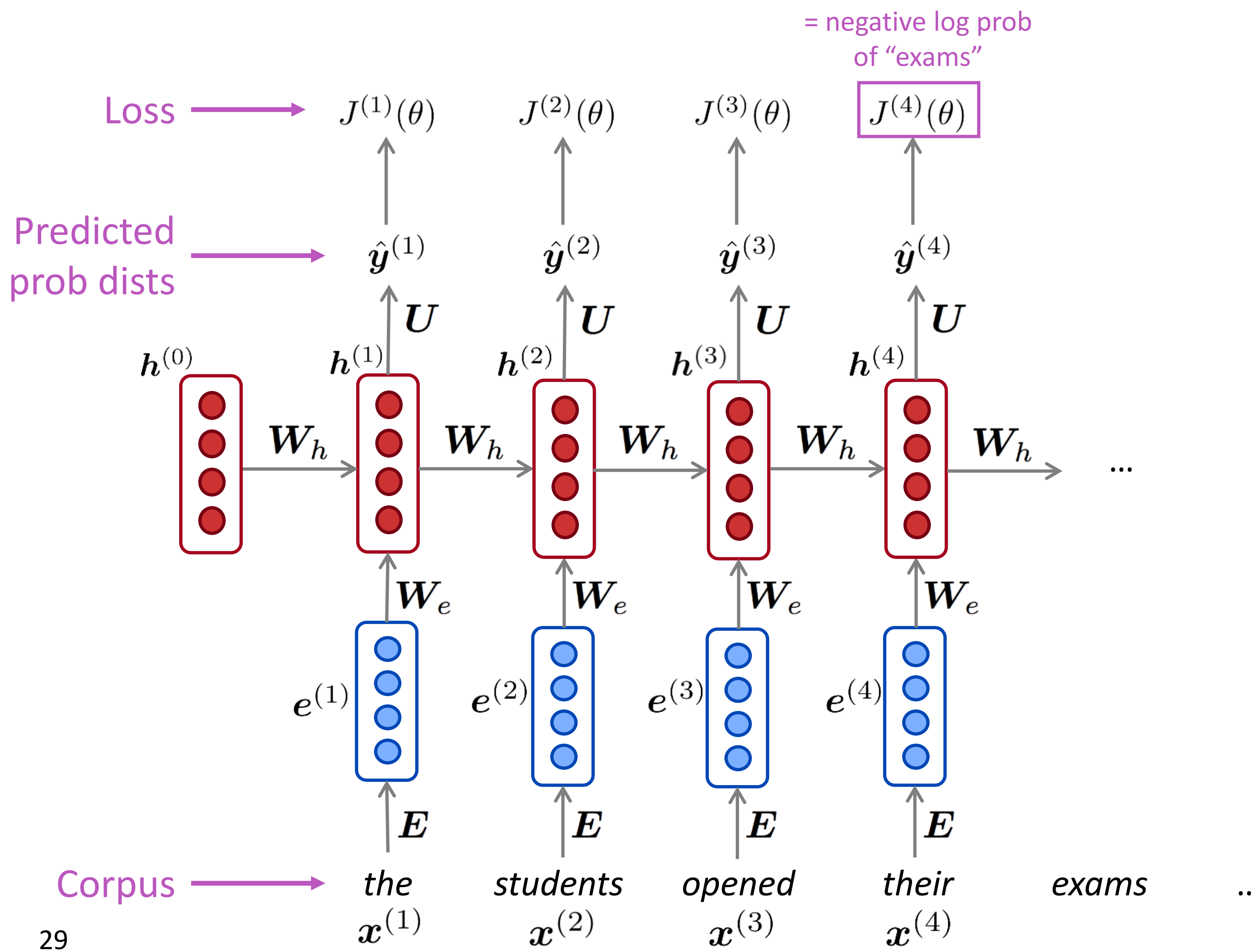
Training a RNN Language Model



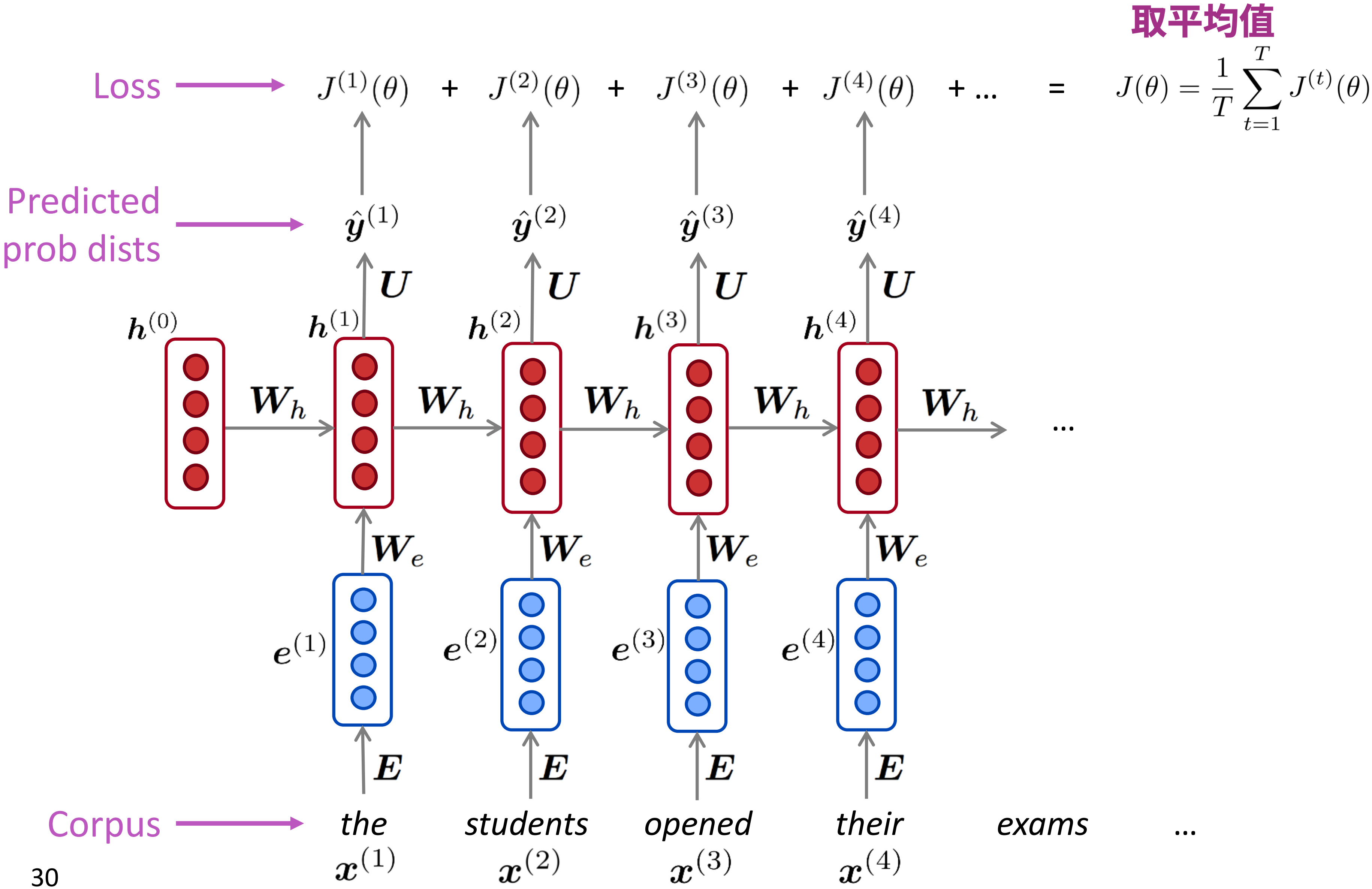
Training a RNN Language Model



Training a RNN Language Model



Training a RNN Language Model



Training a RNN Language Model

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

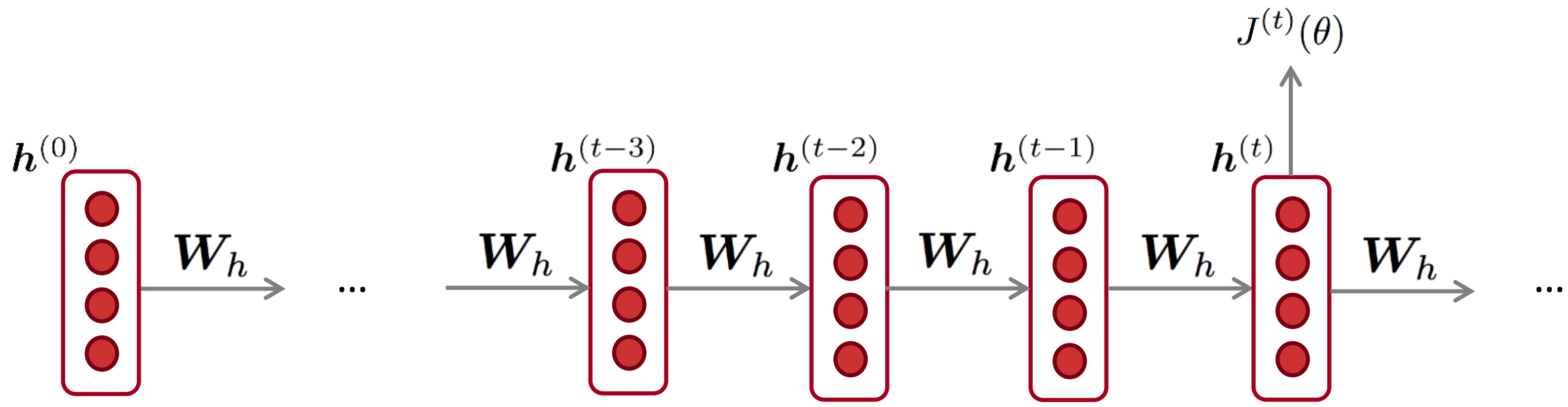
$$J(\theta) = \frac{1}{T} \sum_{t=1}^{T} J^{(t)}(\theta)$$

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

- Recall: Stochastic Gradient Descent allows us to compute loss and gradients for small chunk of data, and update.

- Compute loss $J(\theta)$ for a sentence (actually a batch of sentences), compute gradients and update weights. Repeat.

Backpropagation for RNNs



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

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

第i步的梯度

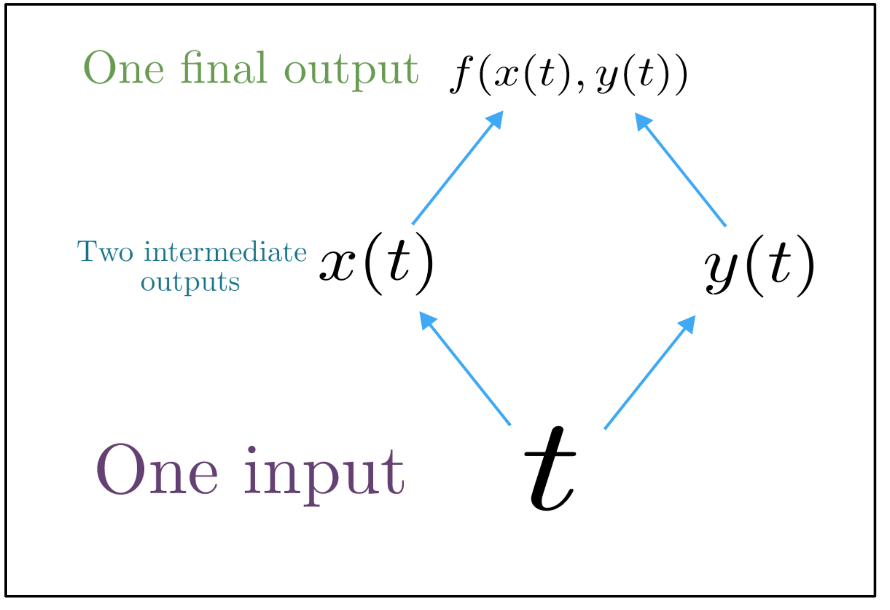
"The gradient w.r.t. a repeated weight is the sum of the gradient w.r.t. each time it appears"

Why?

Multivariable Chain Rule

- 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:

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



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

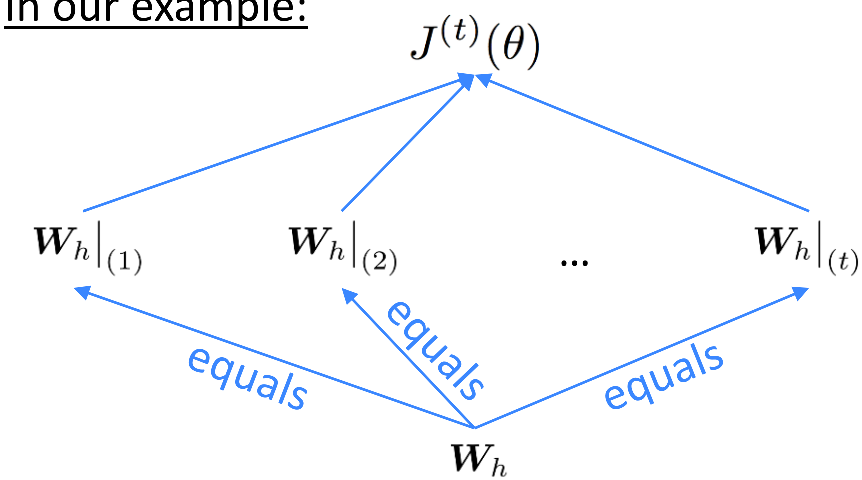
Backpropagation for RNNs: Proof sketch

- 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:

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

J^(t) = f(x_1~x_t, W_h1~W_ht)

In our example:



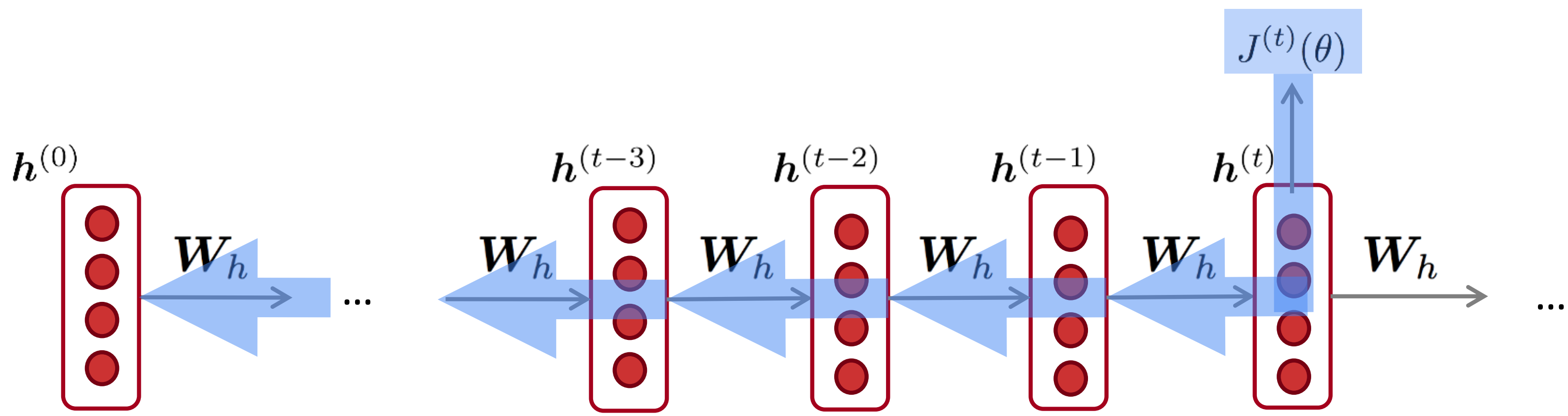
Apply the multivariable chain rule:

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

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

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

Backpropagation for RNNs



Equ的证明：把每步的输出h和变量W_h视作不同的→多元微积分

计算：从最后一步开始，对每步的W_h求梯度，这样只需back prop即可

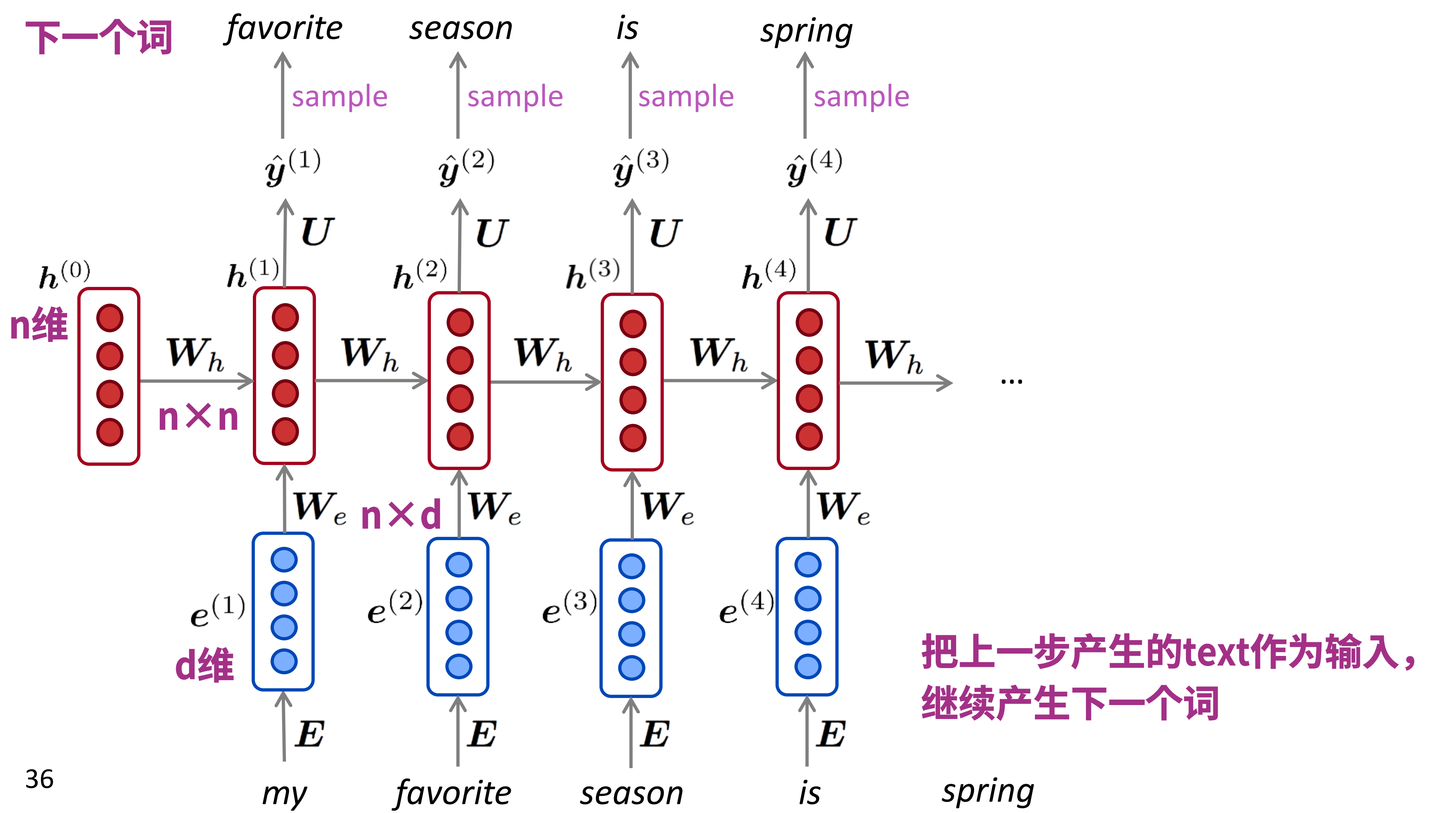
Answer: Backpropagate over timesteps *i=t,…,0*, summing gradients as you go.
This algorithm is called **"backpropagation through time"**

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

Question: How do we calculate this?

Generating text with a RNN Language Model

Just like a n-gram Language Model, you can use a RNN Language Model to generate text by repeated sampling. Sampled output is next step's input.



Generating text with a RNN Language Model

- Let's have some fun!
- You can train a RNN-LM on any kind of text, then generate text in that style.
- RNN-LM trained on Obama speeches:

language model

The United States will step up to the cost of a new challenges of the American people that will share the fact that we created the problem. They were attacked and so that they have to say that all the task of the final days of war that I will not be able to get this done.

still incoherent

Source: https://medium.com/@samim/obama-rnn-machine-generated-political-speeches-c8abd18a2ea0

Generating text with a RNN Language Model

- Let’s have some fun!
- You can train a RNN-LM on any kind of text, then generate text in that style.
- RNN-LM trained on *Harry Potter*:

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

Q: 有左右括号，这说明RNN能分辨括号吗？A: 也许，有研究看hidden states

“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.

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

Generating text with a RNN Language Model

- Let's have some fun!
- You can train a RNN-LM on any kind of text, then generate text in that style.
- RNN-LM trained on recipes:

```
    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.
```

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

Generating text with a RNN Language Model

Q: 可以把RNN结合一些人为规则使用吗？ A: 可以，比如beam search

- Let's have some fun!
- You can train a RNN-LM on any kind of text, then generate text in that style.
- RNN-LM trained on paint color names:

Ghasty Pink 231 137 165	Sand Dan 201 172 143
Power Gray 151 124 112	Grade Bat 48 94 83
Navel Tan 199 173 140	Light Of Blast 175 150 147
Bock Coe White 221 215 236	Grass Bat 176 99 108
Horble Gray 178 181 196	Sindis Poop 204 205 194
Homestar Brown 133 104 85	Dope 219 209 179
Snader Brown 144 106 74	Testing 156 101 106
Golder Craam 237 217 177	Stoner Blue 152 165 159
Hurky White 232 223 215	Burble Simp 226 181 132
Burf Pink 223 173 179	Stanky Bean 197 162 171
Rose Hork 230 215 198	Turdly 190 164 116

This is an example of a character-level RNN-LM (predicts what character comes next)

Source: http://aiweirdness.com/post/160776374467/new-paint-colors-invented-by-neural-network

Evaluating Language Models

text generation不能作为唯一的evaluation，需要一个更可衡量的

- The standard evaluation metric for Language Models is perplexity.

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

Normalized by number of words

Inverse probability of corpus, according to Language Model

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

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

训练目的是使模型能在corpus上预测高的概率

Lower perplexity is better!

RNNs have greatly improved perplexity

n-gram model

Increasingly complex RNNs

Model	Perplexity
Interpolated Kneser-Ney 5-gram (Chelba et al., 2013)	67.6
RNN-1024 + MaxEnt 9-gram (Chelba et al., 2013)	51.3
RNN-2048 + BlackOut sampling (Ji et al., 2015)	68.3
Sparse Non-negative Matrix factorization (Shazeer et al., 2015)	52.9
LSTM-2048 (Jozefowicz et al., 2016)	43.7
2-layer LSTM-8192 (Jozefowicz et al., 2016)	30
Ours small (LSTM-2048)	43.9
Ours large (2-layer LSTM-2048)	39.8

Perplexity improves (lower is better)

Source: https://research.fb.com/building-an-efficient-neural-language-model-over-a-billion-words/

Why should we care about Language Modeling?

general task（预测下一个是什么词，能衡量很多关于语言的东西）

- 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 有些听不清的词，预测是什么
    - Handwriting recognition
    - Spelling/grammar correction
    - Authorship identification 一段文字最可能是谁写的
    - Machine translation
    - Summarization
    - Dialogue 不一定都是RNN，但可以用RNN model
    - etc.

Recap

- Language Model: A system that predicts the next word

- Recurrent Neural Network: A family of neural networks that:
  - Take sequential input of any length
  - Apply the same weights on each step
  - Can optionally produce output on each step

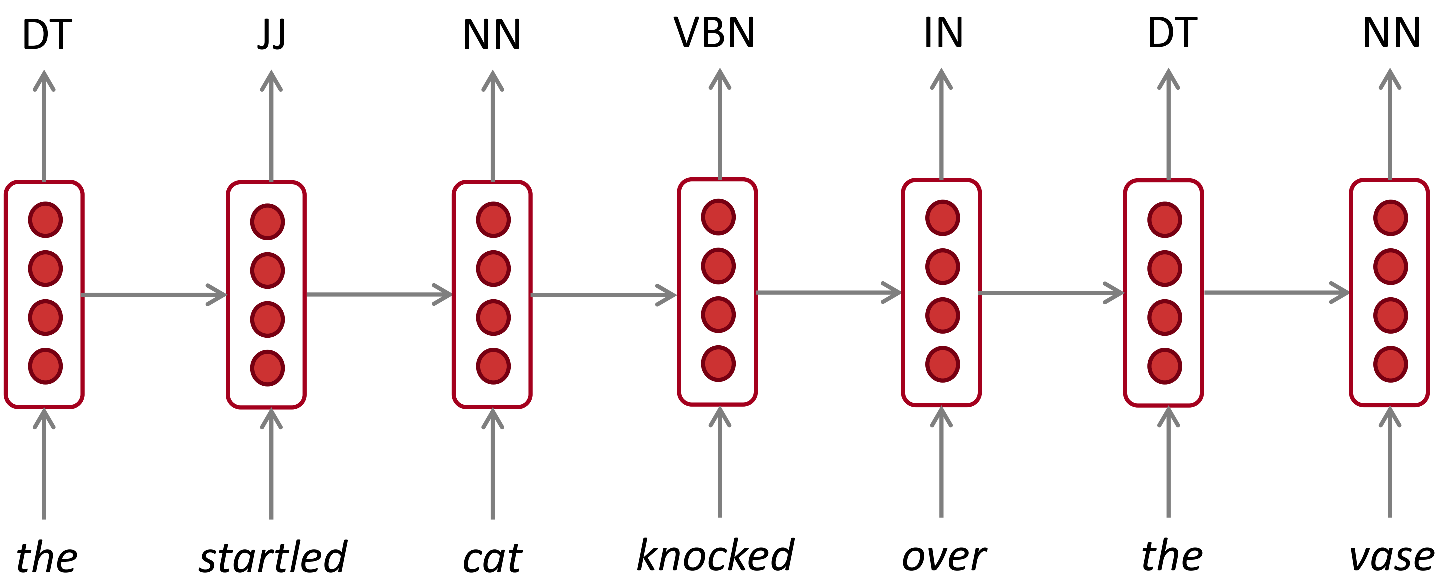
- Recurrent Neural Network ≠ Language Model

- We've shown that RNNs are a great way to build a LM.

- But RNNs are useful for much more!

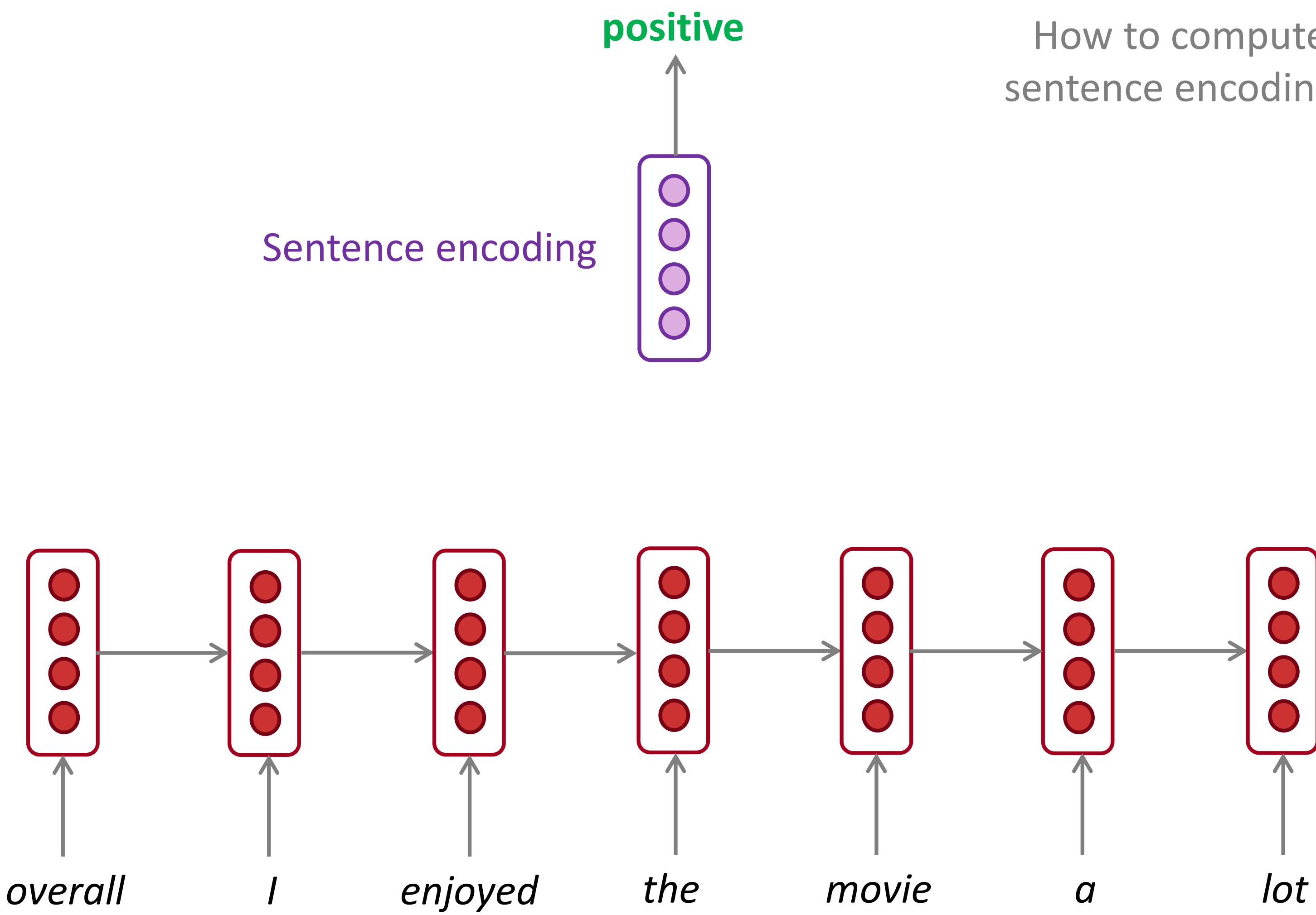
RNNs can be used for tagging

e.g. part-of-speech tagging, named entity recognition



RNNs can be used for sentence classification

e.g. sentiment classification



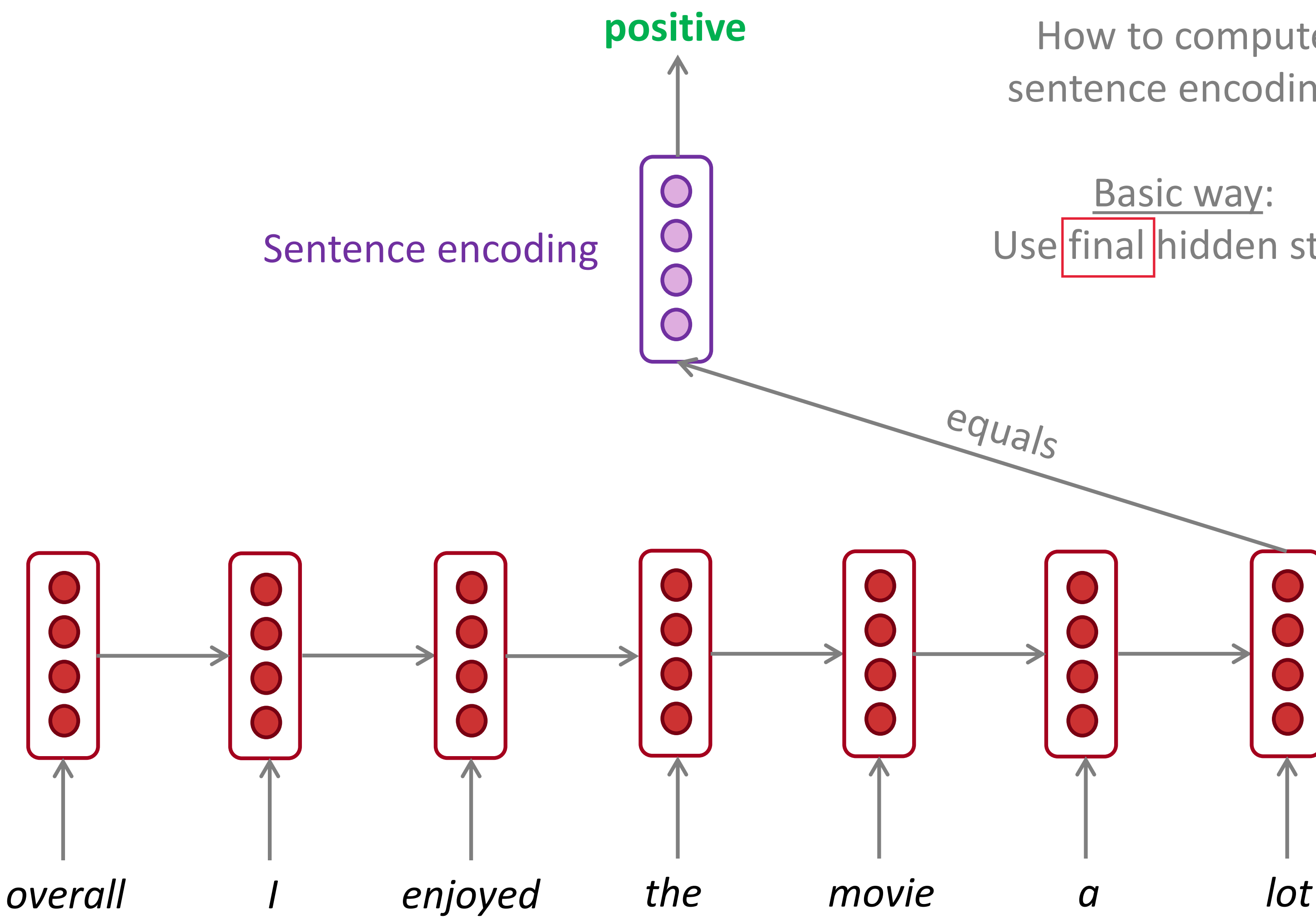
How to compute sentence encoding?

RNNs can be used for sentence classification

e.g. sentiment classification

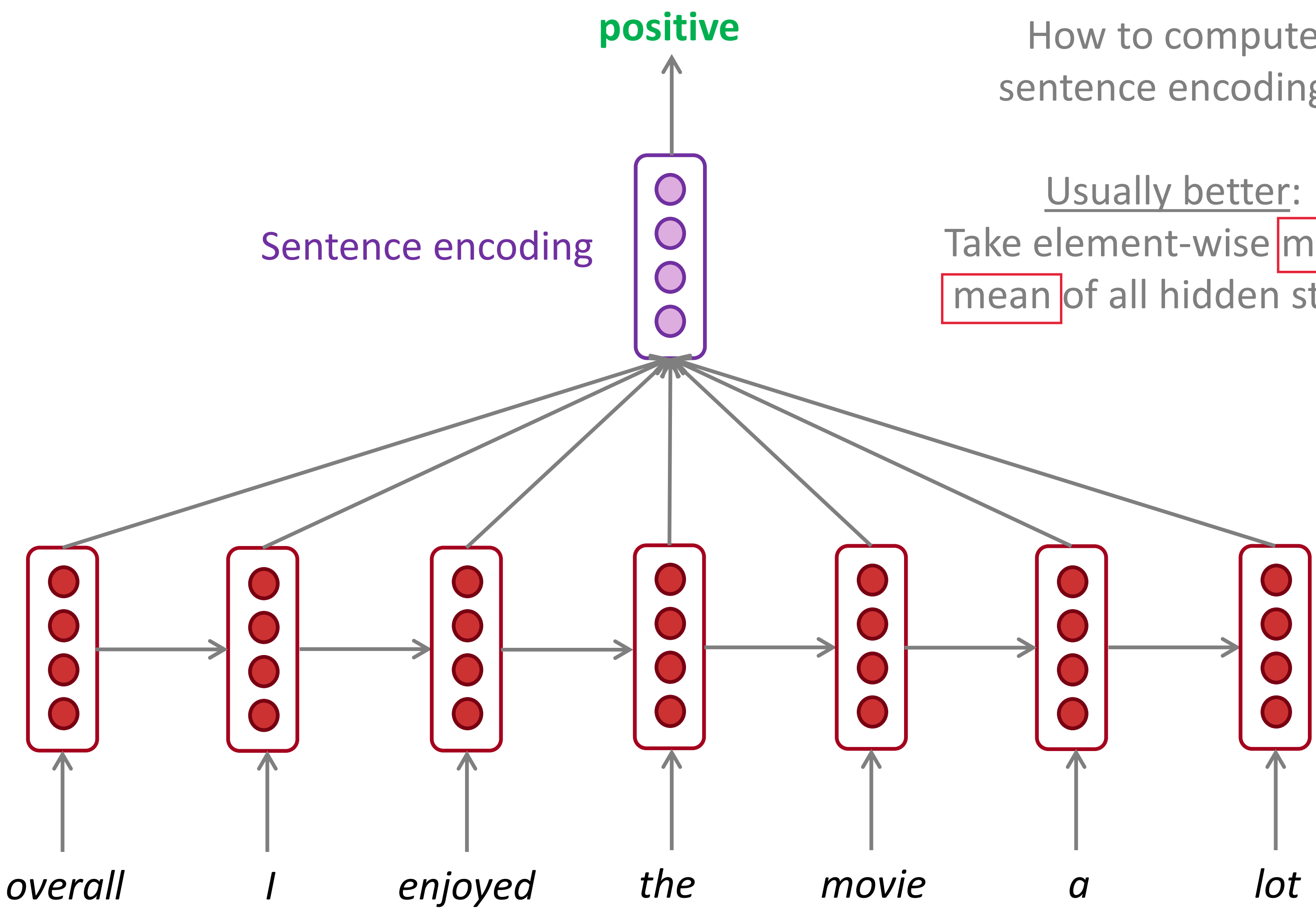
How to compute sentence encoding?

Basic way:
Use final hidden state



RNNs can be used for sentence classification

e.g. sentiment classification



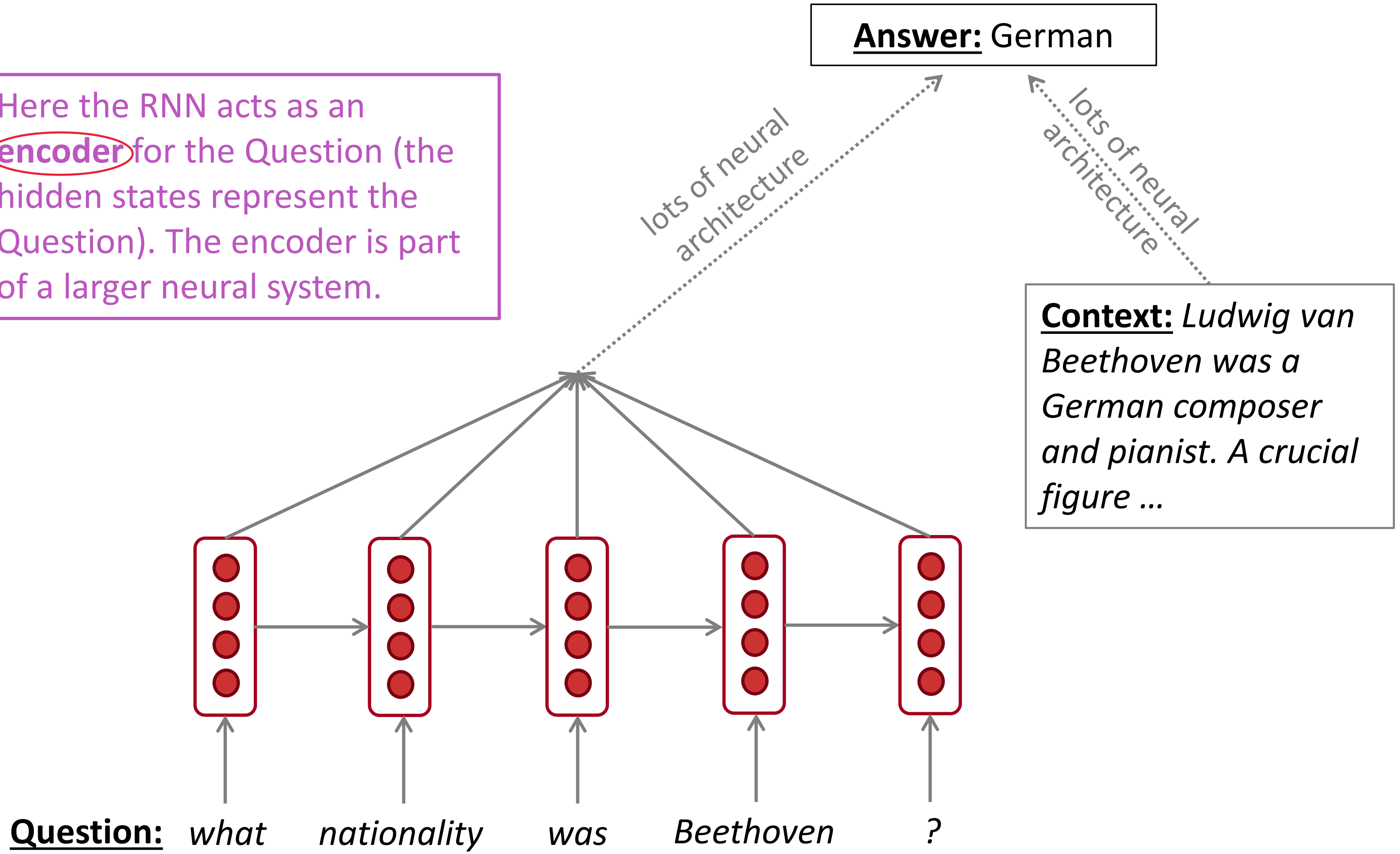
How to compute sentence encoding?

Usually better:
Take element-wise max or mean of all hidden states

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.



RNN-LMs can be used to generate text

e.g. speech recognition, machine translation, summarization

Q: 用word error rate / perplexity作为evaluation metric ?

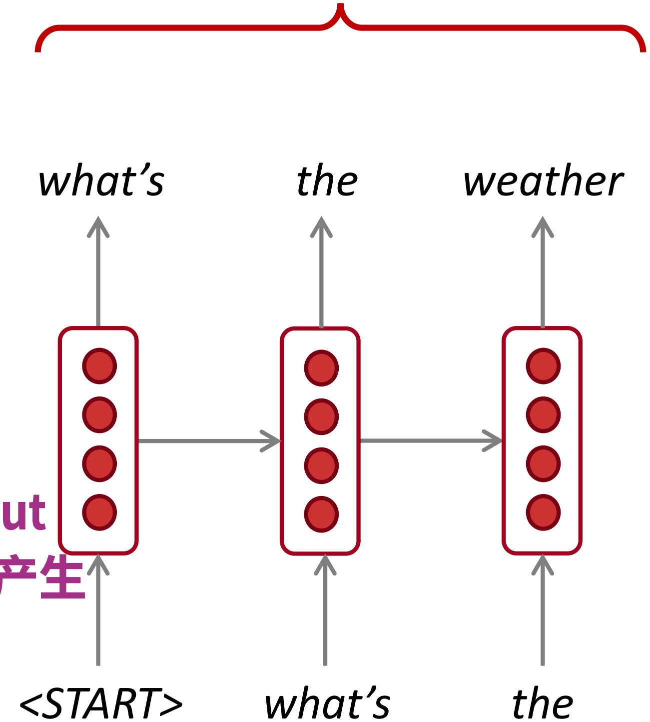
A: 一般用WER

Input (audio)

conditioning

条件：依赖input

不像HP是自己产生



This is an example of a *conditional language model*.

We'll see Machine Translation in much more detail later.

A note on terminology

RNN described in this lecture = "vanilla RNN" 

<u>Next lecture:</u> You will learn about other RNN flavors

like GRU  and LSTM  and multi-layer RNNs   

<u>By the end of the course:</u> You will understand phrases like

"stacked bidirectional LSTM with residual connections and self-attention"

Next time

- Problems with RNNs!
  - Vanishing gradients

$\Big\downarrow$ **motivates**

- Fancy RNN variants!
  - LSTM
  - GRU
  - multi-layer
  - bidirectional