Language Models, RNNLMs

COMP3361 — Week 2

Lingpeng Kong

Department of Computer Science, The University of Hong Kong
Some materials from Stanford University CS224n with special thanks!

Probabilistic Language Models

Assign a probability to a sentence

P(“I am going to school”) > P(“I are going to school”) — Grammar Checking

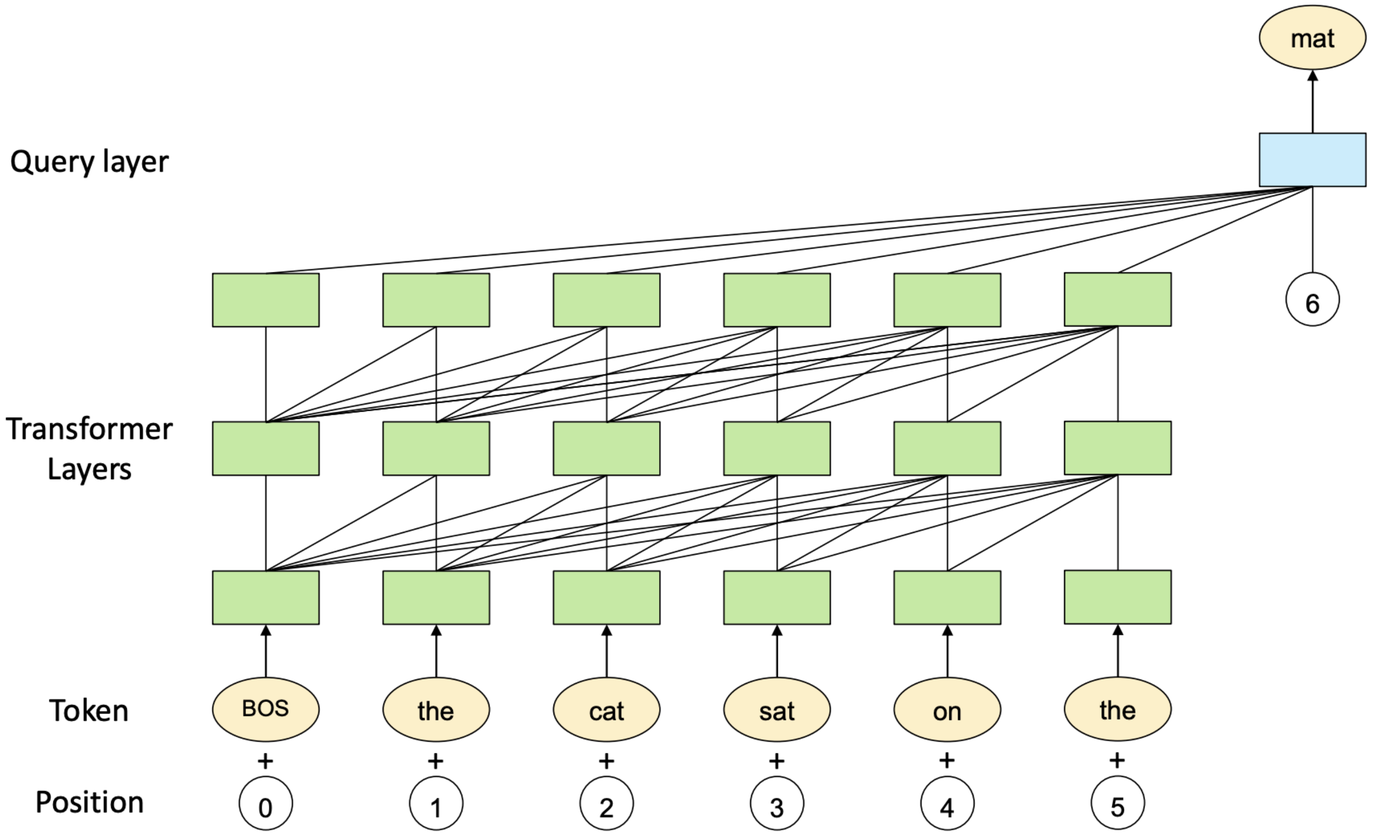
I had some coffee this morning.

P(“我今早喝了一些咖啡”) > P(“我今早吃了一些咖啡”) — Machine translation

P(“Can we put an elephant into the refrigerator? No, we can’t.”) > P(“Can we put an elephant into the refrigerator? Yes, we can.”) — Question Answering

Probabilistic Language Models

This is the reality! (And soon we will know why we can't do that merely a few years ago.)



PanGu-α (Zeng et al, April 2021)

Input (Prompt):

How smart can a cat be?

Output: 

They're incredibly independent animals, they can understand numerous things (even though they may choose to ignore you) and they even have fantastic short- and long-term memories!

Probabilistic Language Models

$$\mathcal{V} = \{\text{the}, \text{dog}, \text{laughs}, \text{saw}, \text{barks}, \text{cat}, \ldots\}$$

A sentence in the language is a sequence of words

$$x_1 x_2 \ldots x_n$$

For example

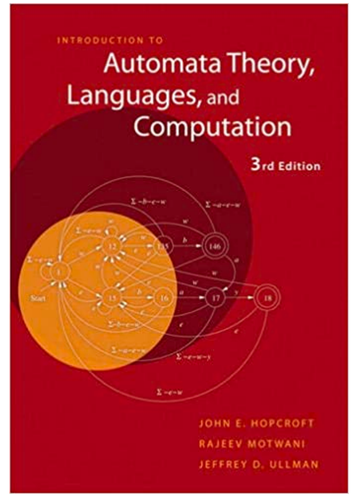
the dog barks STOP

the cat saw the dog STOP

...

Definition (Language Mode)

1. For any $\langle x_1 \ldots x_n \rangle \in \mathcal{V}^\dagger$, $p(x_1, x_2, \ldots x_n) \geq 0$
2. In addition, $\sum_{\langle x_1 \ldots x_n \rangle \in \mathcal{V}^\dagger} p(x_1, x_2, \ldots x_n) = 1$



(Hopcroft, Motwani, Ullman)

A (very bad) method for learning a LM

Number of times the sentence $x_1 \dots x_n$ is seen in the training corpus

$$c(x_1 \dots x_n)$$

Total number of sentences in the training corpus N

$$p(x_1 \dots x_n) = \frac{c(x_1 \dots x_n)}{N}$$

Why this is very bad?

Markov Models

Consider a sequence of random variables $X_1, X_2, \ldots, X_n$, each take any value in $\mathcal{V}$

The joint probability of a sentence is

$$P(X_1 = x_1, X_2 = x_2, \ldots, X_n = x_n)$$

$$= P(X_1 = x_1) \prod_{i=2}^{n} P(X_i = x_i | X_1 = x_1, \ldots, X_{i-1} = x_{i-1})$$

$$= P(X_1 = x_1) \prod_{i=2}^{n} P(X_i = x_i | X_{i-1} = x_{i-1})$$

First-order Markov Assumption

Trigram Language Models

A trigram language model consists of a finite set $\mathcal{V}$, and a parameter $q(w \mid u, v)$

For each trigram u, v, w , such that $w \in \mathcal{V} \cup \{\text{STOP}\}$, $u, v \in \mathcal{V} \cup \{*\}$.

$q(w \mid u, v)$ can be interpreted as the probability of seeing the word w immediately after the bigram (u, v) .

For any sentence $x_1 \dots x_n$, where $x_i \in \mathcal{V}$ for $i = 1 \dots (n-1)$, and $x_n = \text{STOP}$

$$p(x_1 \dots x_n) = \prod_{i=1}^{n} q(x_i \mid x_{i-2}, x_{i-1})$$

where we define $x_0 = x_{-1} = *$

Trigram Language Models

For example, for the sentence

the dog barks STOP

$$p(\text{the dog barks STOP}) = q(\text{the} \mid *, *) \times q(\text{dog} \mid *, \text{the}) \times q(\text{barks} \mid \text{the}, \text{dog}) \times q(\text{STOP} \mid \text{dog}, \text{barks})$$

Problem solved? How can we find $q(w \mid u, v)$

Parameters (of the model)

$q(w \mid u, v)$

How many parameters?

How to “estimate” them from training data?

Trigram Language Models

Parameters (of the model)

$$q(w \mid u, v)$$

How many parameters?

$$|\mathcal{V}|^3$$

How to "estimate" them from training data?

$$q(w \mid u, v) = \frac{c(u, v, w)}{c(u, v)}$$

$$q(\text{barks} \mid \text{the}, \text{dog}) = \frac{c(\text{the}, \text{dog}, \text{barks})}{c(\text{the}, \text{dog})}$$

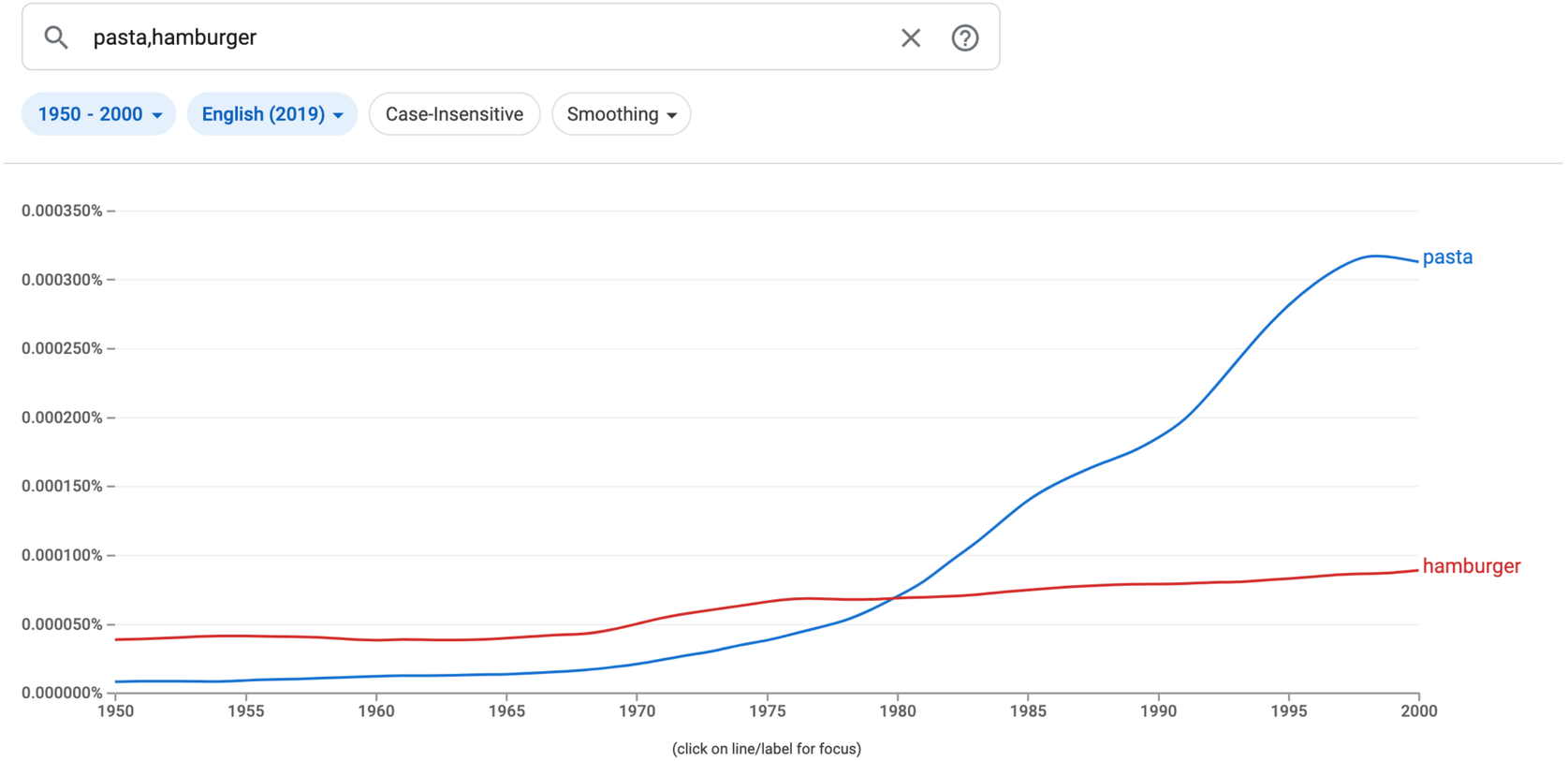
Trigram Language Models

How to “estimate” them from training data?

$$q(w \mid u, v) = \frac{c(u, v, w)}{c(u, v)}$$

$$q(\text{barks} \mid \text{the}, \text{dog}) = \frac{c(\text{the}, \text{dog}, \text{barks})}{c(\text{the}, \text{dog})}$$

N-gram counts!



Pasta v.s. Hamburger (Google Books Ngram Viewer)

Sparse Data Problems

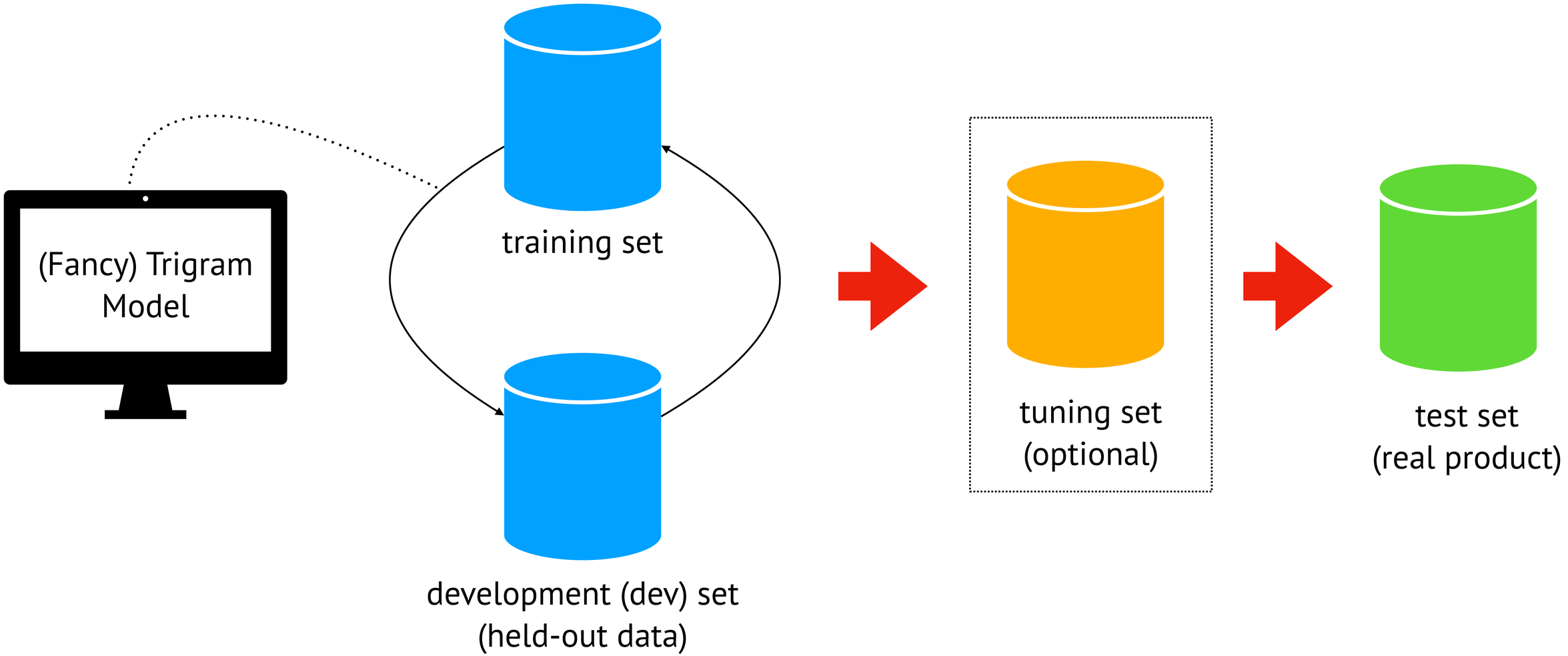
Maximum likelihood estimate:

$$q(w \mid u, v) = \frac{c(u, v, w)}{c(u, v)}$$

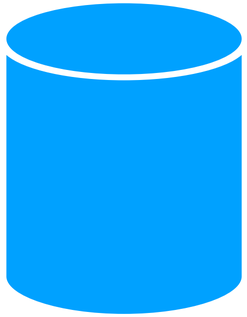
$$q(\text{barks} \mid \text{the}, \text{dog}) = \frac{c(\text{the}, \text{dog}, \text{barks})}{c(\text{the}, \text{dog})}$$

$|\mathcal{V}|^3$ Say vocabulary size is 20000. We have 8 *10^{12} parameters!!

Evaluating Language Models: Perplexity



Evaluating Language Models: Perplexity



development (dev) set
(held-out data)

. . .

$x^{(i)}$ the cat laughs STOP

$x^{(i+1)}$ the dog laughs at the cat STOP

. . .

We can compute the probability it assigns to the entire set of test sentences

$$\prod_{i=1}^{m} p(x^{(i)})$$

The higher this quantity is, the better the language model is at modeling unseen sentences.

Evaluating Language Models: Perplexity

The higher this quantity is, the better the language model is at modeling unseen sentences.

$$\prod_{i=1}^{m} p(x^{(i)})$$

Perplexity on the test corpus is derived as a direction transformation of this.

$$\text{ppl} = 2^{-l}$$

$$l = \frac{1}{M} \sum_{i=1}^{m} \log_2 p(x^{(i)})$$

M is the total length of the sentences in the test corpus.

What if the model estimate $q(w \mid u, v) = 0$ and the trigram appears in the dataset?

Wait, why we love this number in the first place?

Let the model predicts $q(w \mid u, v) = 1/N$

$$l = \frac{1}{M} \sum_{i=1}^{m} \log_2 p(x^{(i)})$$

$$\text{ppl} = 2^{-l} = N$$

A uniform probability model — The perplexity is equal to the vocabulary size!

Perplexity can be thought of as the effective vocabulary size under the model! For example, the perplexity of the model is 120 (even though the vocabulary size is say 10,000), then this is roughly equivalent to having an effective vocabulary of 120.

How much your language model updates the uniform guess!

Bayes Factors:
https://www.youtube.com/watch?v=lG4VkPoG3ko

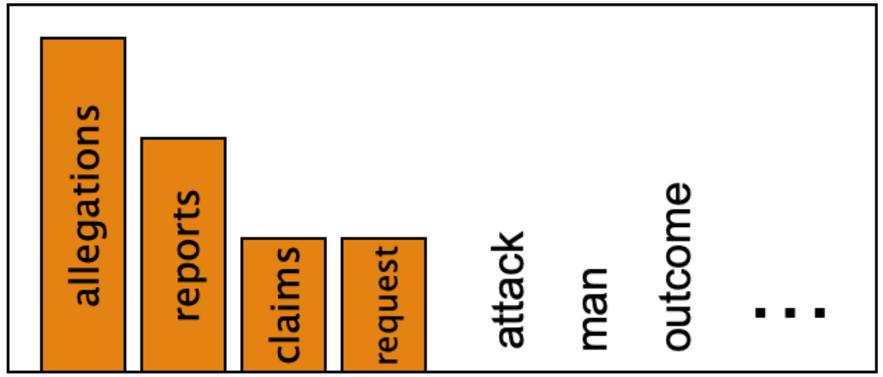
Smoothing for Language Models

If the model estimate $q(w \mid u, v) = 0$ and the trigram appears in the test data, ppl goes up to infinity.

When we have sparse statistics:

P(w | denied the)
- 3 allegations
- 2 reports
- 1 claims
- 1 request

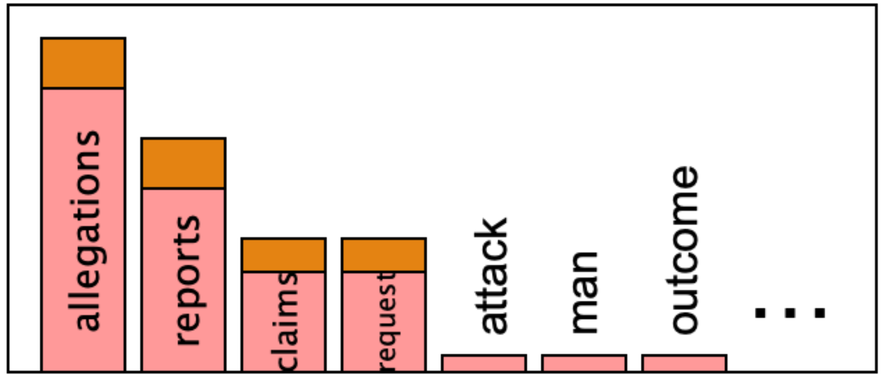
7 total



Steal probability mass to generalize better:

P(w | denied the)
- 2.5 allegations
- 1.5 reports
- 0.5 claims
- 0.5 request
- 2 other

7 total



Example from Dan Klein

Add-one (Laplace) smoothing

Considering a bigram model here, pretend we saw each word one more time than we did.

MLE estimate:

$$q_{\text{MLE}}(w_i|w_{i-1}) = \frac{c(w_{i-1}, w_i)}{c(w_{i-1})}$$

Add-one smoothing:

$$q_{\text{Laplace}}(w_i|w_{i-1}) = \frac{c(w_{i-1}, w_i) + 1}{c(w_{i-1}) + |\mathcal{V}|}$$

Linear Interpolation (Stupid Backoff)

Trigram Model, Bigram Model, Unigram Model

Trigram maximum-likelihood estimate: $q(w_i \mid w_{i-2}, w_{i-1}) = \frac{c(w_{i-2}, w_{i-1}, w_i)}{c(w_{i-2}, w_{i-1})}$

Bigram maximum-likelihood estimate: $q(w_i \mid w_{i-1}) = \frac{c(w_{i-1}, w_i)}{c(w_{i-1})}$

Unigram maximum-likelihood estimate: $q(w_i) = \frac{c(w_i)}{c(\cdot)}$

Which one suffers from the data sparsity problem the most?
Which one is more accurate?

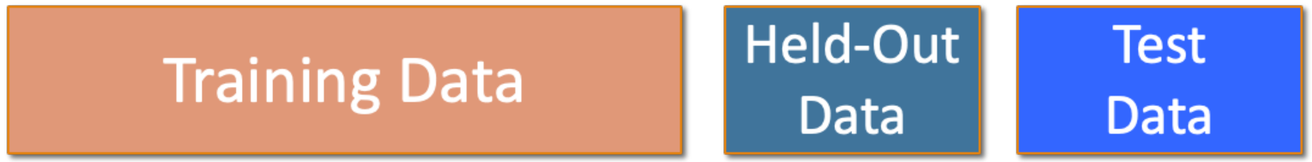
Linear Interpolation (Stupid Backoff)

$$
\begin{aligned}
q(w_i \mid w_{i-2}, w_{i-1}) = \quad & \lambda_1 \times q_{\mathrm{ML}}(w_i \mid w_{i-2}, w_{i-1}) \\
& + \lambda_2 \times q_{\mathrm{ML}}(w_i \mid w_{i-1}) \\
& + \lambda_3 \times q_{\mathrm{ML}}(w_i)
\end{aligned}
$$

$$\text{where } \lambda_1 + \lambda_2 + \lambda_3 = 1, \text{ and } \lambda_i \geq 0 \text{ for all } i.$$

How to choose the value of $\lambda_1, \lambda_2, \lambda_3$

Use the held-out corpus Hyperparameters



maximize the probability of held-out data.

Markov Models in Retrospect

Consider a sequence of random variables $X_1, X_2, \ldots, X_n$, each take any value in $\mathcal{V}$

The joint probability of a sentence is

$$P(X_1 = x_1, X_2 = x_2, \ldots, X_n = x_n)$$

$$= P(X_1 = x_1) \prod_{i=2}^{n} P(X_i = x_i | X_1 = x_1, \ldots, X_{i-1} = x_{i-1})$$

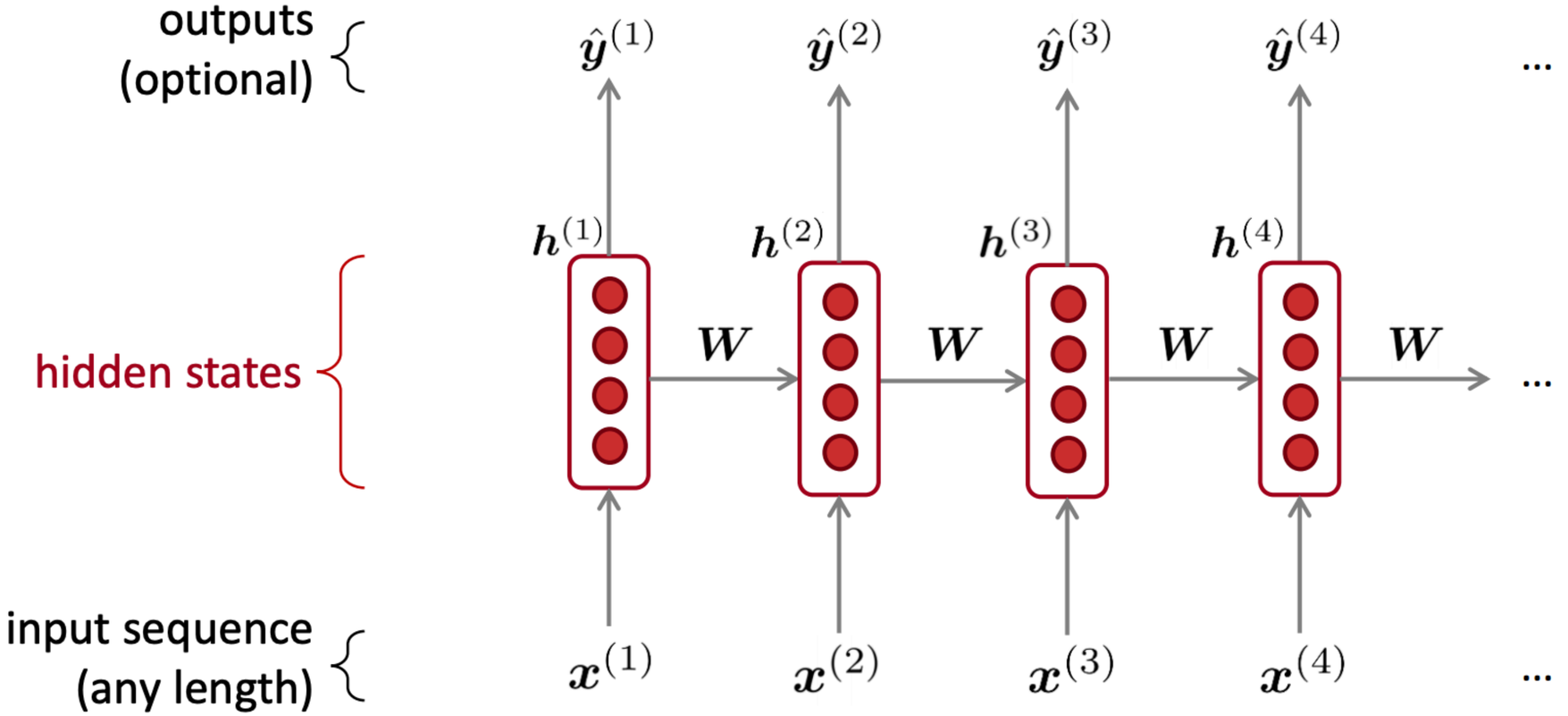
$$= P(X_1 = x_1) \prod_{i=2}^{n} P(X_i = x_i | X_{i-1} = x_{i-1})$$

First-order Markov Assumption

$$P(X_i = x_i | X_1 = x_1, \ldots, X_{i-1} = x_{i-1})$$

Is it possible to directly model this probability?

Recurrent Neural Networks (RNNs)



$$p(x^{(t)} \mid x^{(t-1)}, \ldots, x^{(1)})$$

A RNN Language Model

output distribution

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

hidden states

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

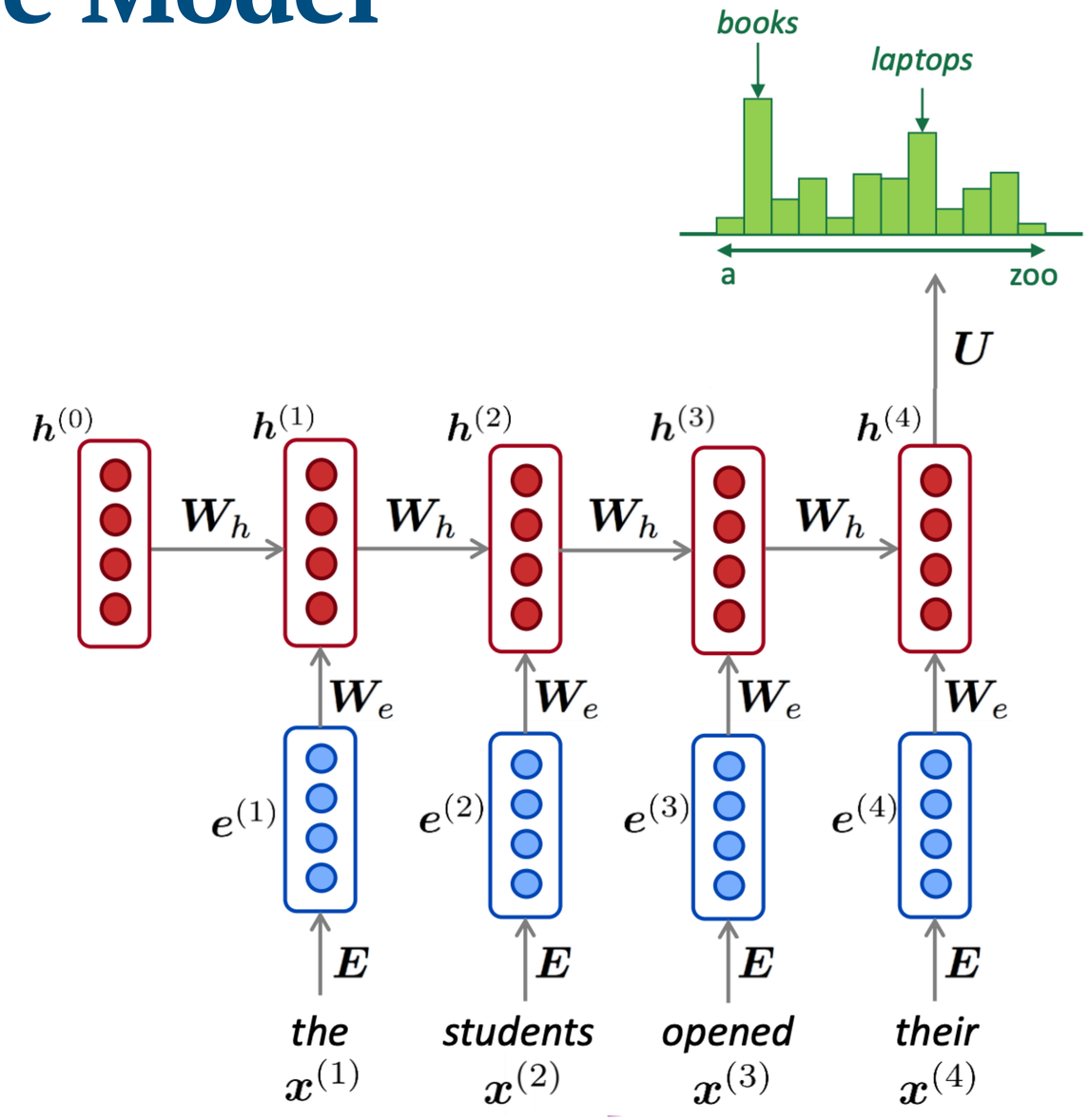
$\boldsymbol{h}^{(0)}$ is the initial hidden state

word embeddings

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

words / one-hot vectors

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



A RNN Language Model

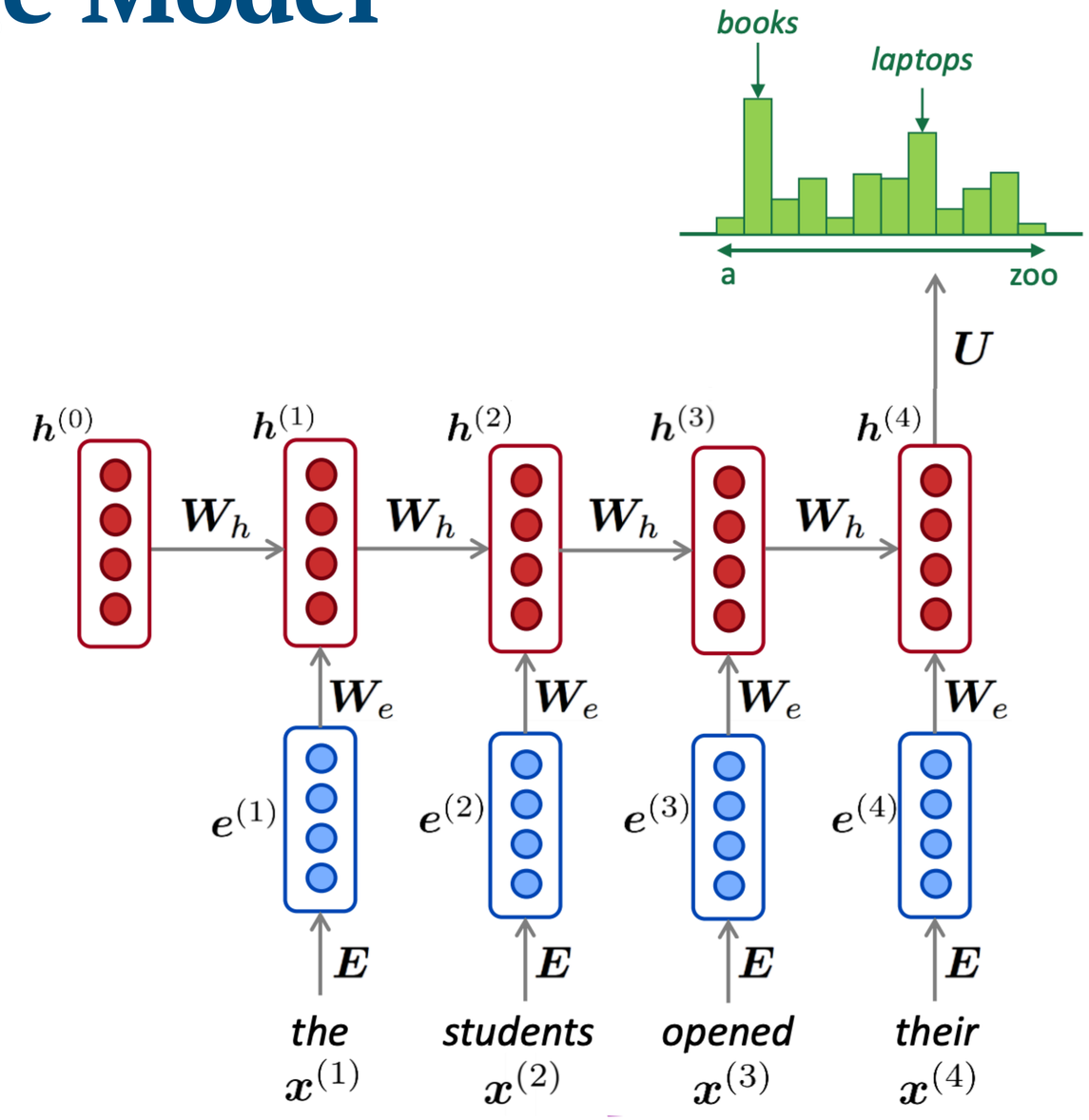
$$p(x^{(t)} \mid x^{(t-1)}, \ldots, x^{(1)})$$

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

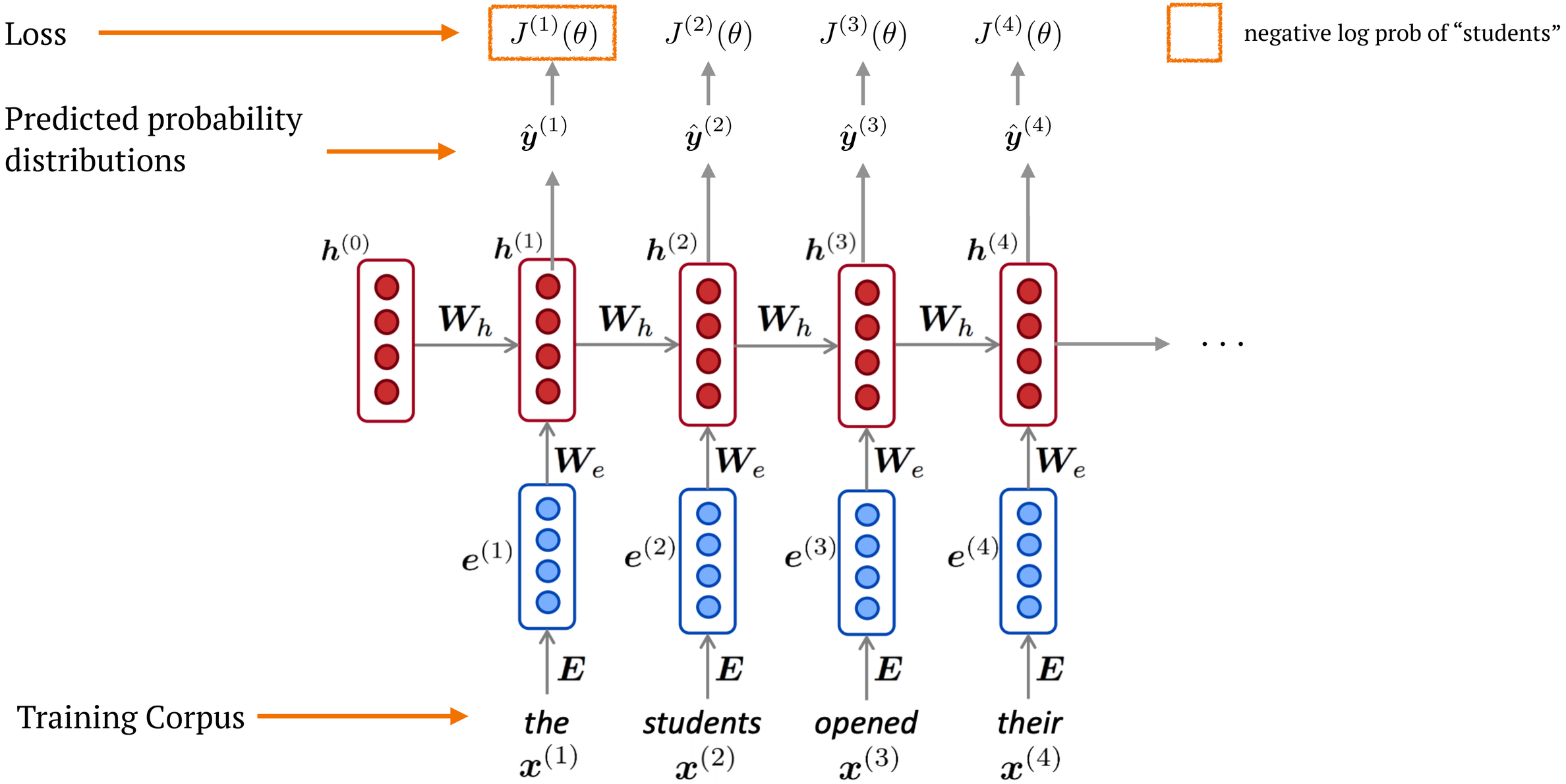
$$\boldsymbol{h}^{(t)} = \sigma\left(\boldsymbol{W}_h\boldsymbol{h}^{(t-1)} + \boldsymbol{W}_e\boldsymbol{e}^{(t)} + \boldsymbol{b}_1\right)$$

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

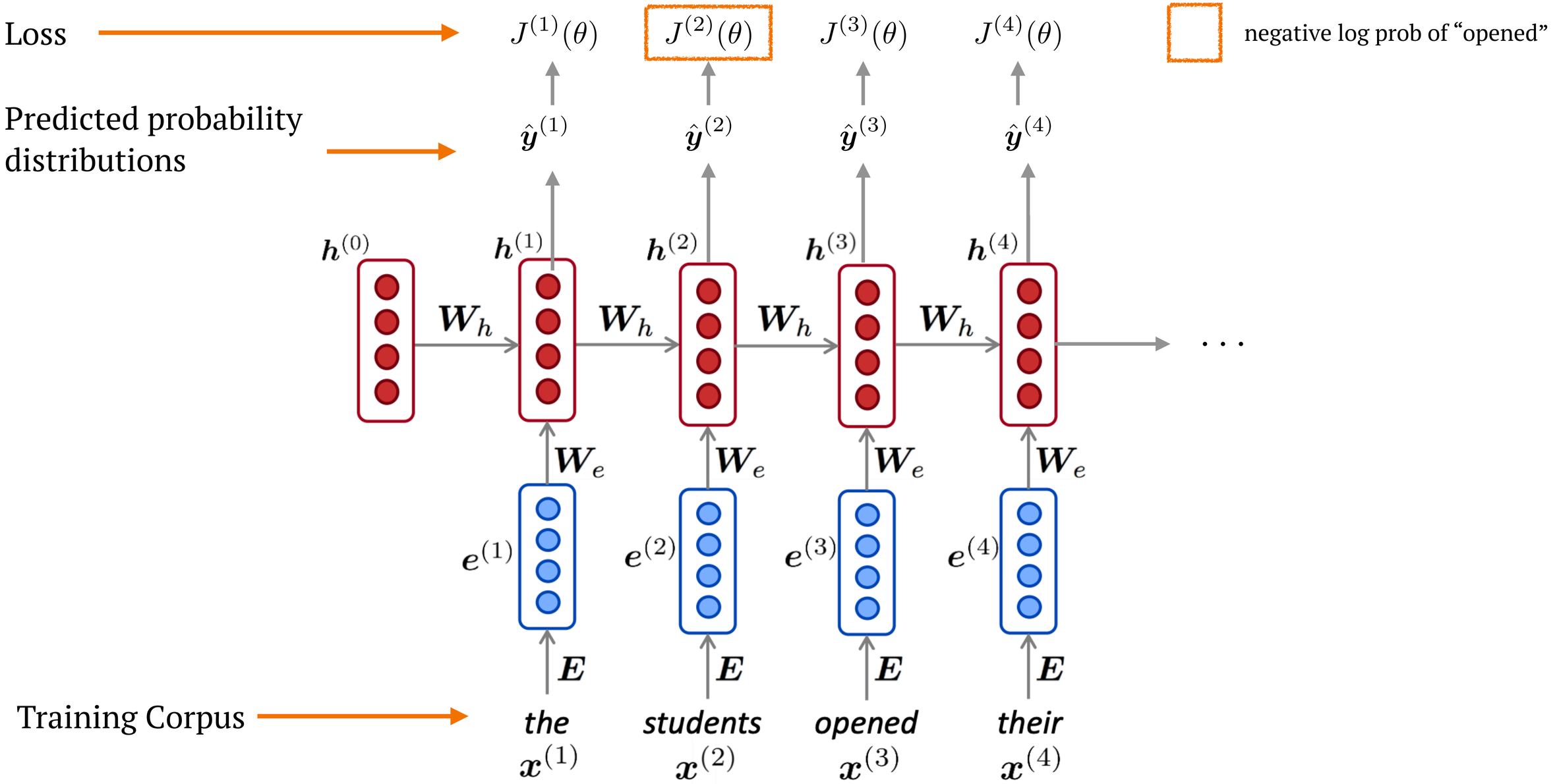
The recurrent function here, takes into consideration all the history!



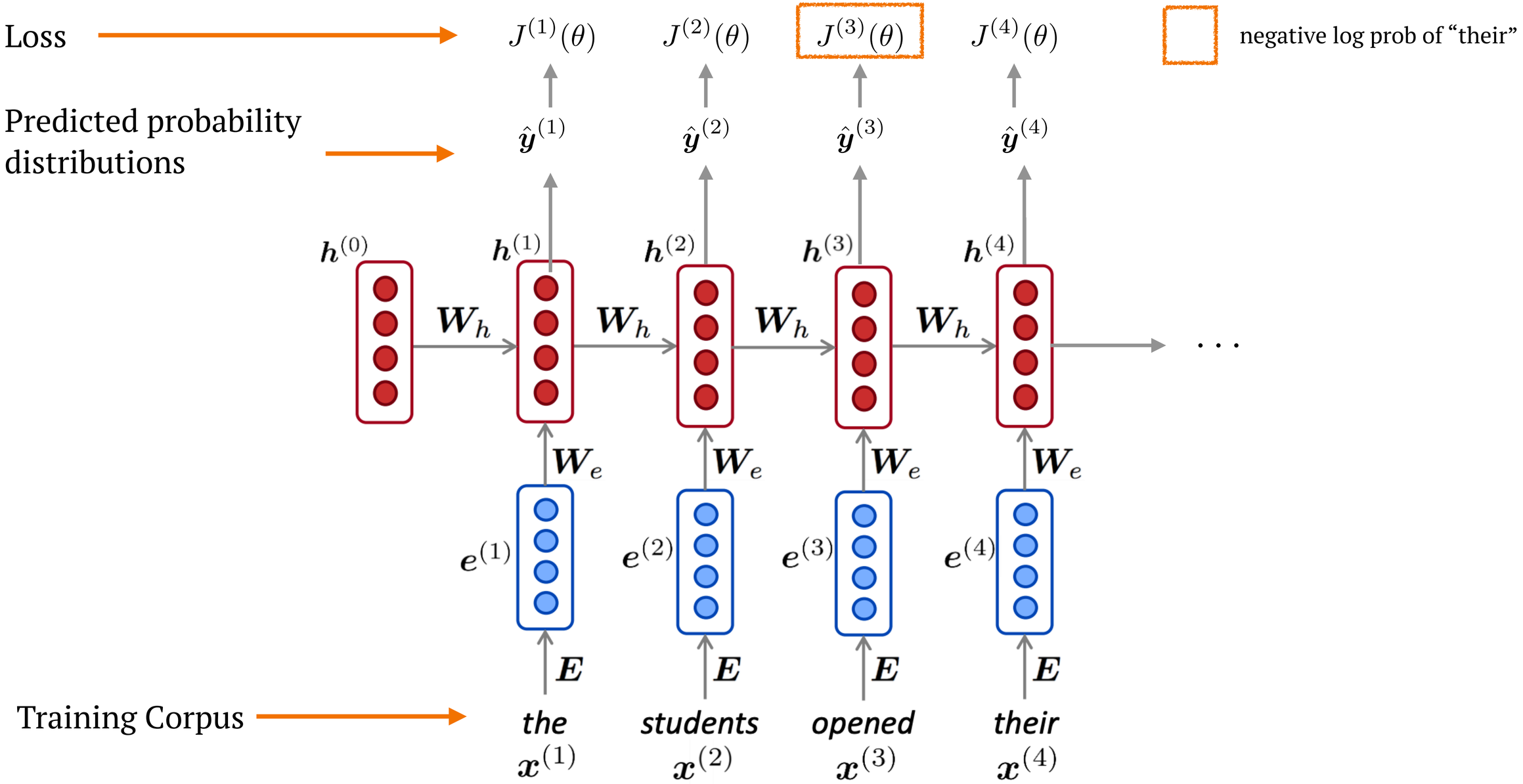
Training a RNN Language Model



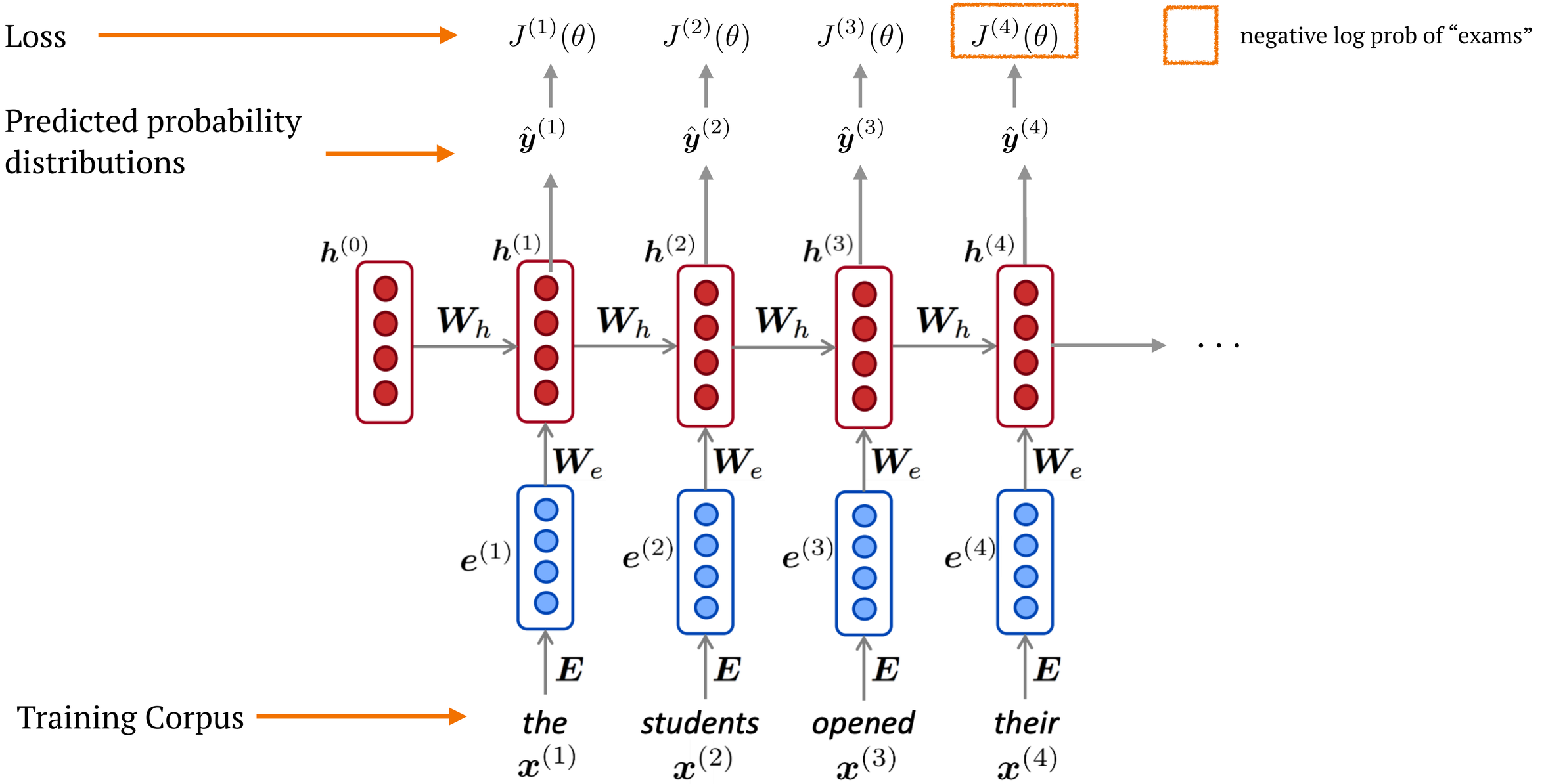
Training a RNN Language Model



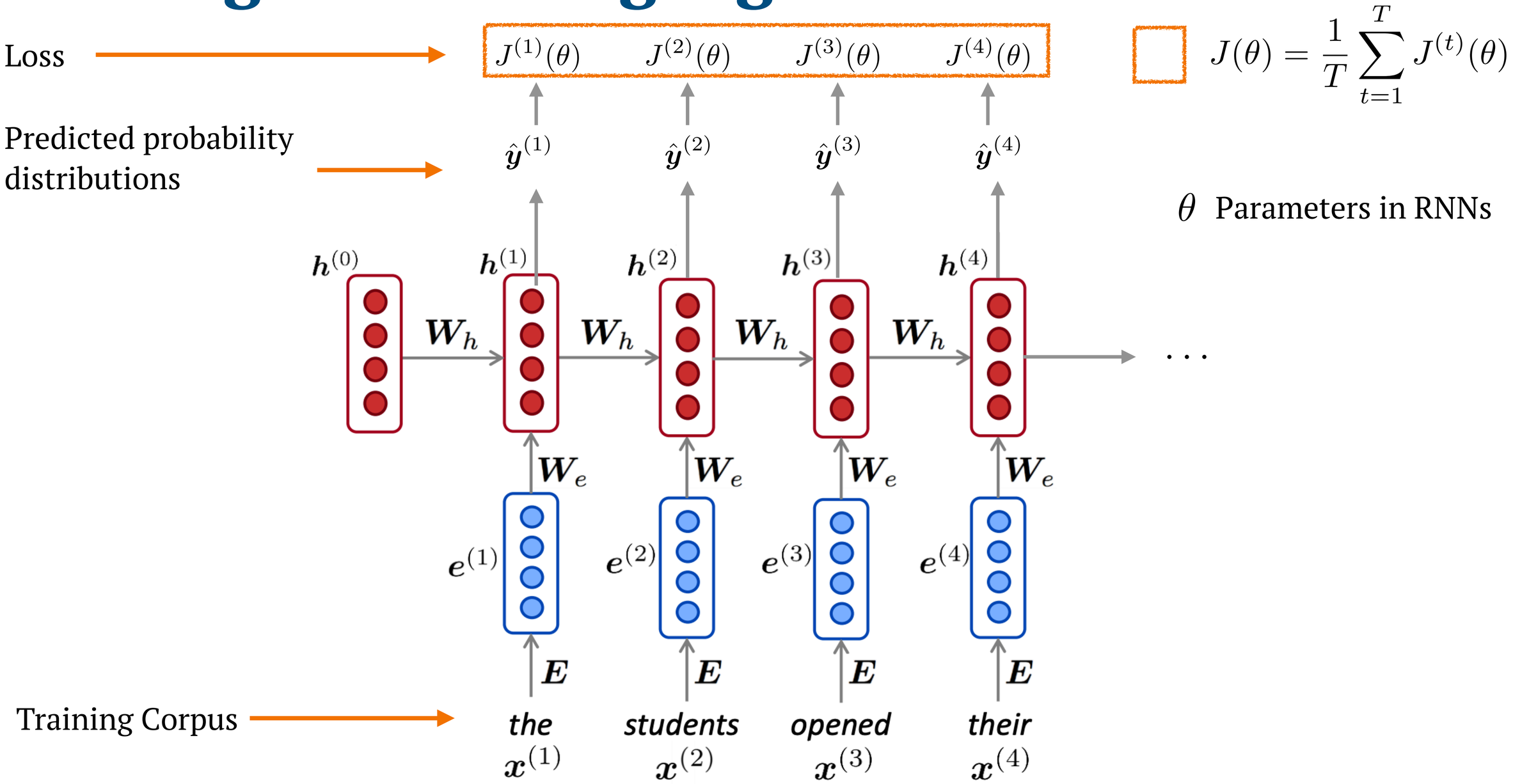
Training a RNN Language Model



Training a RNN Language Model



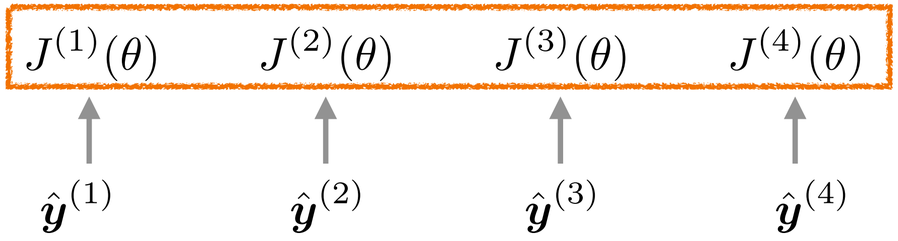
Training a RNN Language Model



GPT-3 Model

Loss

Predicted probability distributions



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

θ Parameters in Transformer

What else can we do?

Assign a probability to a sentence

P(“I am going to school”) > P(“I are going to school”) — Grammar Checking

I had some coffee this morning.

P(“我今早喝了一些咖啡”) > P(“我今早吃了一些咖啡”) — Machine translation

P(“Can we put an elephant into the refrigerator? No, we can’t.”) > P(“Can we put an elephant into the refrigerator? Yes, we can.”) — Question Answering