STEVENS
INSTITUTE OF TECHNOLOGY
1870

Recurrent Neural Networks and LSTM

FA690 Machine Learning in Finance

Dr. Zonghao Yang

2025 Spring

Learning Objective

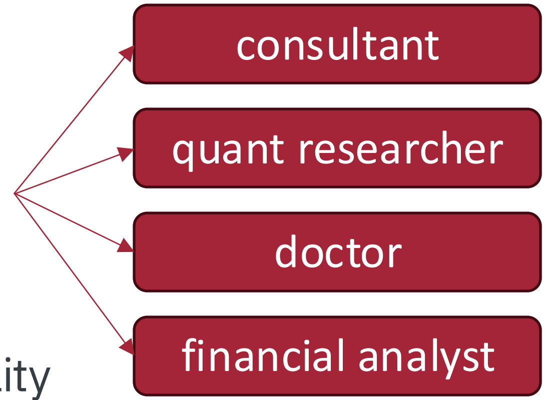
- Understand the fundamental architecture and operation of Recurrent Neural Networks (RNNs) and their application to sequential data processing
- Identify the vanishing gradient problem in vanilla RNNs and explain why it limits their ability to capture long-term dependencies
- Master the Long Short-Term Memory (LSTM) architecture and its key components: cell state, forget gate, input gate, and output gate
- Explore advanced RNN variants including bidirectional and multi-layer RNNs, and their practical applications in language modeling

Language Modeling

Language Modeling

- Language modeling is the task of predicting what word comes next

I study Financial Engineering at Stevens. I want to be a ____



- More formally: A sentence can be considered from the perspective of probability

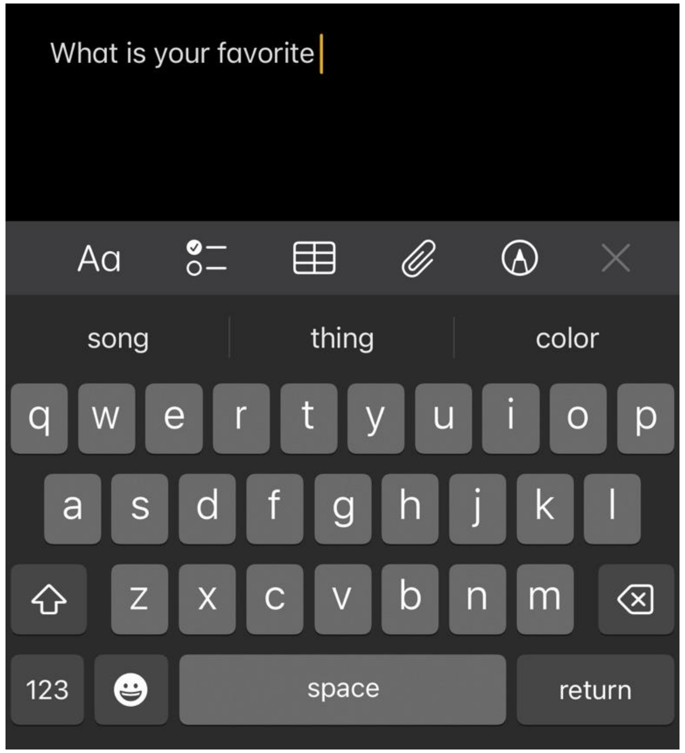
$$P\left(x^{(1)},\dots,x^{(t)}\right) = P\left(x^{(1)}\right) \times P\left(x^{(2)}|x^{(1)}\right) \times \cdots \times P\left(x^{(T)}|x^{(T-1)},\dots,x^{(1)}\right)$$
$$= \prod_{t=1}^{T} P\left(x^{(t)}|x^{(t-1)},\dots,x^{(1)}\right)$$

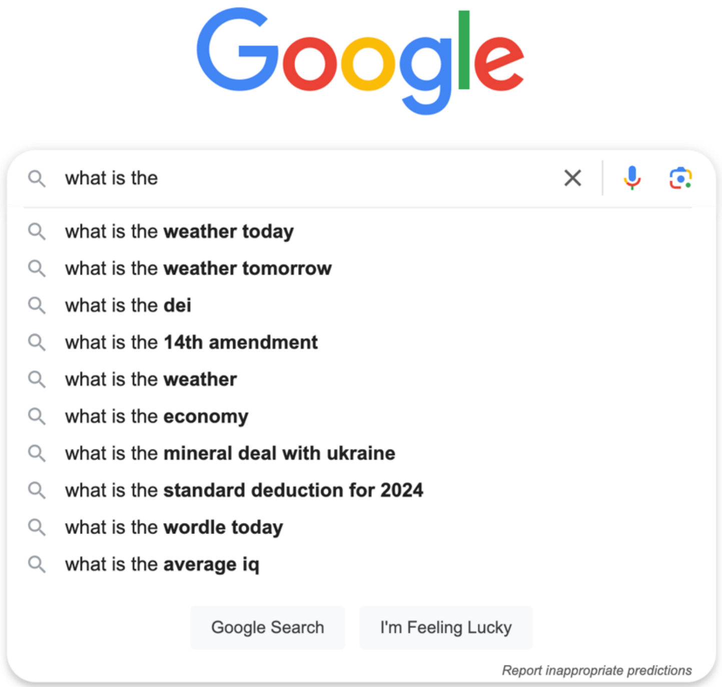
- Language model: Given a sequence of words $x^{(1)}, x^{(2)}, \dots, x^{(t)}$, compute the probability distribution of the next word $x^{(t+1)}$:

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

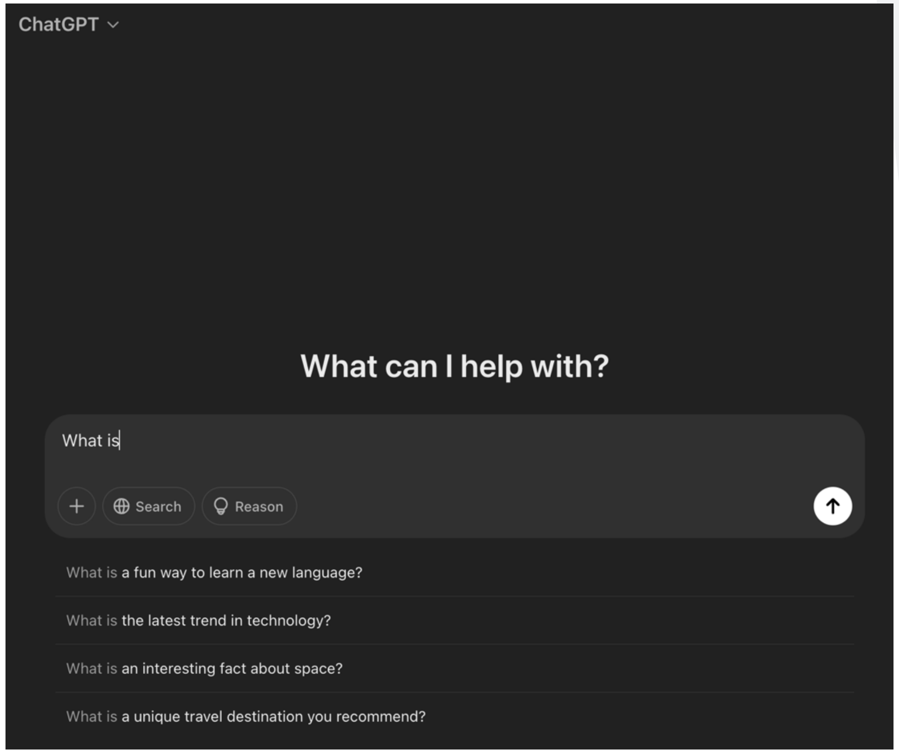
where $x^{(t+1)}$ can be any word in the vocabulary $V = \{w_1,\dots,w_{|V|}\}$

Language Models in Daily Life


What is your favorite


what is the

what is the **weather today**
what is the **weather tomorrow**
what is the **dei**
what is the **14th amendment**
what is the **weather**
what is the **economy**
what is the **mineral deal with ukraine**
what is the **standard deduction for 2024**
what is the **wordle today**
what is the **average iq**



What can I help with?

What is

What is a fun way to learn a new language?
What is the latest trend in technology?
What is an interesting fact about space?
What is a unique travel destination you recommend?

Why Language Modeling?

- Language Modeling is a benchmark task that helps us measure our progress on predicting language use
  - Similar to image classification for computer vision
- Language Modeling is a subcomponent of many NLP tasks, especially those involving generating text or estimating the probability of text:
  - Predictive typing, speech recognition, spelling/grammar correction, machine translation, summarization, etc.
- Everything else in NLP has been rebuilt upon language modeling
  - ChatGPT is a language model

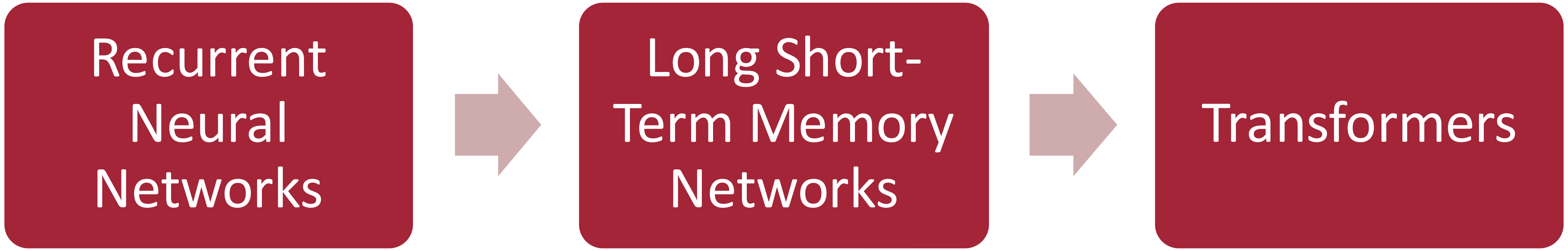
Next-Word Prediction

What can we do with next-word prediction? A sufficiently strong language model can do many things.

- Facts: Stevens Institute of Technology is located in ____, New Jersey.
- Grammar: I put ____ fork down on the table.
- Lexical semantics/topic: I went to the ocean to see the fish, turtles, seals, and ____.
- Sentiment: Overall, the value I got from the two hours watching it was the sum of the popcorn and the drink. The movie was ____.
- Reasoning: Military service during the Vietnam era ____ earnings later in life.
- Arithmetic: I was thinking about the sequence that goes 1, 1, 2, 3, 5, 8, 13, 21, ____.

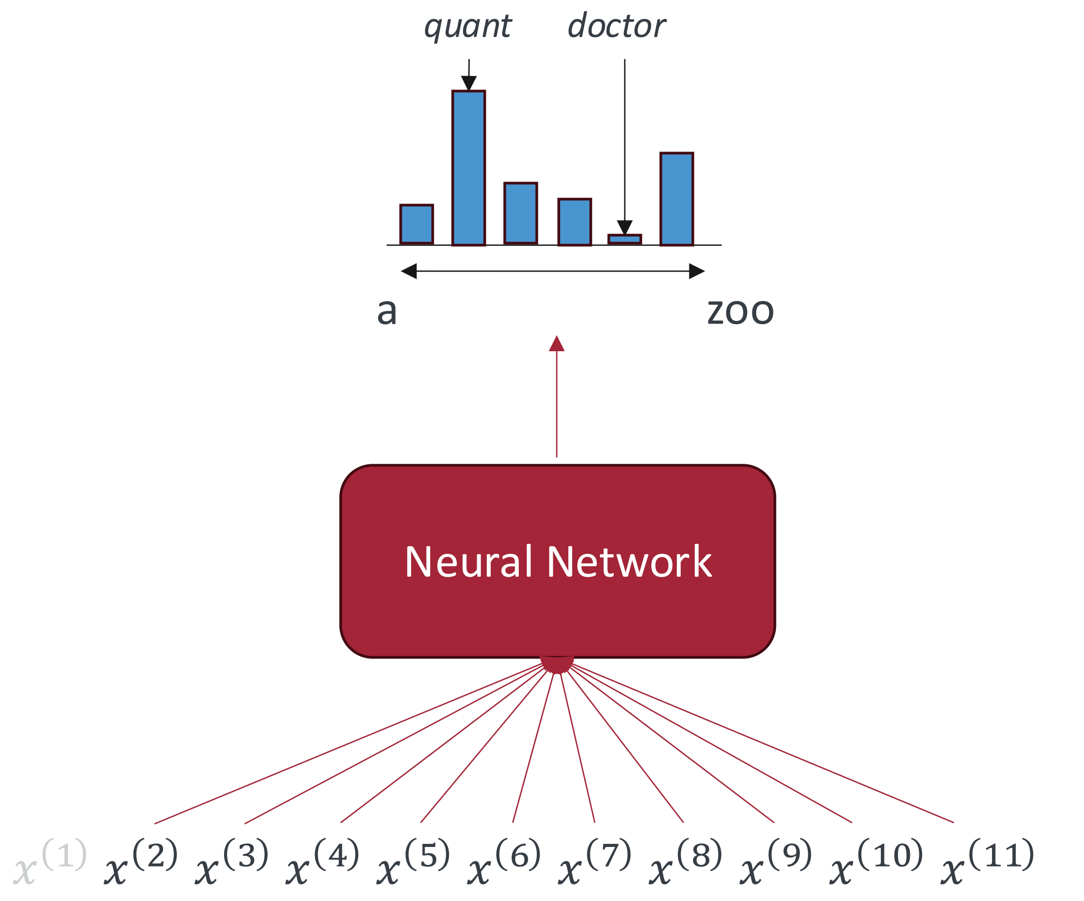
Recurrent Neural Network

Roadmap


Recurrent Neural Networks
Long Short-Term Memory Networks
Transformers

Neural Network

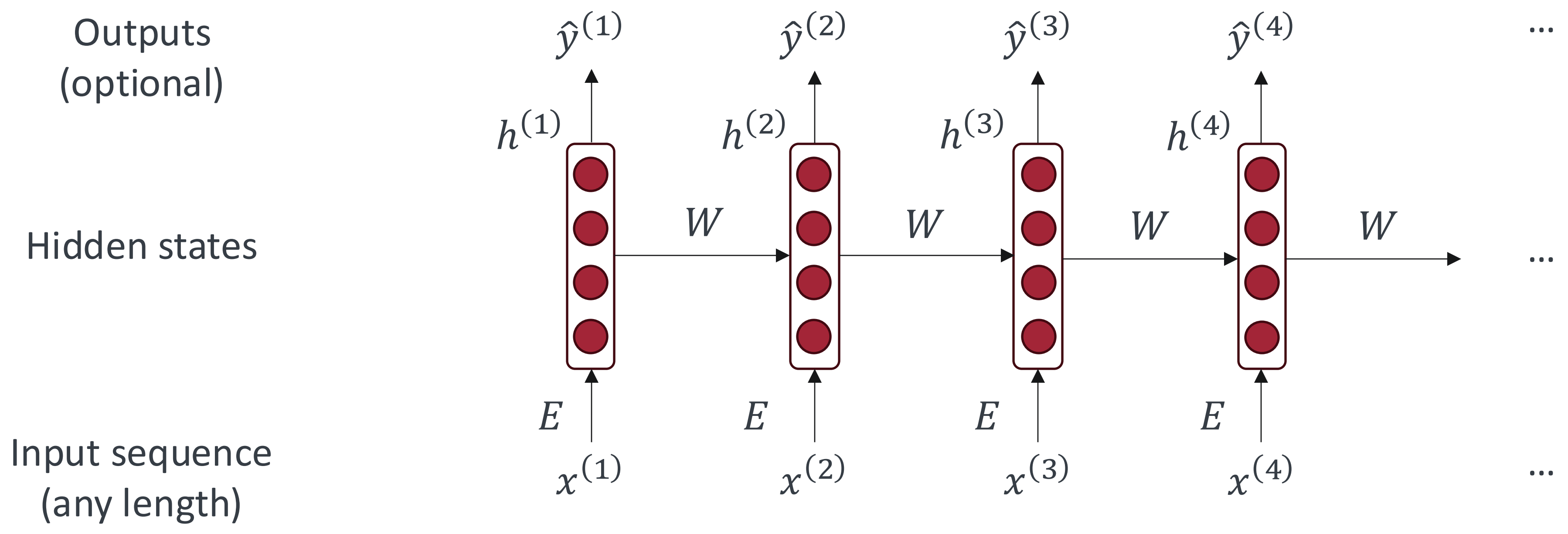
- Neural network approach to language models: Fixed window of inputs (e.g., 10 tokens)
- Limitations:
  - Text has varying length
  - The number of parameters grows exponentially with increasing window length
- Similar to convolutional neural network (CNN) as of images, we need a neural network architecture designed for text input (sequence input with varying length)


quant
doctor
a
zoo
Neural Network
$x^{(1)}$ $x^{(2)}$ $x^{(3)}$ $x^{(4)}$ $x^{(5)}$ $x^{(6)}$ $x^{(7)}$ $x^{(8)}$ $x^{(9)}$ $x^{(10)}$ $x^{(11)}$

I study Financial Engineering at Stevens. I want to be a

Recurrent Neural Network

- The core idea of an RNN is to apply the same weights W repeatedly



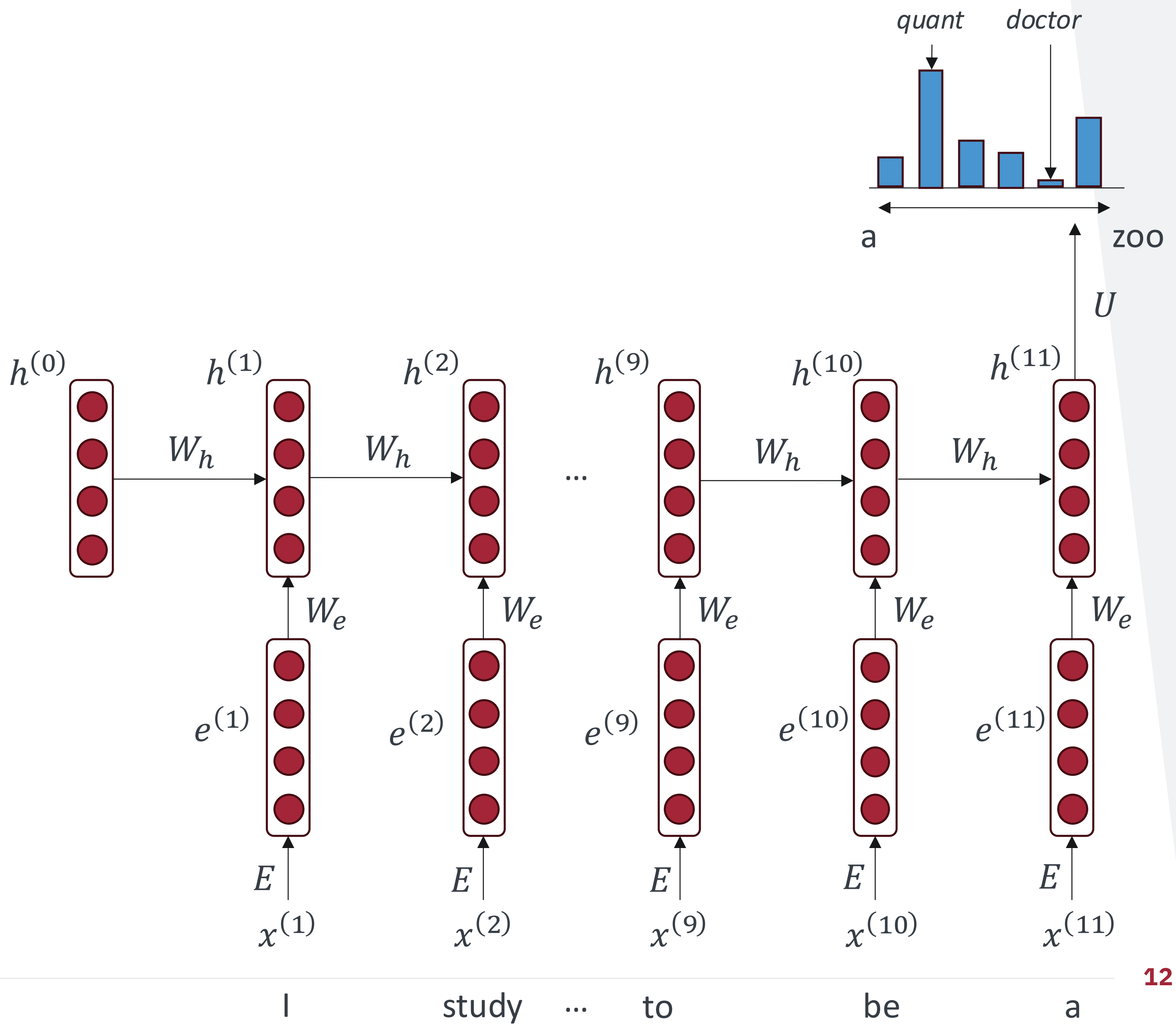
A Vanilla RNN

Example:

I study Financial Engineering at Stevens. I want to be a ____

- Output distribution
  $\hat{y}^{(t)} = \text{softmax}\,(Uh^{(t)} + b_2) \in \mathbb{R}^{|V|}$
- Hidden states: $h^{(0)}$ is the initial hidden state
  $h^{(t)} = \sigma(W_h h^{(t-1)} + W_e e^{(t)} + b_1)$
- Word embeddings
  $e^{(t)} = Ex^{(t)}$
- Words / one-hot vectors
  $x^{(t)} \in \mathbb{R}^{|V|}$

$\hat{y}^{(12)} = P\left(x^{(12)} | x^{(11)}, \dots, x^{(1)}\right)$



A Vanilla RNN

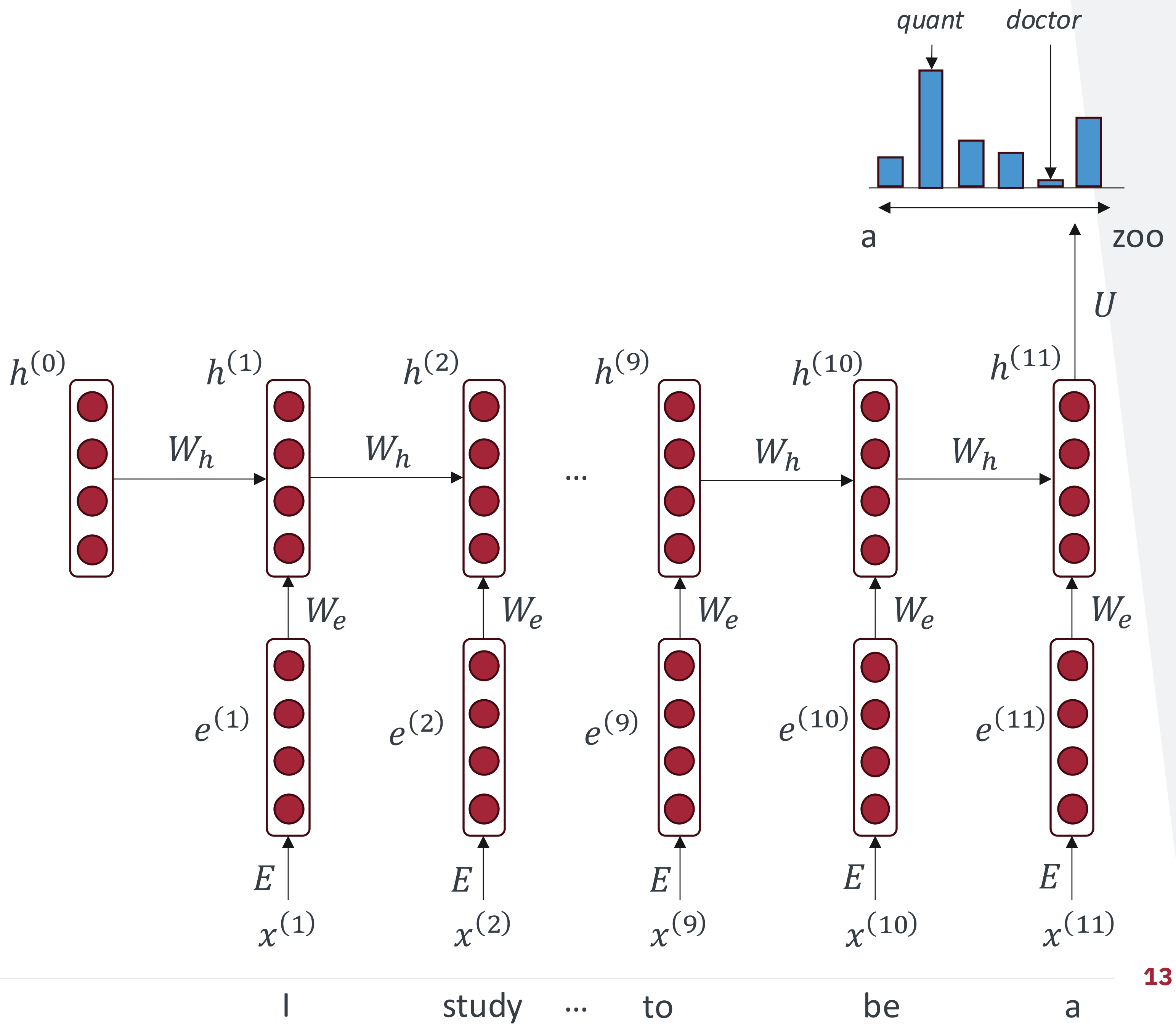
Advantages

- Process input with any length
- Computation for step t can (in theory) use information from many steps back
- Model size doesn't increase for longer input context
- Same weights are applied on every timestep

Disadvantages

- Recurrent computation is slow
- Vanishing gradient: In practice, difficult to access information from many steps back

$$\hat{y}^{(12)} = P\left(x^{(12)} \middle| x^{(11)}, \dots, x^{(1)}\right)$$



Training RNN

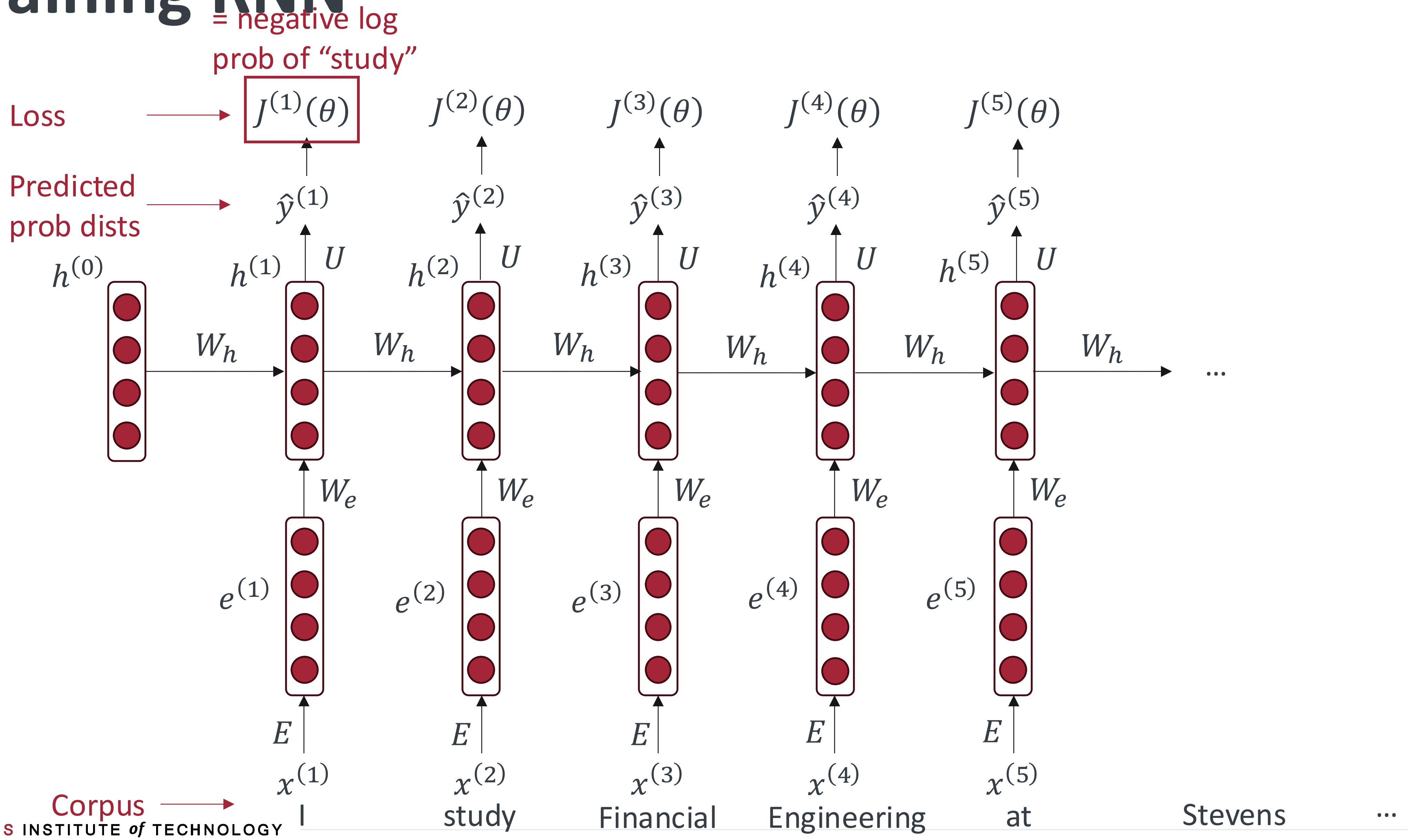
- Get a big corpus of text which is a sequence of words $x^{(1)}, x^{(2)}, \dots, x^{(T)}$
  - The text corpus is fed into an RNN
- Compute output distribution $\hat{y}^{(t)}$ for every step t, i.e., $P\left(x^{(t)} | x^{(t-1)}, \dots, x^{(1)}\right)$
  - That is predict the probability distribution of every word given words so far
- Loss function on step t is cross-entropy between predicted probability distribution $\hat{y}^{(t)}$ and the true next word $y^{(t)}$

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

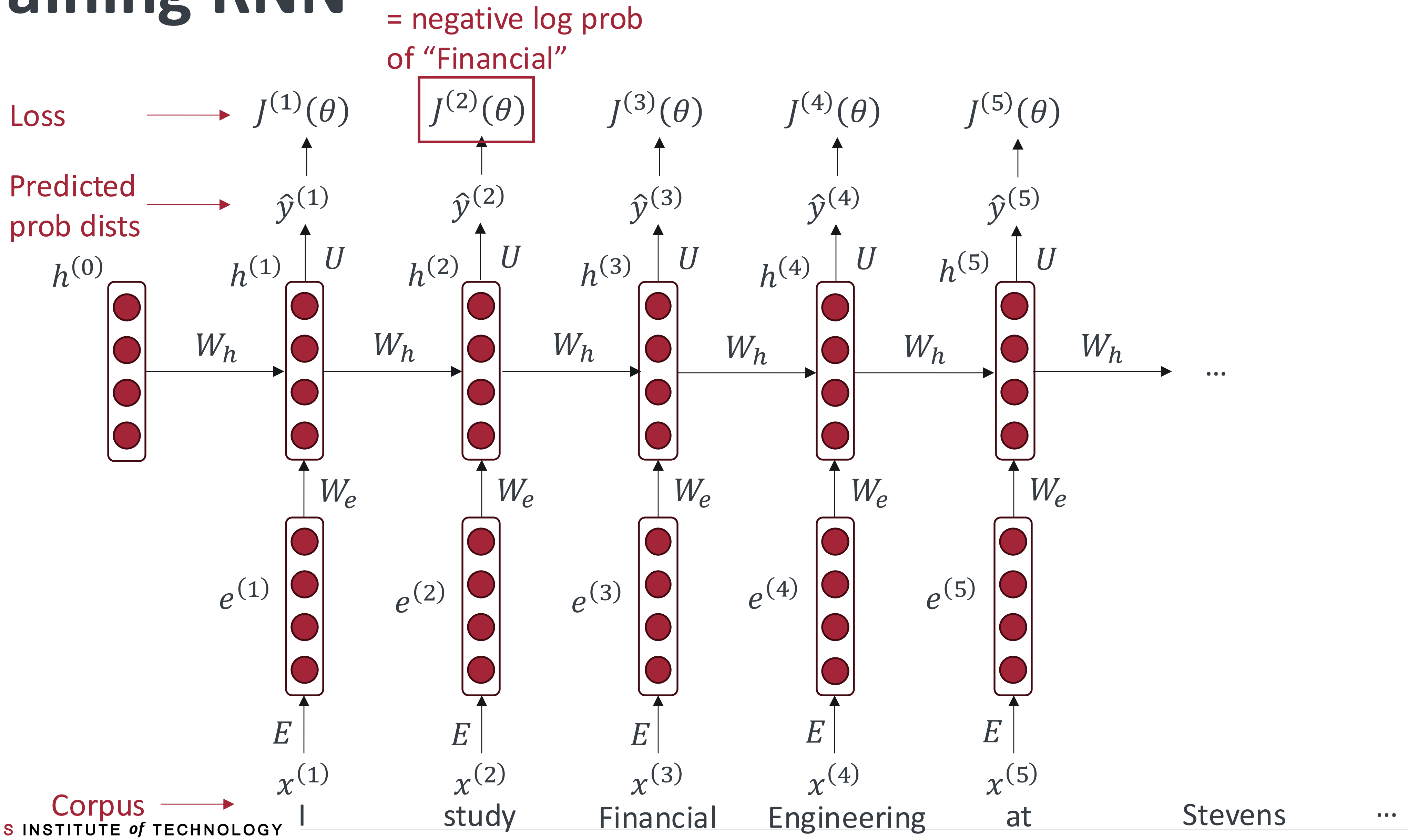
- Average the loss for every step t of the entire training set:

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

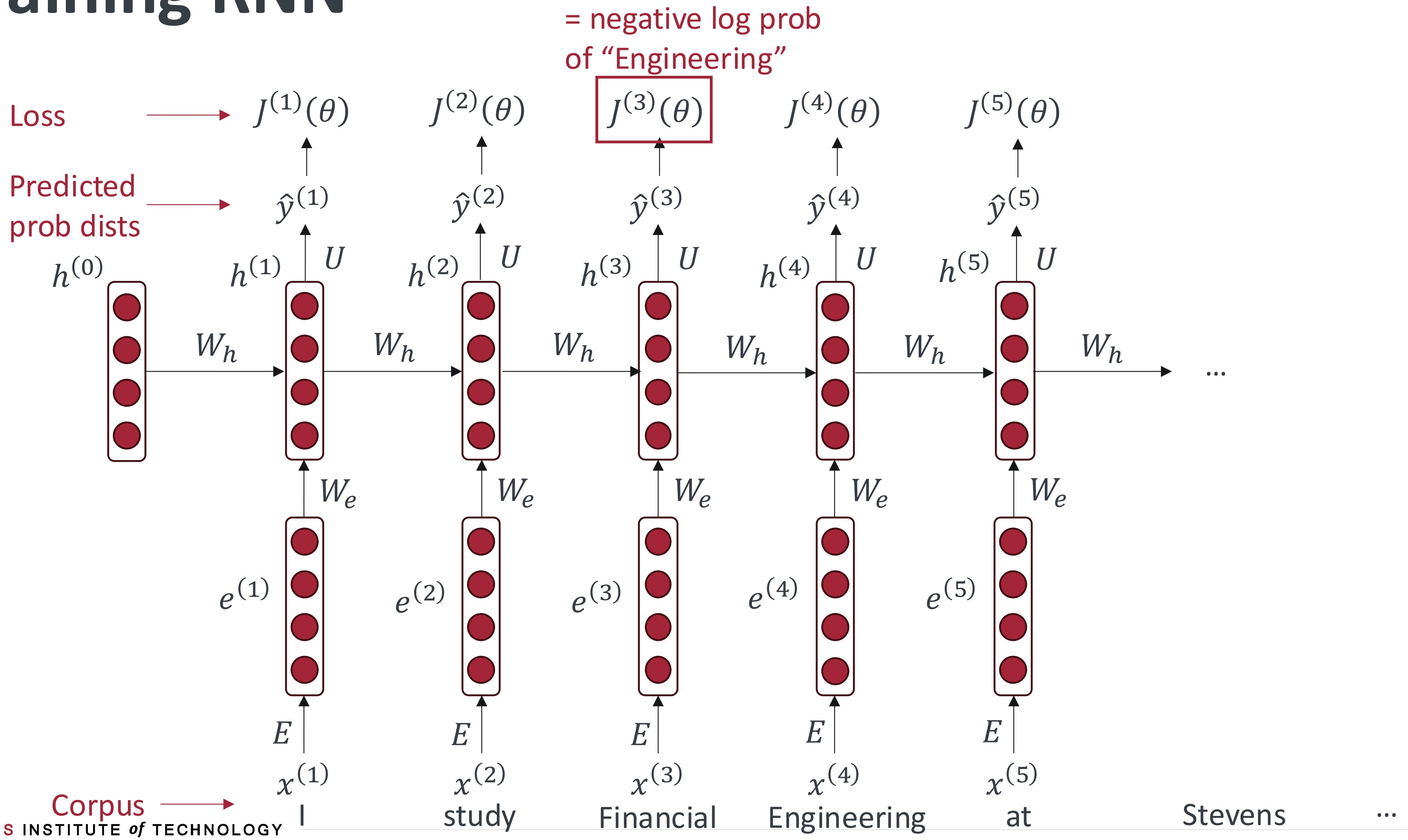
Training RNN



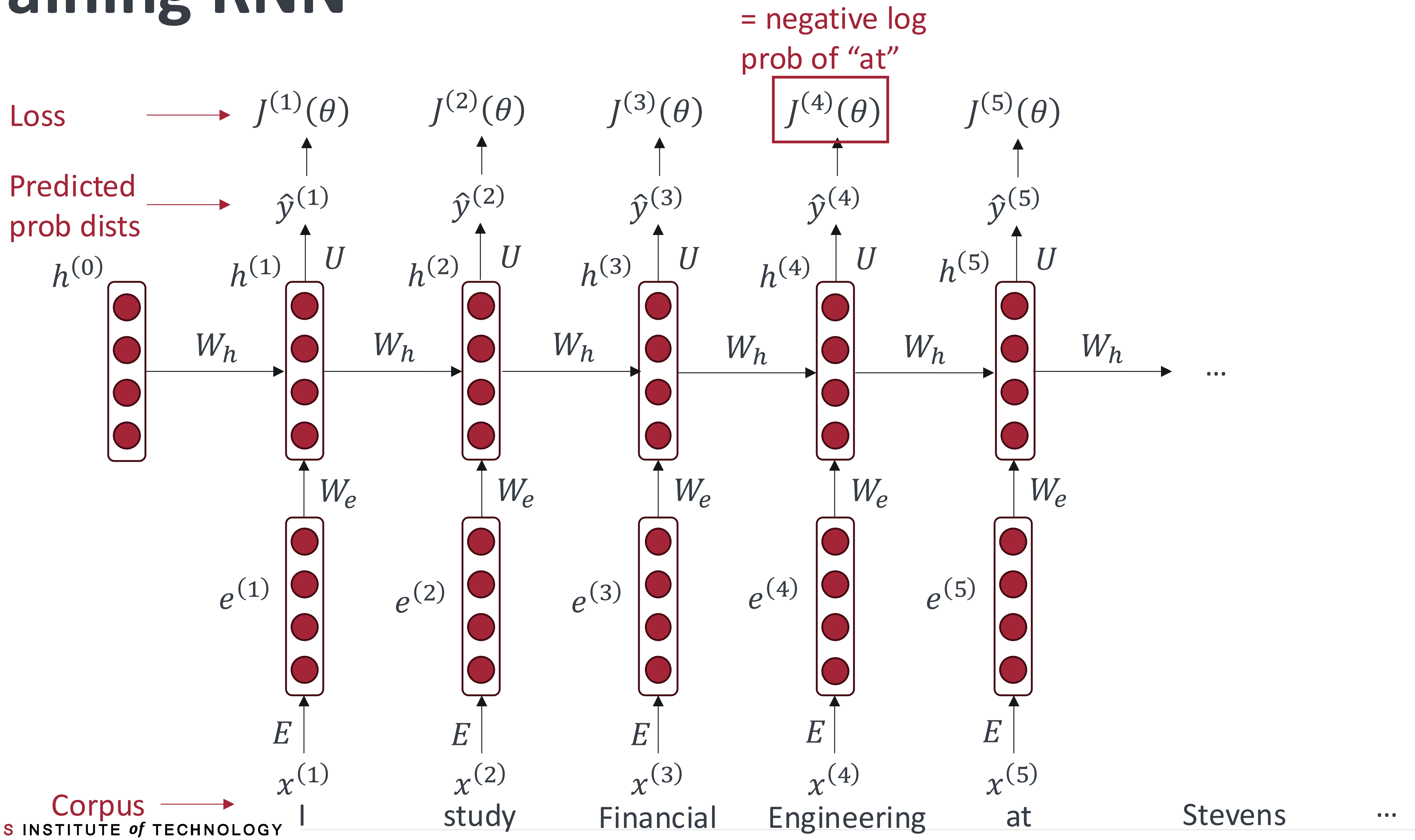
Training RNN



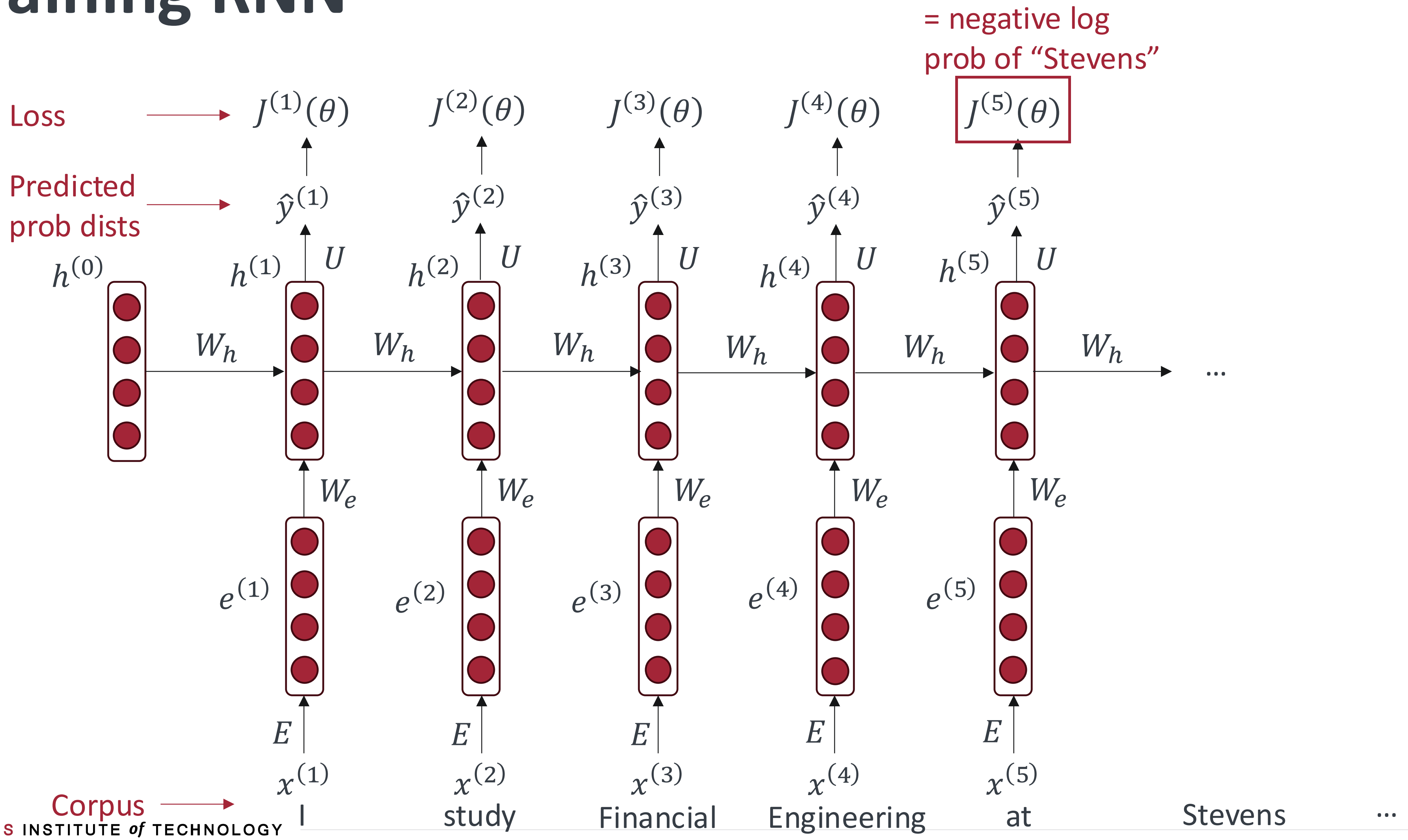
Training RNN


= negative log prob of “Engineering”
Loss
$J^{(1)}(\theta)$ $J^{(2)}(\theta)$ $\boxed{J^{(3)}(\theta)}$ $J^{(4)}(\theta)$ $J^{(5)}(\theta)$
Predicted prob dists
$\hat{y}^{(1)}$ $\hat{y}^{(2)}$ $\hat{y}^{(3)}$ $\hat{y}^{(4)}$ $\hat{y}^{(5)}$
U
$h^{(0)}$ $h^{(1)}$ $h^{(2)}$ $h^{(3)}$ $h^{(4)}$ $h^{(5)}$
W_h
…
W_e
$e^{(1)}$ $e^{(2)}$ $e^{(3)}$ $e^{(4)}$ $e^{(5)}$
E
$x^{(1)}$ $x^{(2)}$ $x^{(3)}$ $x^{(4)}$ $x^{(5)}$
Corpus
I study Financial Engineering at Stevens …

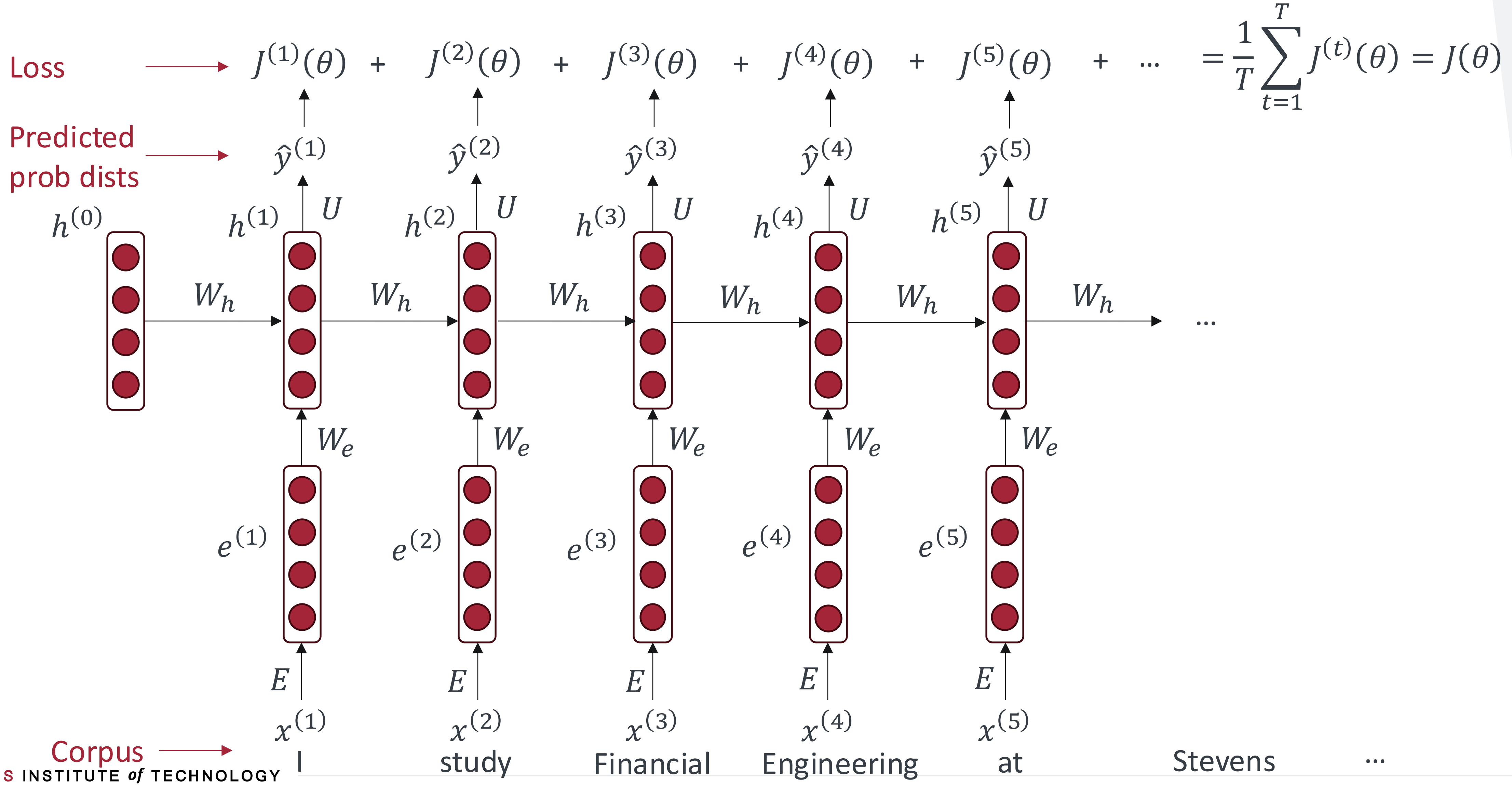
Training RNN



Training RNN



Training RNN



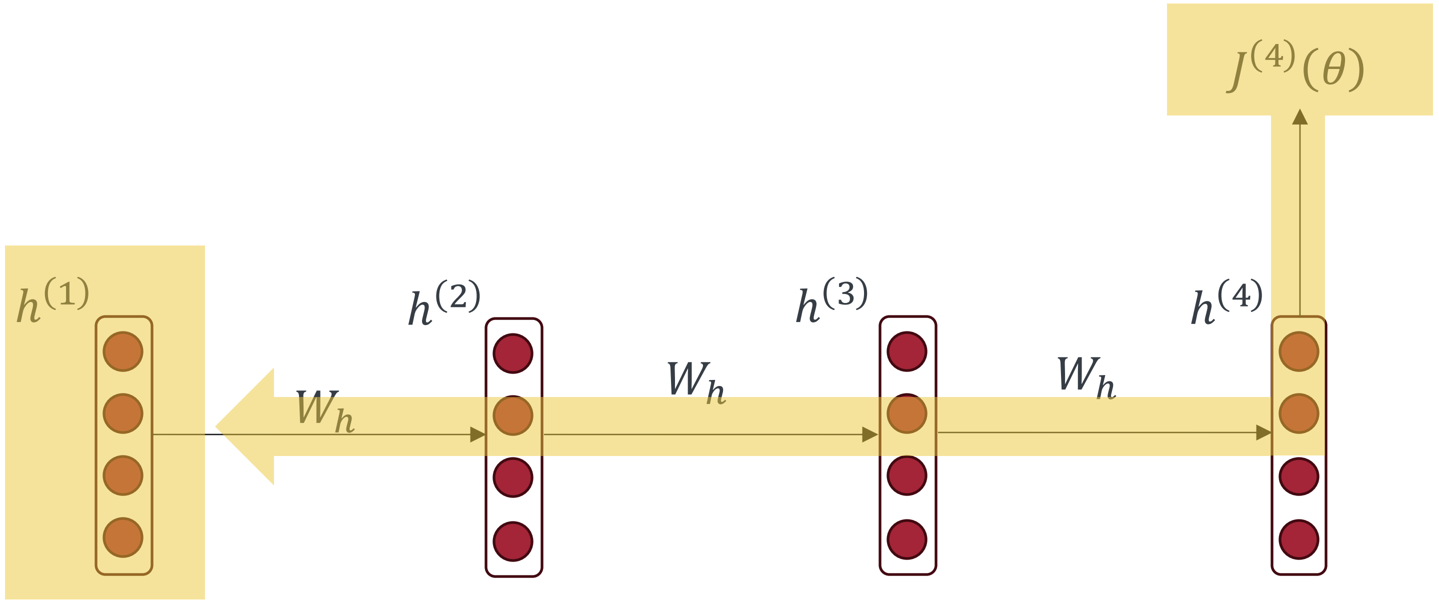
Loss
Predicted prob dists
$J^{(1)}(\theta) + J^{(2)}(\theta) + J^{(3)}(\theta) + J^{(4)}(\theta) + J^{(5)}(\theta) + \dots = \frac{1}{T}\sum_{t=1}^{T} J^{(t)}(\theta) = J(\theta)$
$\hat{y}^{(1)}$ $\hat{y}^{(2)}$ $\hat{y}^{(3)}$ $\hat{y}^{(4)}$ $\hat{y}^{(5)}$
U
$h^{(0)}$ $h^{(1)}$ $h^{(2)}$ $h^{(3)}$ $h^{(4)}$ $h^{(5)}$
W_h
…
W_e
$e^{(1)}$ $e^{(2)}$ $e^{(3)}$ $e^{(4)}$ $e^{(5)}$
E
$x^{(1)}$ $x^{(2)}$ $x^{(3)}$ $x^{(4)}$ $x^{(5)}$
Corpus
I
study
Financial
Engineering
at
Stevens
…

Training RNN

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

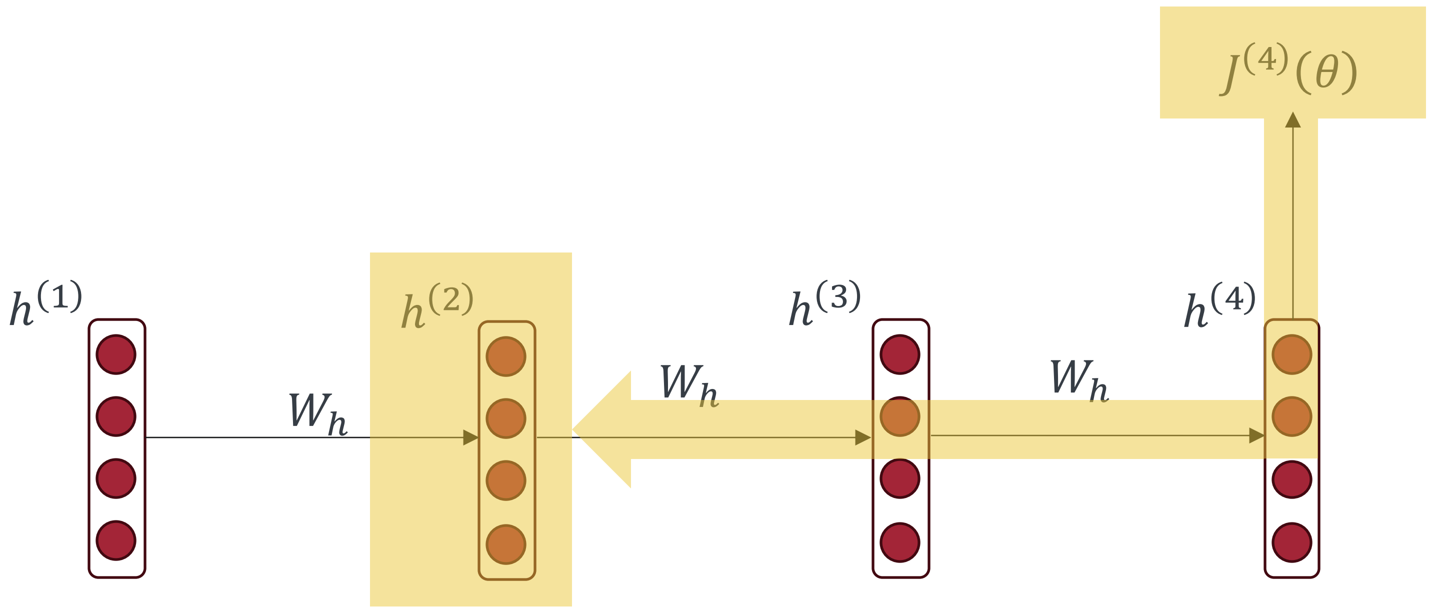
- Solving the optimization problem: Gradient descent and its variants
  - We often compute the loss at the sentence level (or batch of sentences)
  - Compute the gradients
  - Update the weights θ
- The challenge of solving the optimization problem for a vanilla RNN: **Vanishing gradients**

Vanishing Gradient Intuition



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

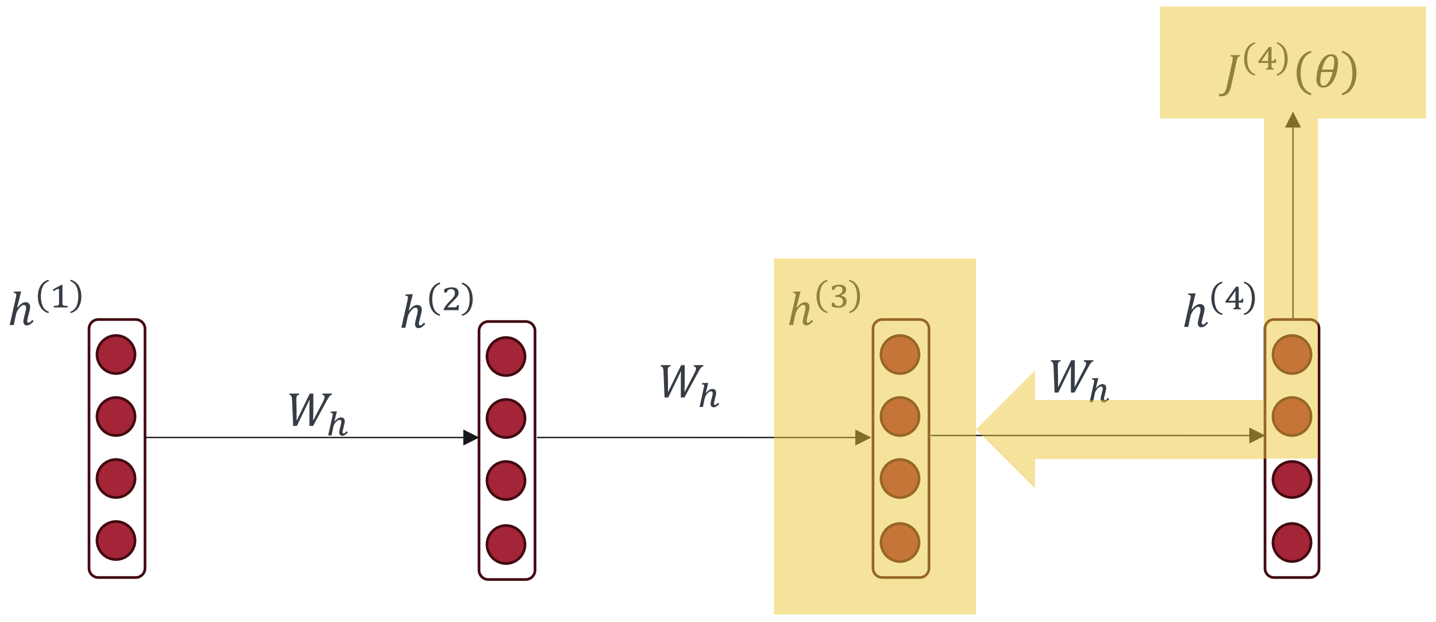
Vanishing Gradient Intuition



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

Chain rule

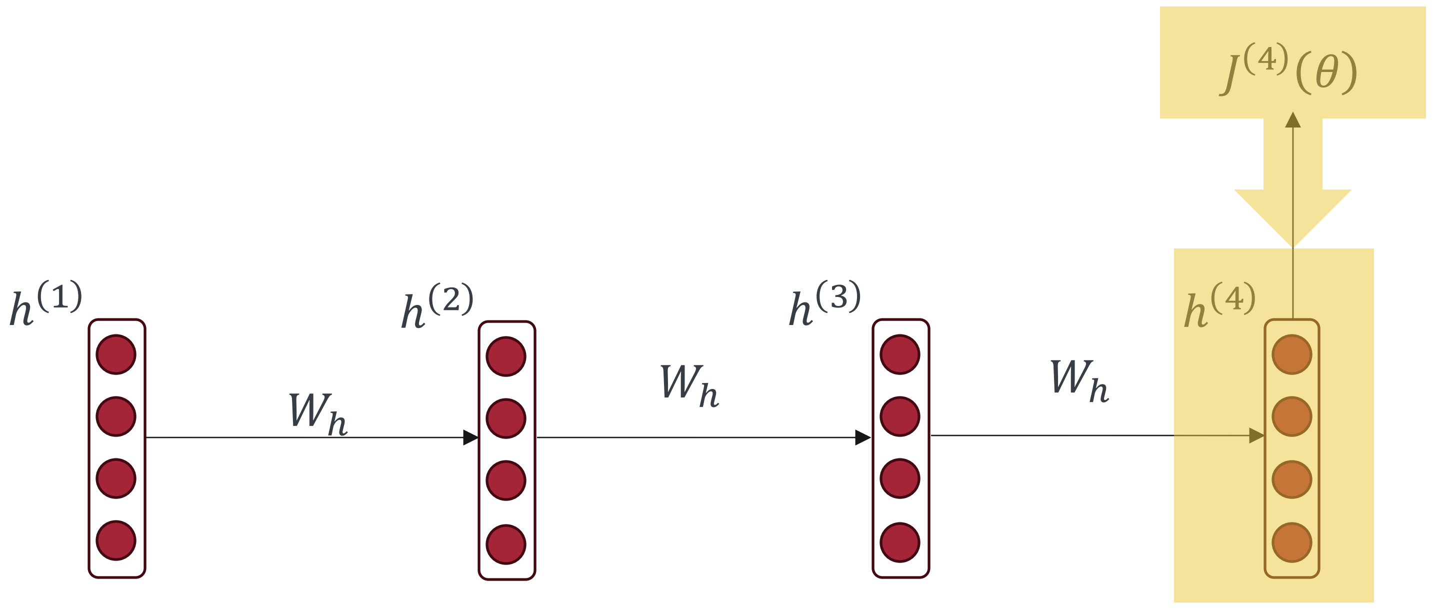
Vanishing Gradient Intuition



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

Chain rule

Vanishing Gradient Intuition



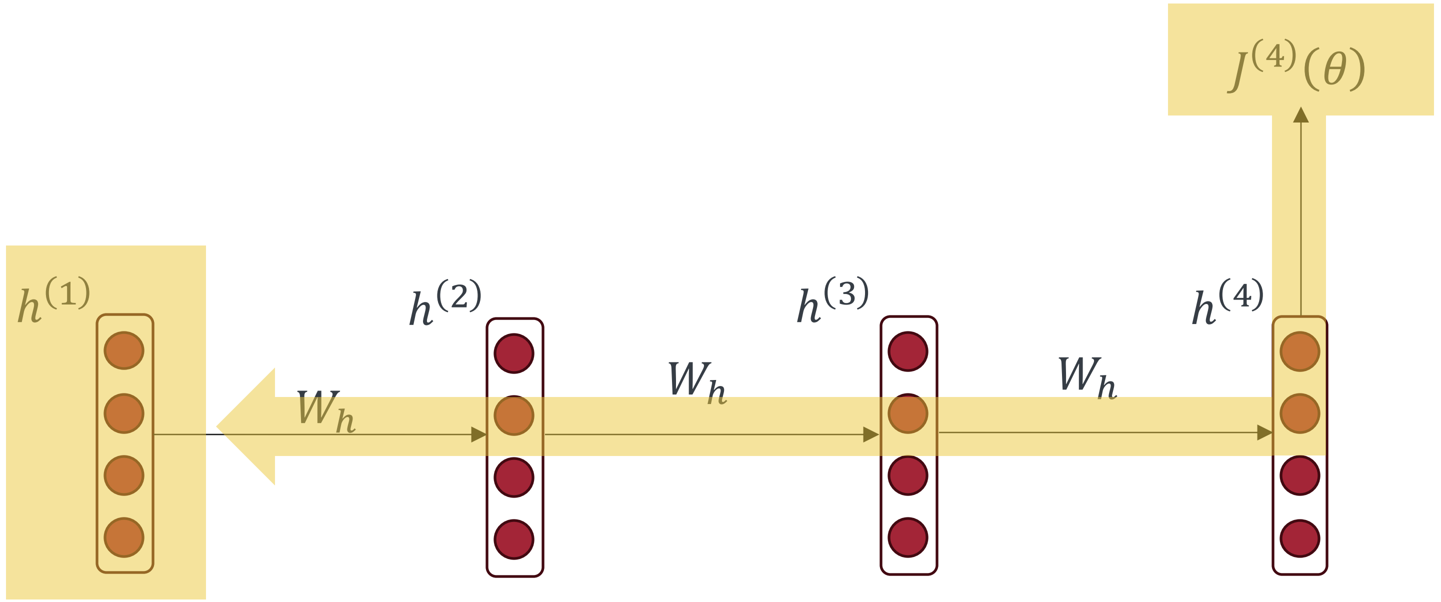
$$\frac{\partial J^{(4)}}{\partial h^{(1)}} = \frac{\partial h^{(2)}}{\partial h^{(1)}} \times \qquad \frac{\partial h^{(3)}}{\partial h^{(2)}} \times \qquad \frac{\partial h^{(4)}}{\partial h^{(3)}} \times \frac{\partial J^{(4)}}{\partial h^{(4)}}$$

Chain rule

Vanishing Gradient Intuition

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

- Vanishing gradient problem: When the gradients in the chain are small (i.e., $\frac{\partial h^{(2)}}{\partial h^{(1)}}, \frac{\partial h^{(3)}}{\partial h^{(2)}}, \frac{\partial h^{(4)}}{\partial h^{(3)}}$), the gradient signal gets smaller and smaller as it backpropagates further
- Why is it a problem
  - Gradient signal from far away is lost because it's much smaller than gradient signal from close-by
  - The model weights are updated only with respect to near effects, not long-term effects



Effects of Vanishing Gradient on RNN

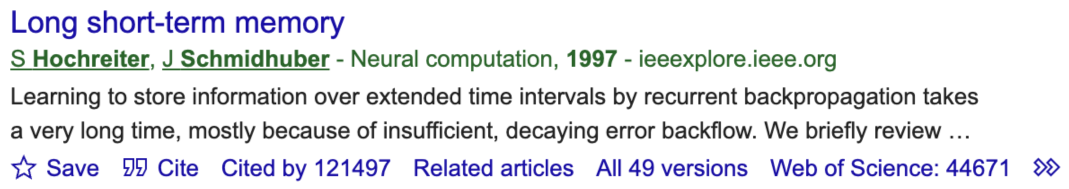
- LM task: When she tried to print her tickets, she found that the printer was out of toner. She went to the stationery store to buy more toner. It was very overpriced. After installing the toner into the printer, she finally printed her ____.
- To learn from this training example, the RNN-LM needs to model the dependency between “tickets” on the 7th step and the target word “tickets” at the end
- If the gradient is small, the model can’t learn this dependency
  - The model is unable to predict similar long-distance dependencies at test time

Fix of the Vanishing Gradient Problem

- The main problem is that it’s too difficult for the RNN to learn to preserve information over many timesteps
- In a vanilla RNN, the hidden state is constantly being rewritten
$$h^{(t)} = \sigma(W_h h^{(t-1)} + W_x x^{(t)} + b)$$
- Long Short-Term Memory: An RNN architecture designed to store both long- and short-term memories (this lecture)
- Attention: The attention mechanism allows the model to focus on specific parts of the input sequence when making predictions, rather than relying solely on the final hidden state of the RNN (next lecture)

Long Short-Term Memory and Other RNNs

Long Short-Term Memory (LS


Long short-term memory

S Hochreiter, J Schmidhuber - Neural computation, 1997 - ieeexplore.ieee.org

Learning to store information over extended time intervals by recurrent backpropagation takes a very long time, mostly because of insufficient, decaying error backflow. We briefly review …

Save Cite Cited by 121497 Related articles All 49 versions Web of Science: 44671

- LSTM is a type of RNN proposed by Hochreiter and Schmidhuber in 1997 as a solution to the problem of vanishing gradients
- On step t, there is a hidden state $h^{(t)}$ and a cell state $c^{(t)}$
  - Both are vectors of length n
  - Cell state: The core component of an LSTM that acts as a "memory"
  - Hidden state: The output of the LSTM at each time step; i.e., the information that is passed to the next time step and is also used to make predictions
- The selection of which information is erased/written/read is controlled by three corresponding gates
  - The gates are also vectors of length n
  - On each time step, each element of the gates can be open (1), closed (0), or somewhere in-between
  - The gates are dynamic in the sense that their value is computed based on the current context

Long Short-Term Memory (LSTM)

- We have a sequence of inputs $x^{(t)}$, and we will compute a sequence of hidden states $h^{(t)}$ and cell states $c^{(t)}$
- Three gates of LSTM
  - Forget gate: Controls what is kept vs forgotten, from previous cell state
  - Input gate: Controls what parts of the new cell content are written to cell
  - Output gate: Controls what parts of the cell are output to hidden state
- Information transfer
  - New cell content: The new content to be written to the cell
  - Cell state: Erase ("forget") content from last cell state, and write ("input") new cell content
  - Hidden state: Read ("output") content from the cell

Sigmoid function: All gate values are between 0 and 1

$$f^{(t)} = \sigma(W_f h^{(t-1)} + U_f x^{(t)} + b_f)$$

$$i^{(t)} = \sigma(W_i h^{(t-1)} + U_i x^{(t)} + b_i)$$

$$o^{(t)} = \sigma(W_o h^{(t-1)} + U_o x^{(t)} + b_o)$$

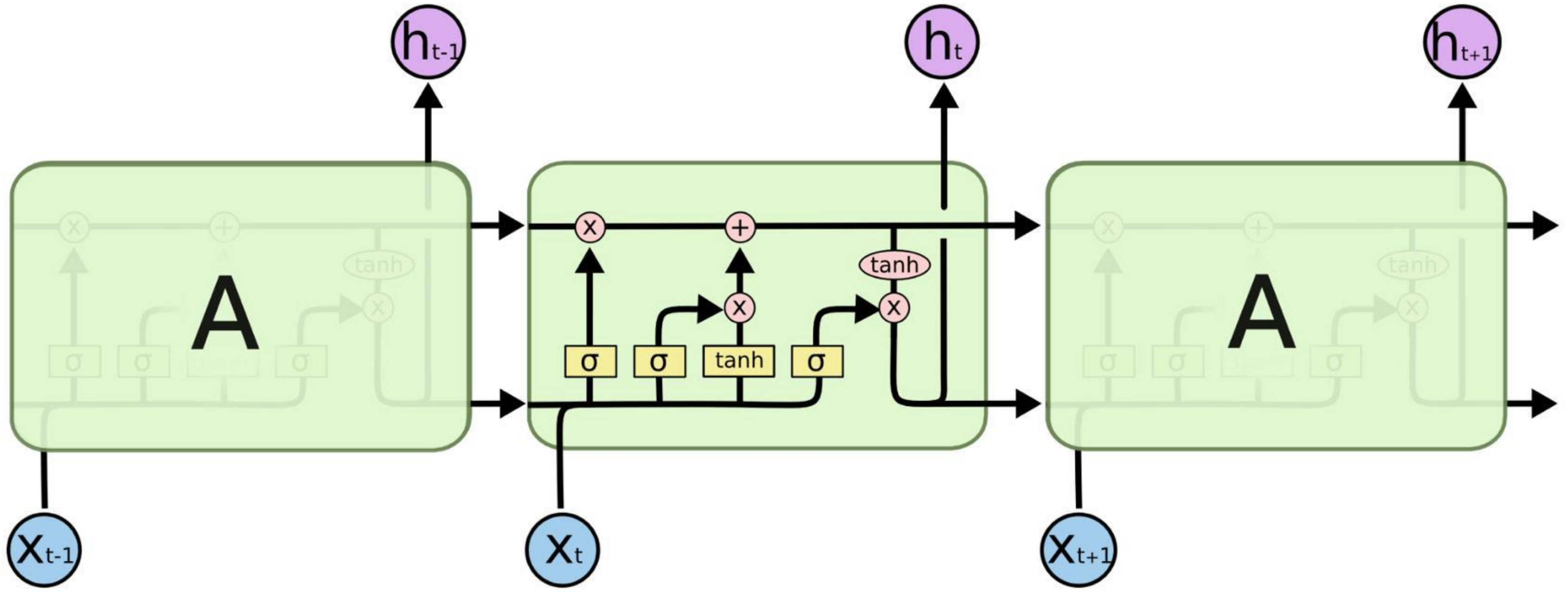
$$\tilde{c}^{(t)} = \tanh(W_c h^{(t-1)} + U_c x^{(t)} + b_c)$$

$$c^{(t)} = f^{(t)} \circ c^{(t-1)} + i^{(t)} \circ \tilde{c}^{(t)}$$

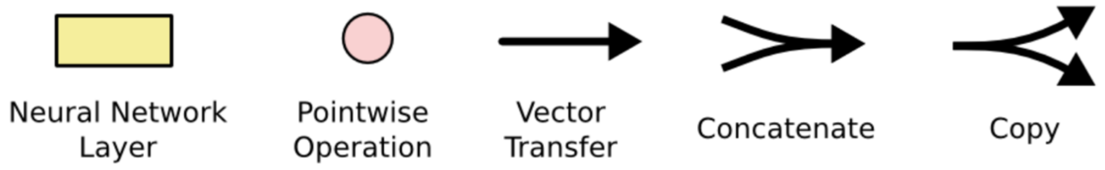
$$h^{(t)} = o^{(t)} \circ \tanh c^{(t)}$$

Element-wise product

LSTM

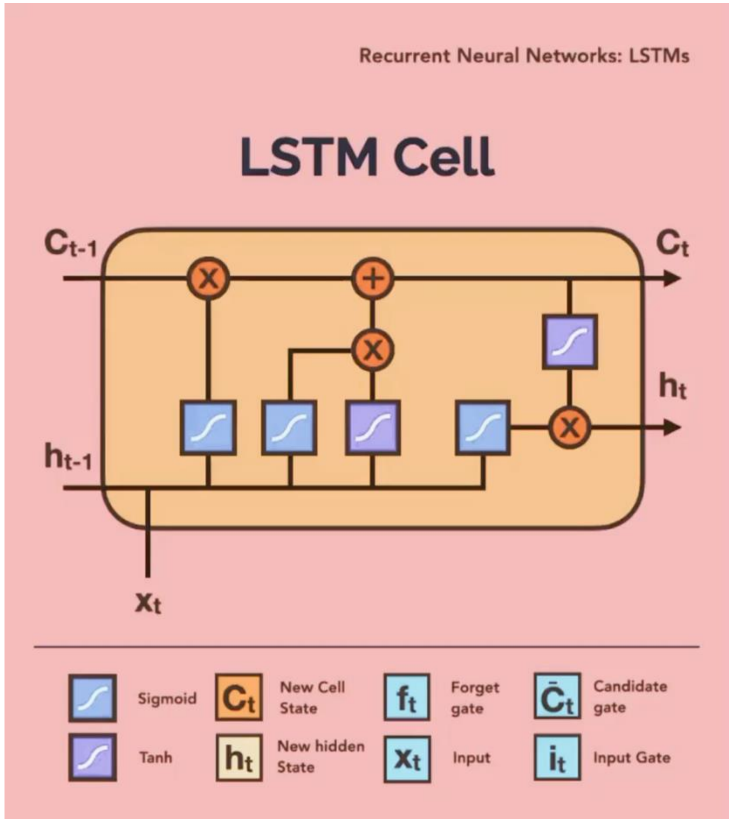


h_{t-1}
h_t
h_{t+1}
A
A
σ
σ
tanh
σ
tanh
X_{t-1}
X_t
X_{t+1}

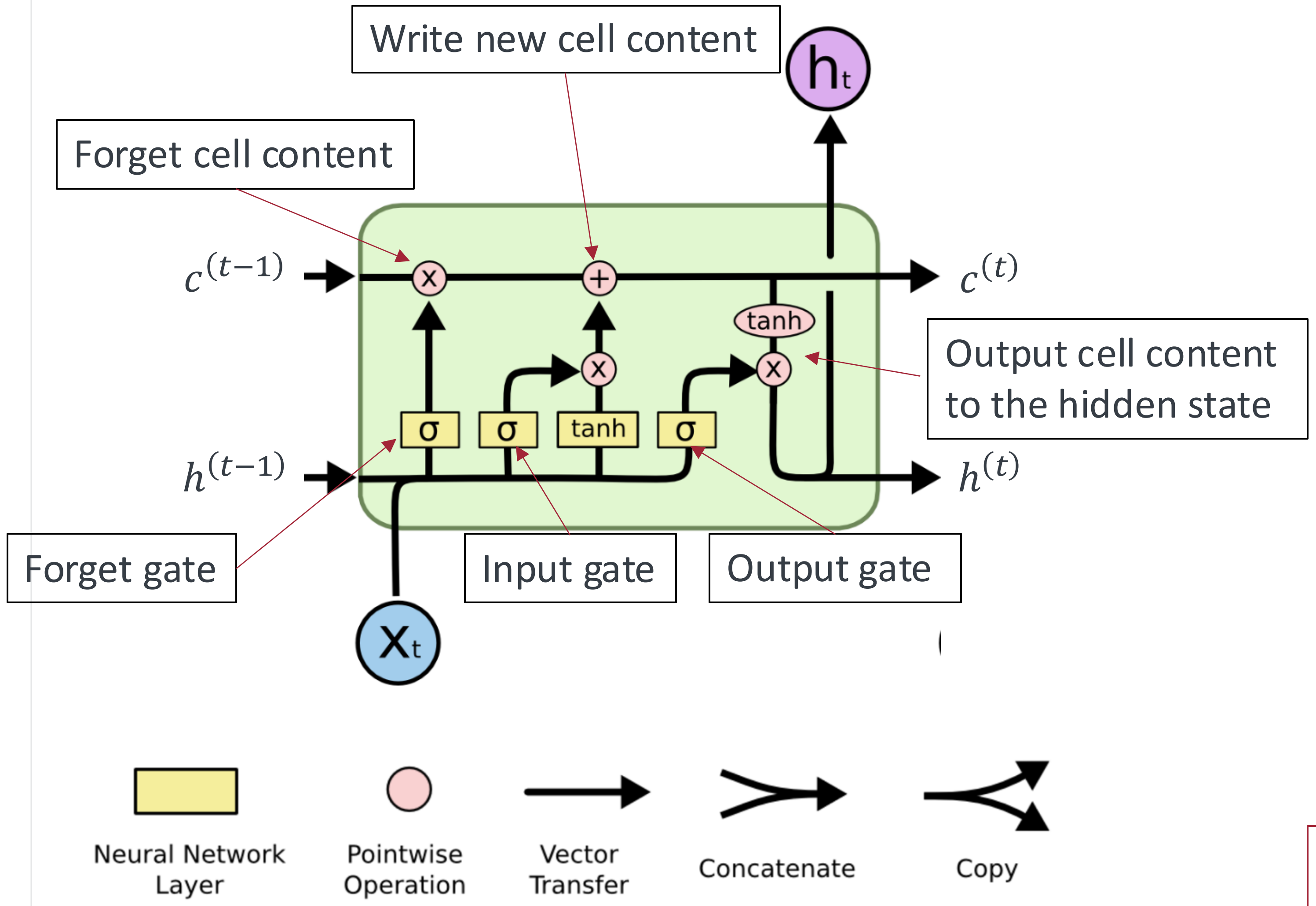


Neural Network Layer
Pointwise Operation
Vector Transfer
Concatenate
Copy

LSTM


Recurrent Neural Networks: LSTMs
LSTM Cell
C_{t-1}
C_t
h_t
h_{t-1}
x_t
Sigmoid
Tanh
C_t New Cell State
h_t New hidden State
f_t Forget gate
x_t Input
$\tilde{C}_t$ Candidate gate
i_t Input Gate

LSTM



$$f^{(t)} = \sigma(W_f h^{(t-1)} + U_f x^{(t)} + b_f)$$

$$i^{(t)} = \sigma(W_i h^{(t-1)} + U_i x^{(t)} + b_i)$$

$$o^{(t)} = \sigma(W_o h^{(t-1)} + U_o x^{(t)} + b_o)$$

$$\tilde{c}^{(t)} = \tanh(W_c h^{(t-1)} + U_c x^{(t)} + b_c)$$

$$c^{(t)} = f^{(t)} \circ c^{(t-1)} + i^{(t)} \circ \tilde{c}^{(t)}$$

$$h^{(t)} = o^{(t)} \circ \tanh c^{(t)}$$

Reference: Understanding LSTM Networks, colah's blog [Link]

Final Words on LSTM

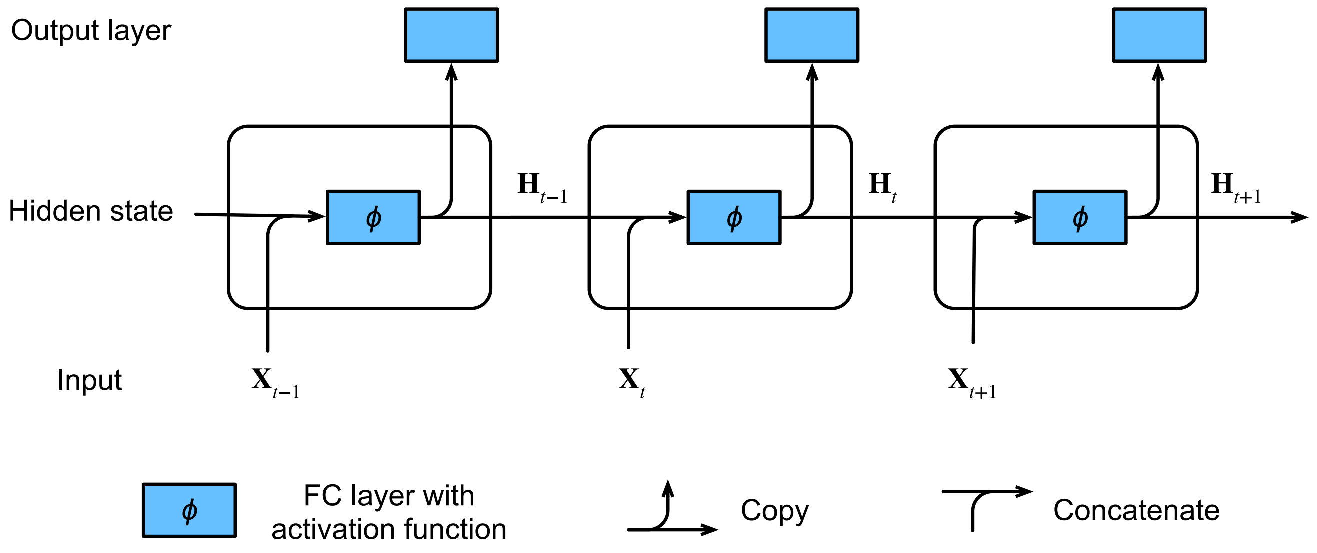
- The LSTM architecture makes it much easier for an RNN to preserve information over many timesteps
  - e.g., if the forget gate is set to 1 for a cell dimension and the input gate set to 0, then the information of that cell is preserved indefinitely.
  - In contrast, it's harder for a vanilla RNN to learn a recurrent weight matrix W_h that preserves info in the hidden state
- LSTM was the dominant approach for most NLP tasks in 2013 – 2015
  - Successful tasks include handwriting recognition, speech recognition, machine translation, parsing, and image captioning, as well as language models
- Recently (2019 – Present), Transformers have become dominant for all tasks

Question

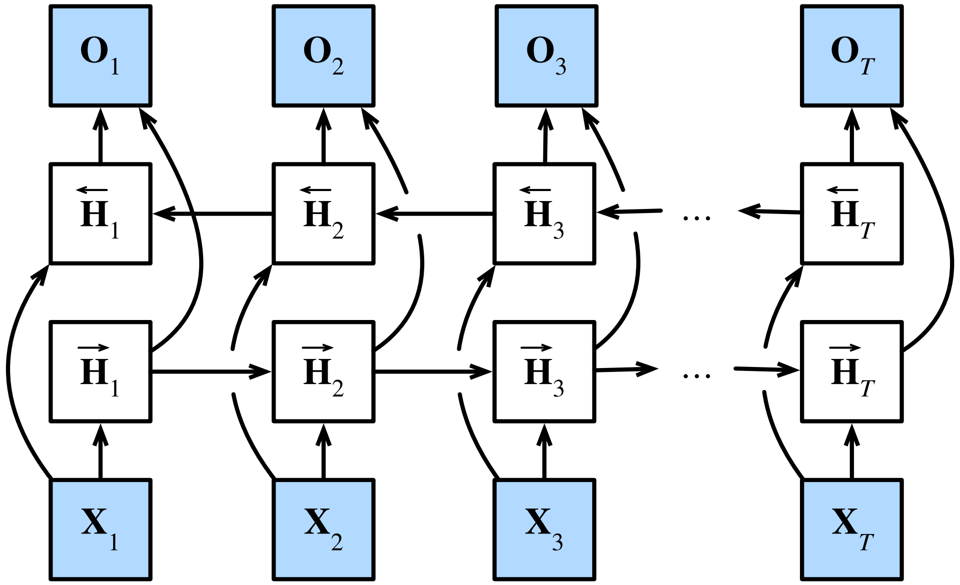
- Which one of the following is the right spell-out of LSTM? And why?
  A. Long-Short Term Memory
  B. Long Short-Term Memory
  C. Long Short Term-Memory
  D. Long Short Term Memory

Bidirectional RNNs

- Example: The sentiment for "The movie was terribly exciting!"
  - If we use vanilla RNN and read the sentence from left to right, the sentiment is likely negative after "terribly"
  - If we consider the right context of "terribly", "exciting" modifies the meaning from negative to positive



Vanilla RNN



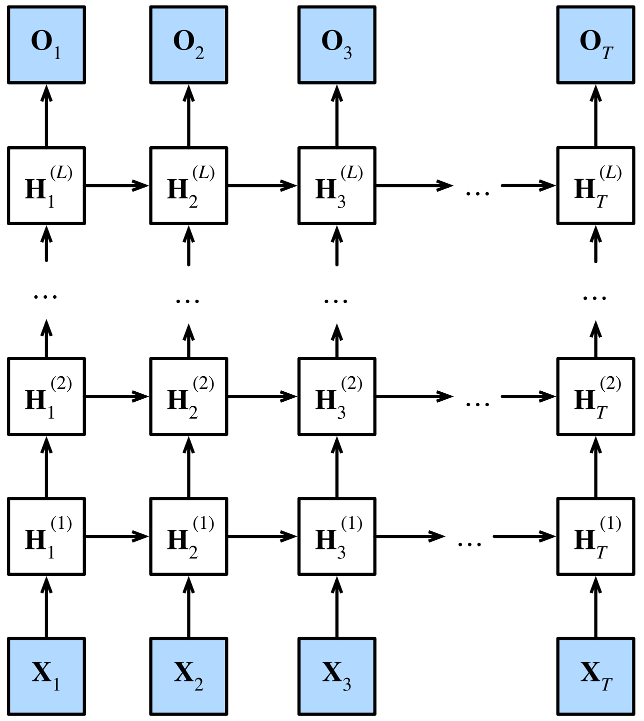
Bidirectional RNN

Bidirectional RNNs

- On time step t:
  - Forward RNN: $\overrightarrow{h}^{(t)} = \text{RNN}_{FW}(\overrightarrow{h}^{(t-1)}, x^{(t)})$
  - Backward RNN: $\overleftarrow{h}^{(t)} = \text{RNN}_{BW}(\overleftarrow{h}^{(t+1)}, x^{(t)})$
  - Concatenated hidden states: $h^t = [\overrightarrow{h}^{(t)}; \overleftarrow{h}^{(t)}]$
- The RNN could be a simple RNN or LSTM
- The forward and backward RNNs have separate weights
- The concatenated hidden states are regarded as "the hidden state" of a bidirectional RNN. This is what we pass on to the next parts of the network
- Note: The bidirectional RNNs are only applicable if you have access to the entire input sequence
  - For example, sentiment analysis is applicable
  - If you do have entire input sequence, bidirectionality is powerful and should be used by default
  - BERT (Bidirectional Encoder Representations from Transformers) is a powerful pretrained contextual representation system built on bidirectionality

Multi-layer RNNs

- RNNs are already "deep" on one dimension since they unroll over many time steps
- We can also make them deep in another dimension by applying multiple RNNs – multi-layer RNN
  - The hidden states from RNN layer i are the inputs to RNN layer $i + 1$
- Multi-layer RNN allows the network to compute more complex representations
  - The lower RNNs should compute lower-level features and the higher RNNs should compute higher-level features
- High-performing RNNs are usually multi-layer
  - For example: In a 2017 paper, Britz et al. find that for Neural Machine Translation, 2 to 4 layers is best for the encoder RNN, and 4 layers is best for the decoder RNN
  - Transformer-based networks (e.g., BERT) are usually deeper, like 12 or 24 layers



Tutorial

- IMDB movie review data
- Using word embeddings such as Word2Vec or GloVe in conjunction with LSTM networks for sentiment analysis
- Leveraging pre-trained models from HuggingFace for sentiment analysis

- Thanks for our TA Dong Woo Kim!

Acknowledgement

The lecture note has benefited from various resources, including those listed below. Please contact Zonghao Yang (zyang99@stevens.edu) with any questions or concerns about the use of these materials.

- Lecture Notes on Recurrent Neural Networks and LSTM from CS224N 2025 Winter at Stanford University

STEVENS
INSTITUTE OF TECHNOLOGY
1870

THANK **YOU**

Stevens Institute of Technology
1 Castle Point Terrace, Hoboken, NJ 07030