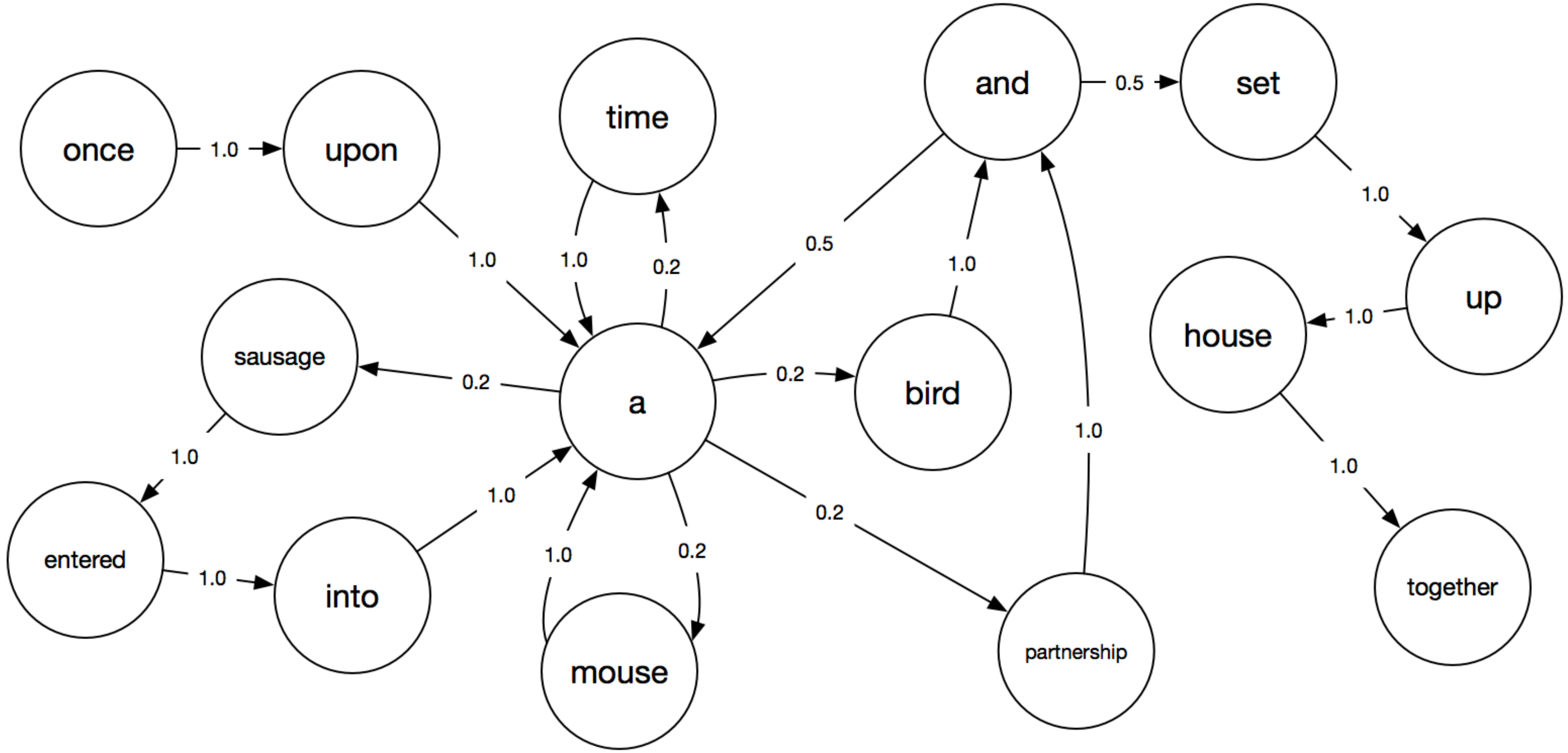
Topic 12
Text Generation

Text generation

- Markov Chains
- RNNs and LSTMs
- Seq-2-Seq

Markov Models for Text Generation

- One of the earliest algorithms used for text generation



A Markov chain

- Typically consists of two entities
  - transition matrix
  - initial state
- Simulate a random walk through the possible states starting from a state of our choice
- Follow the transition probabilities

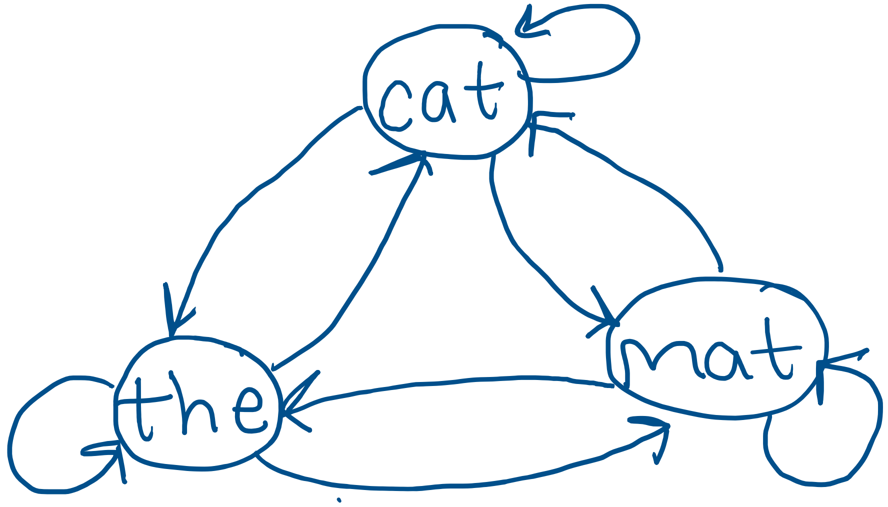
$$MC = \{\text{States}, \text{transitions}\}$$

words

$$\text{word}_i \xrightarrow{P_{ij}} \text{word}_j$$

A Markov chain

- Typically consists of two entities
  - transition matrix
  - initial state
- Simulate a random walk through the possible states starting from a state of our choice
- Follow the transition probabilities

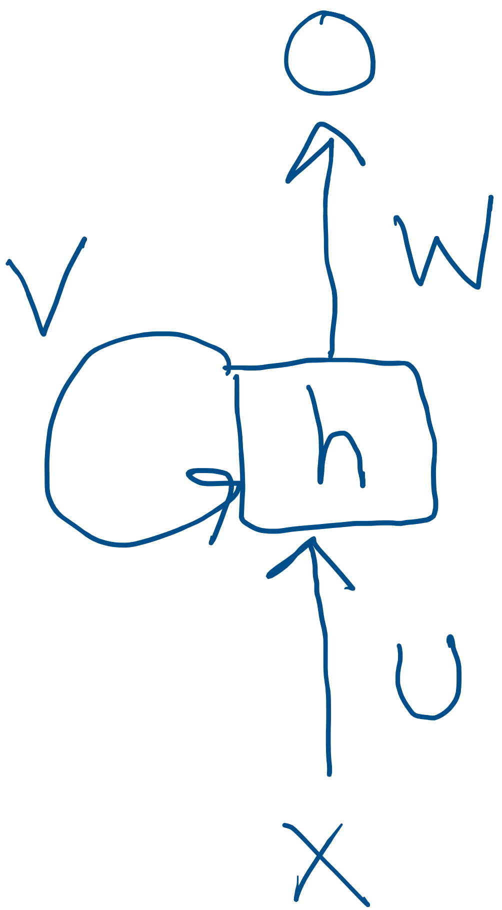


	cat	mat	the
cat			$P_{cat,the}$
mat			
the			

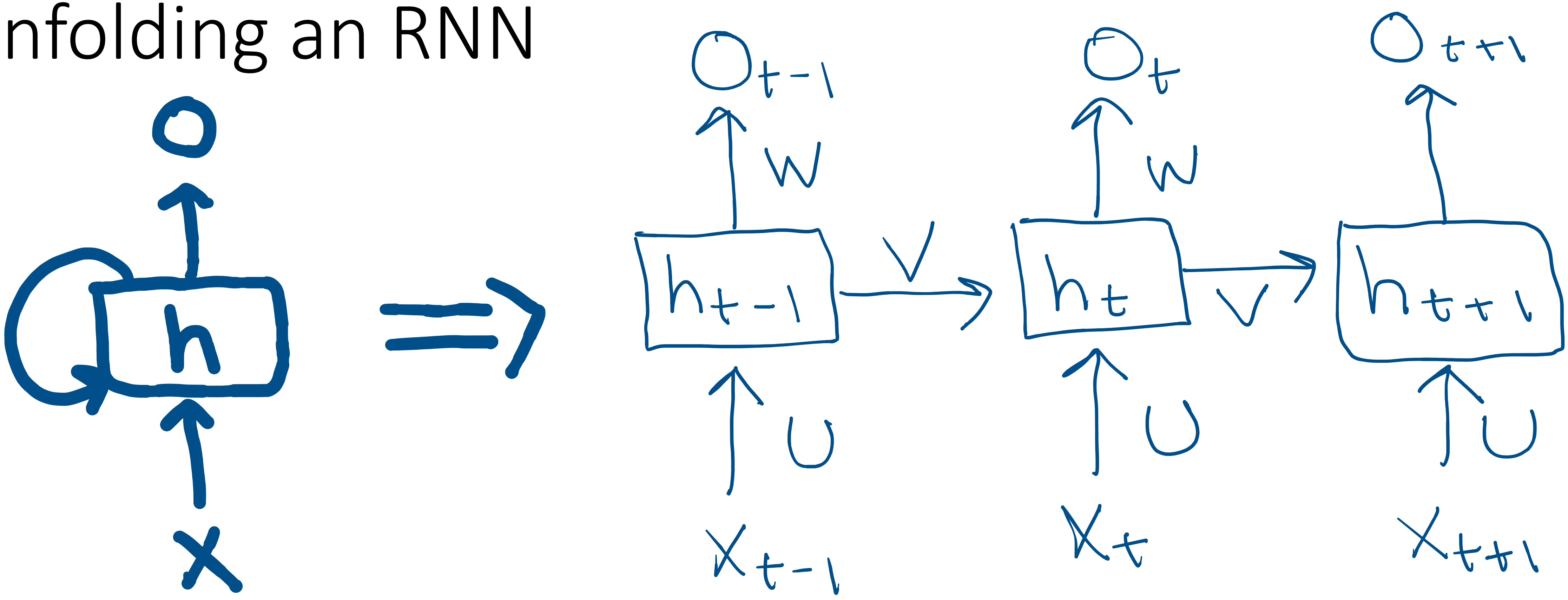
$$(P_{cat,the})^2 = (P_{cat,mat} \cdot P_{mat,the}) + (P_{cat,cat} \cdot P_{cat,the})$$
$$+ (P_{cat,the} \cdot P_{the,the})$$

Recurrent Neural Networks

- Input (x): the current input at time t
- Hidden state (h): the network's internal memory, which is updated at each time step t
- Output (o): the output at time t, which depends on the current input and the previous hidden state

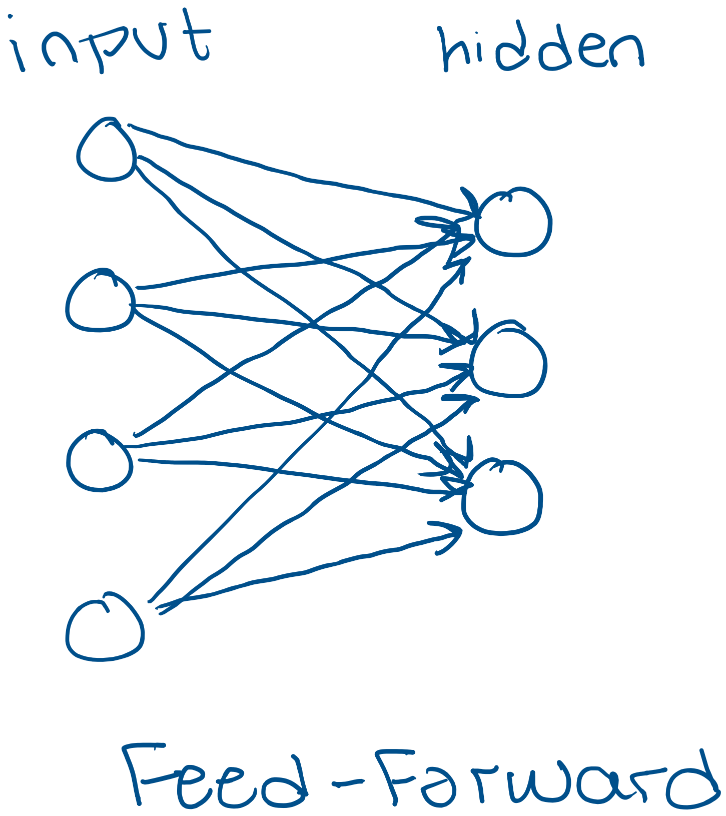


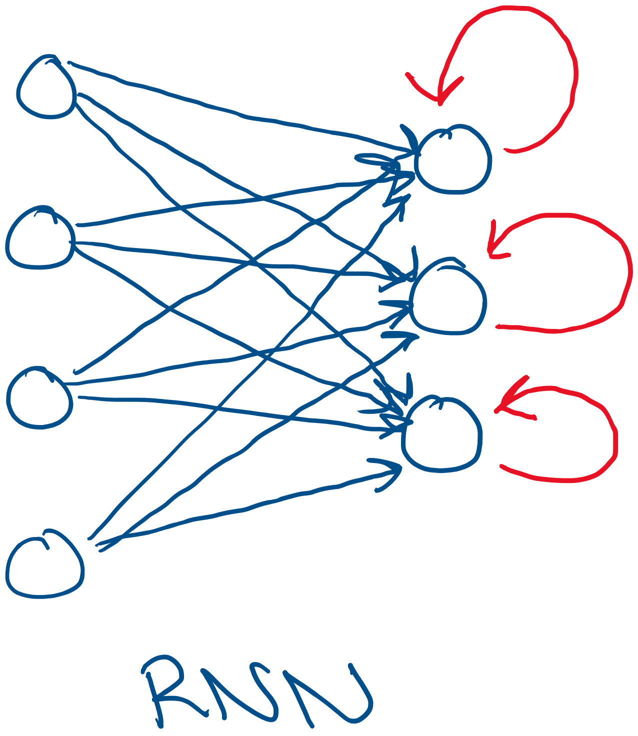
Unfolding an RNN



An unrolled recurrent neural network

RNN vs Fully Connected Feed Forward Network





Training an RNN

1. Initialize the hidden state to zero.
2. For each time step t:
    1. Compute the input to the hidden layer, $h_t = \sigma_1(Ux_t + Vh_t + b_1)$
    2. Compute the input to the output layer, $o_t = \sigma_2(Wh_t + b_2)$

- The final output of the network is the output at the last time step, o(N), where N is the length of the input sequence.

Backpropagation (Review)

- Backpropagation training algorithm modifies the weights of a neural network in order to minimize the error of the network outputs compared to some expected output
    1. Compare the predicted outputs to the expected outputs and calculate the error.
    2. Calculate the derivatives of the error with respect to the network weights.
    3. Adjust the weights to minimize the error.
    4. Repeat.

What about sequence data that may be temporally ordered?

Backpropagation Through Time (BPTT)

Unfold the network over time and propagate the errors backwards

1. Present a sequence of timesteps of input and output pairs to the network.
2. Unroll the network then calculate and accumulate errors across each timestep.
3. Roll-up the network and update weights.
4. Repeat.

Backpropagation Through Time (BPTT)

$$h_t = \sigma_1(Ux_t + Vh_t + b_1)$$ hidden states

$$O_t = \sigma_2(Wh_t + b_2)$$ output distribution

softmax

$$E_t = (g_t - O_t)^2$$

error

ground truth

what was output at time t

Backpropagation Through Time (BPTT)

$$E_t = (g_t - O_t)^2$$

$$E_3 = (g_3 - O_3)^2$$

Adjusting W:

$$\frac{\partial E_3}{\partial W} = \frac{\partial E_3}{\partial O_3} \cdot \frac{\partial O_3}{\partial W}$$

$$\begin{tikzcd}
 & & O_3 \\
h_1 \arrow[r] & h_2 \arrow[r] & h_3 \arrow[u, "W"']
\end{tikzcd}$$

Backpropagation Through Time (BPTT)

$$E_t = (g_t - O_t)^2$$

$$E_3 = (g_3 - O_3)^2$$

Adjusting V:

$$\frac{\partial E_3}{\partial V} = \sum_{i=1}^{N} \frac{\partial E_N}{\partial O_N} \cdot \frac{\partial O_N}{\partial h_i} \cdot \frac{\partial h_i}{\partial V}$$

$$h_1 \xrightarrow{V} h_2 \xrightarrow{V} h_3 \xrightarrow{W} O_3$$

have to consider previous time steps.

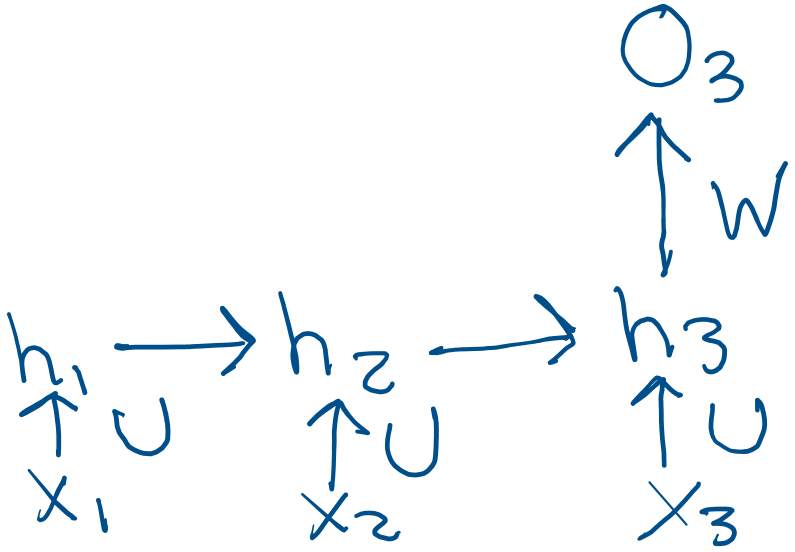
Backpropagation Through Time (BPTT)

$$E_t = (g_t - O_t)^2$$

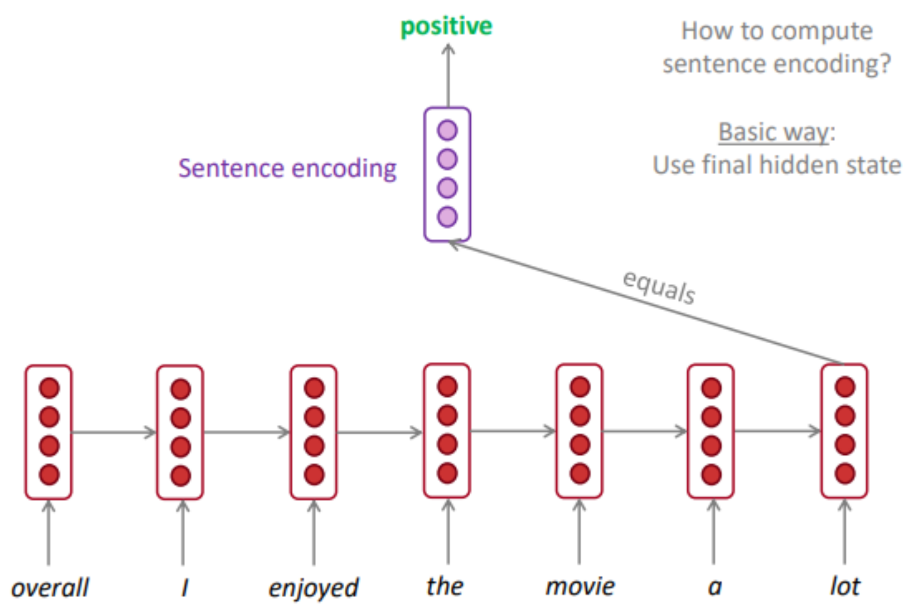
$$E_3 = (g_3 - O_3)^2$$

Adjusting U:

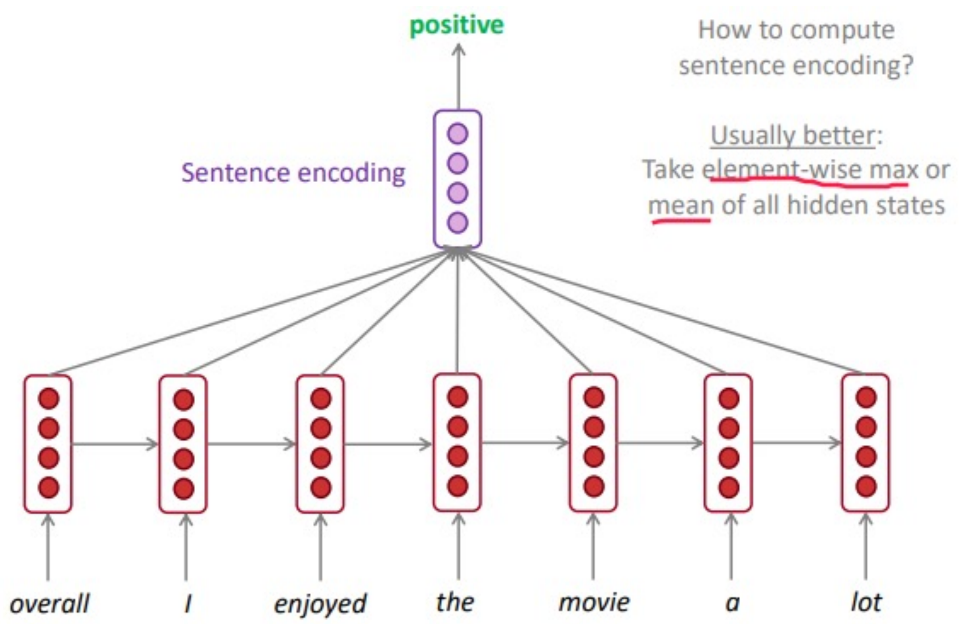
$$\frac{\partial E_3}{\partial U} = \sum_{i=1}^{N} \frac{\partial E_N}{\partial O_N} \cdot \frac{\partial O_N}{\partial h_i} \cdot \frac{\partial h_i}{\partial U}$$



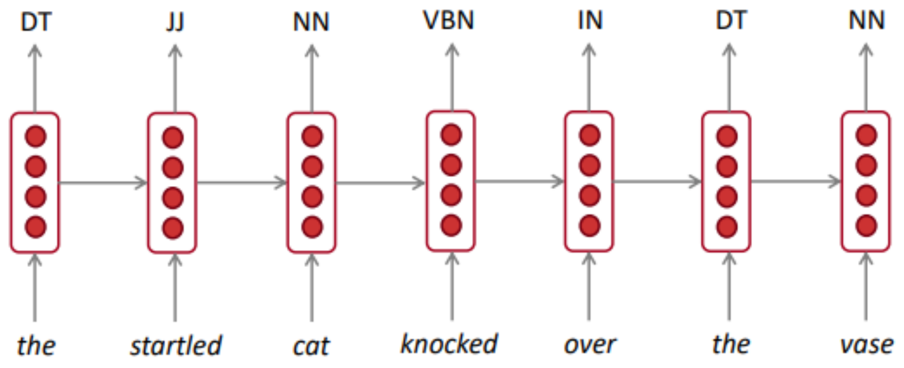
have to consider previous time steps.



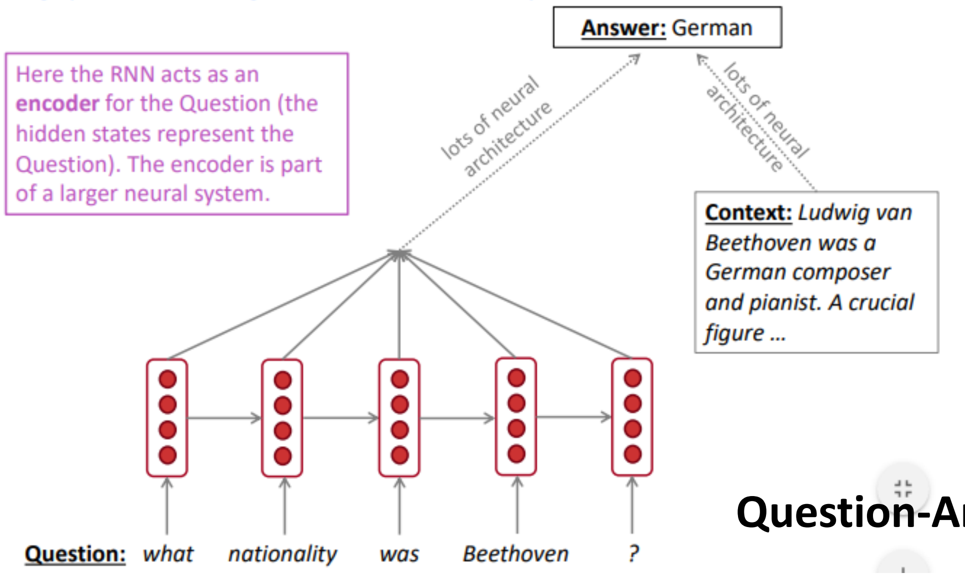
Sentiment Analysis



Sentiment Analysis



POS Tagging



Question-Answer

Drawbacks of RNNs

- RNNs consist of sequential steps
  - If input sequences are comprised of thousands of timesteps, then this will be the number of derivatives required for a single update weight update

- Errors accumulate over time:

The values in the calculations in these backpropagation steps are large (> 1)

- Exploding Gradients

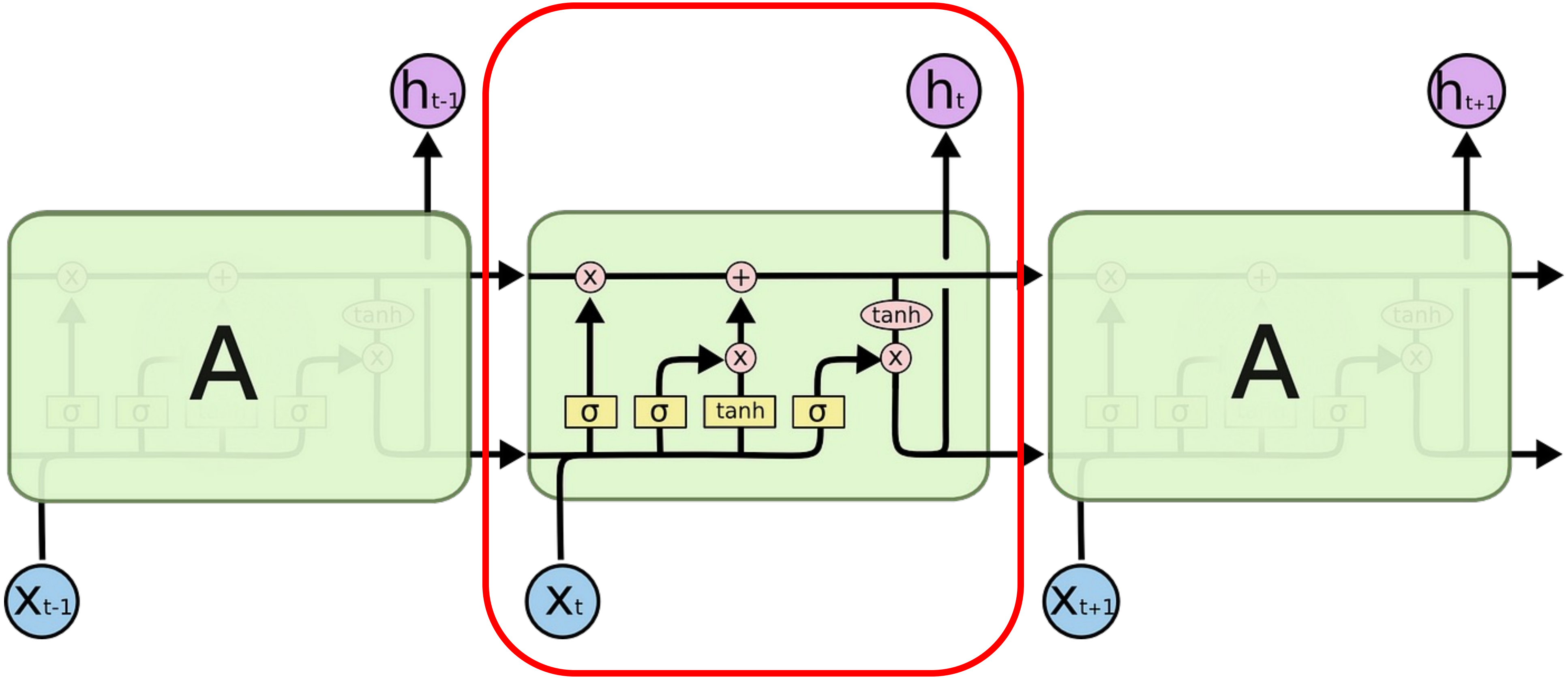
The values in the calculations in these backpropagation steps are small (< 1)

- Vanishing Gradients

Methods to solve these issues

- Exploding Gradients
  - Truncated backpropagation through time
  - Gradient clipping
- Vanishing Gradients
  - Early stopping
  - Weight initialization
  - Use LSTMs or GRUs

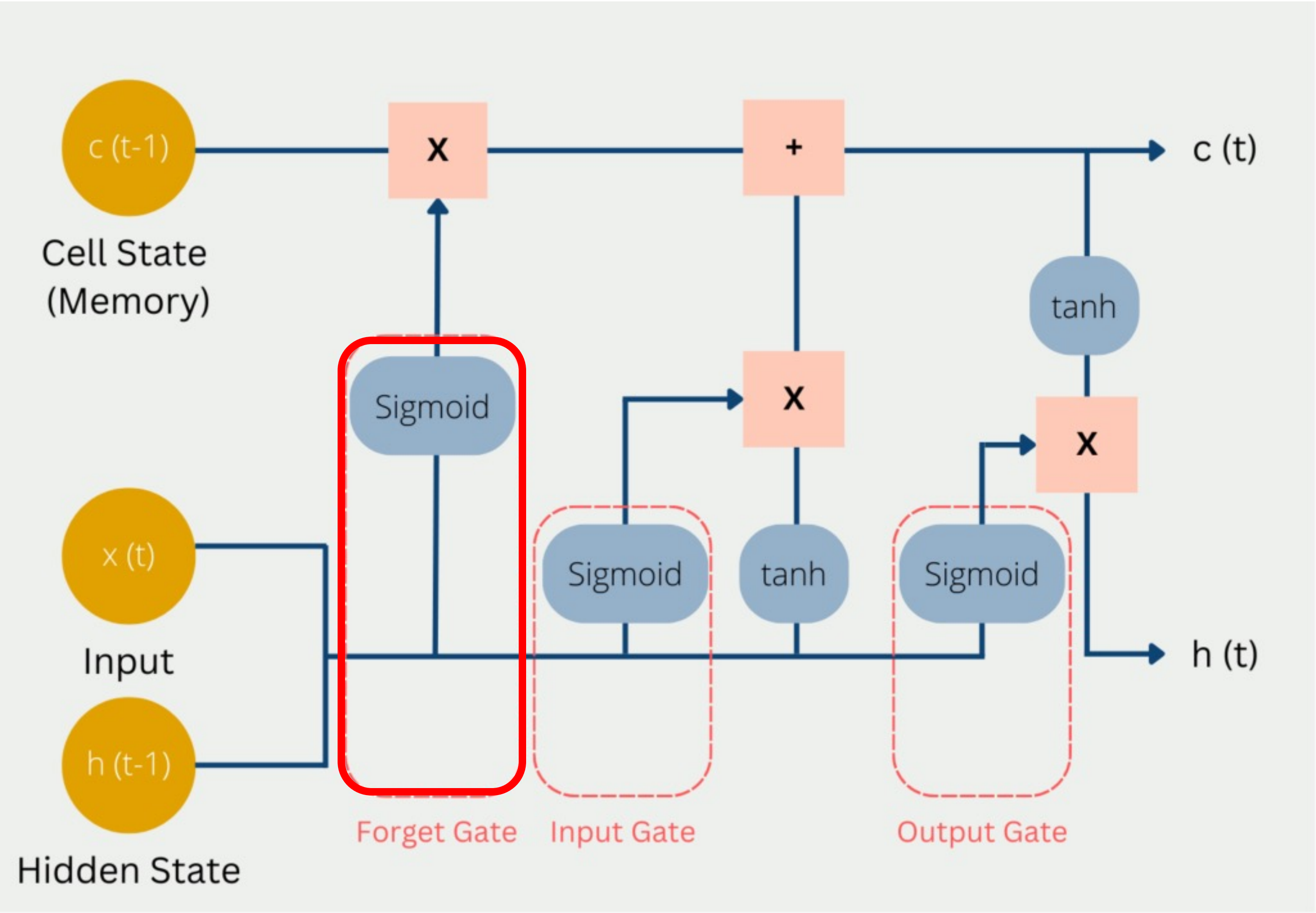
Long Short-Term Memory Networks



Long Short-Term Memory Networks

- Forget gate: controls which information from the previous cell state should be discarded

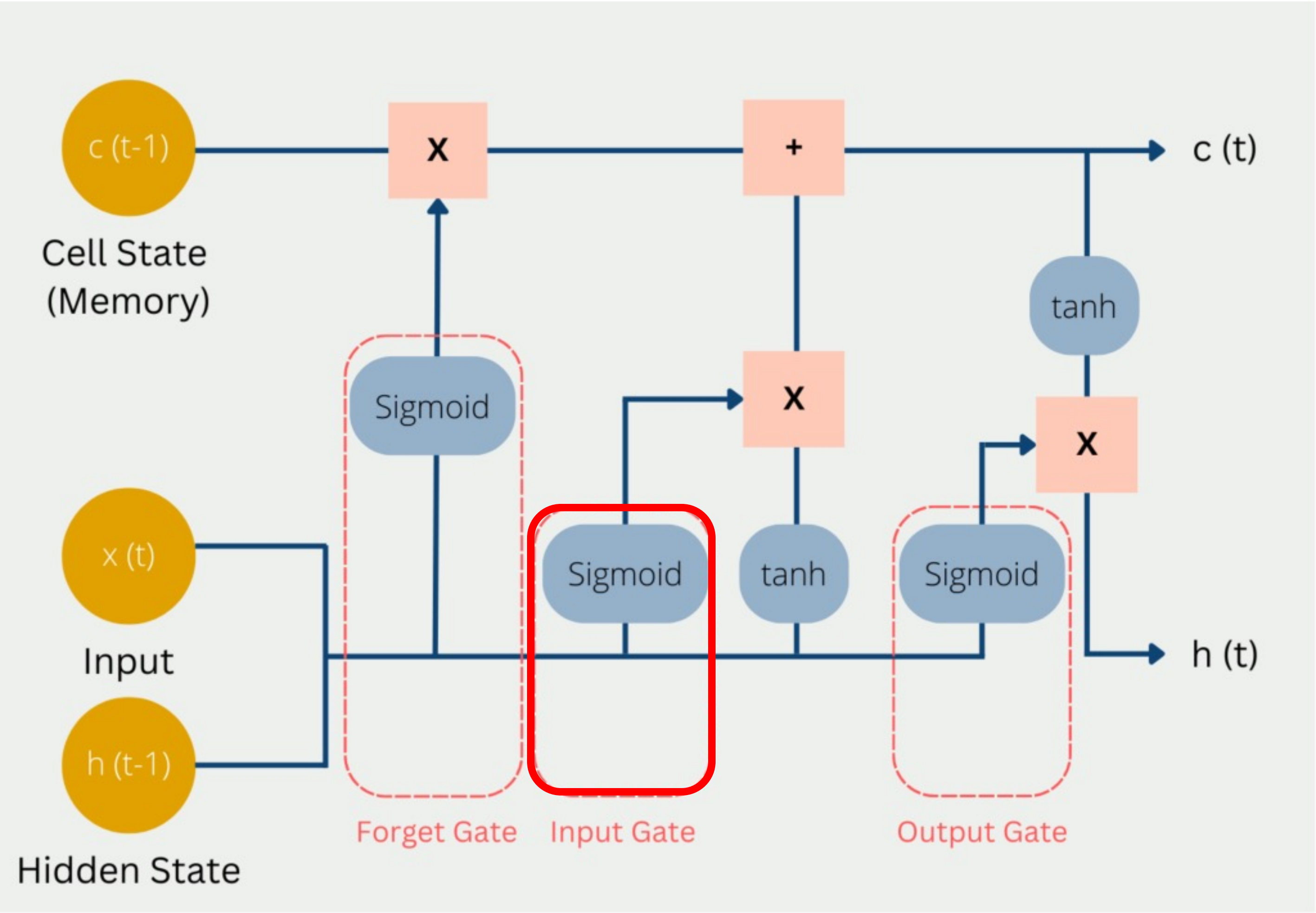
$$f_t = \sigma(W_f\ [x_t, h_{t-1}] + b_f)$$



Long Short-Term Memory Networks

- Input gate: controls which information from the current input and previous hidden state should be stored in the cell state

$$i_t = \sigma(W_i[x_t, h_{t-1}] + b_i)$$

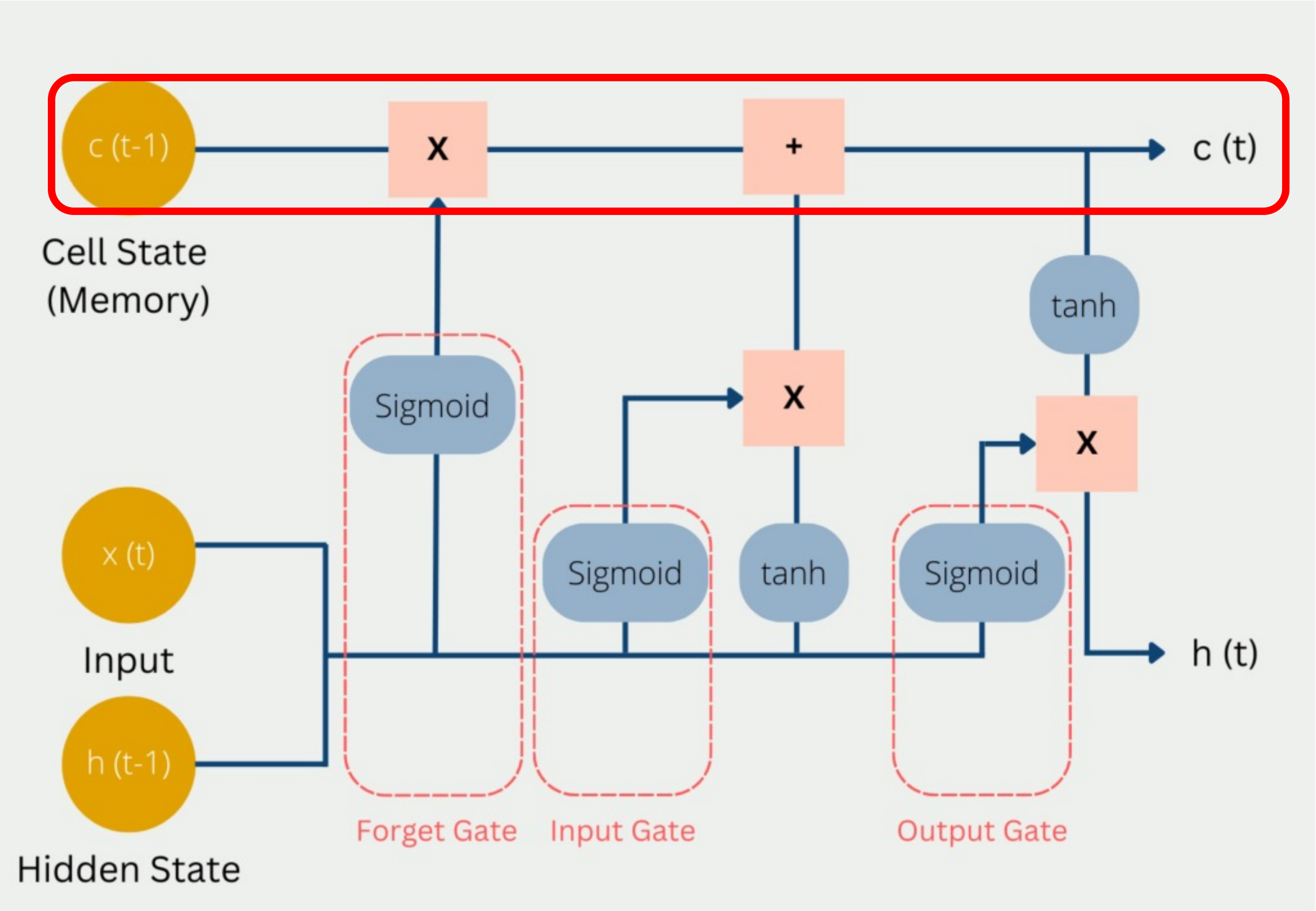


Long Short-Term Memory Networks

- Cell state: the long-term memory of the LSTM

$$C_t' = tahn(W_c * [h_{t-1}, x_t] + \mathrm{b_c}$$

$$C_t = f_t * C_{t-1} + i_t * C_t'$$

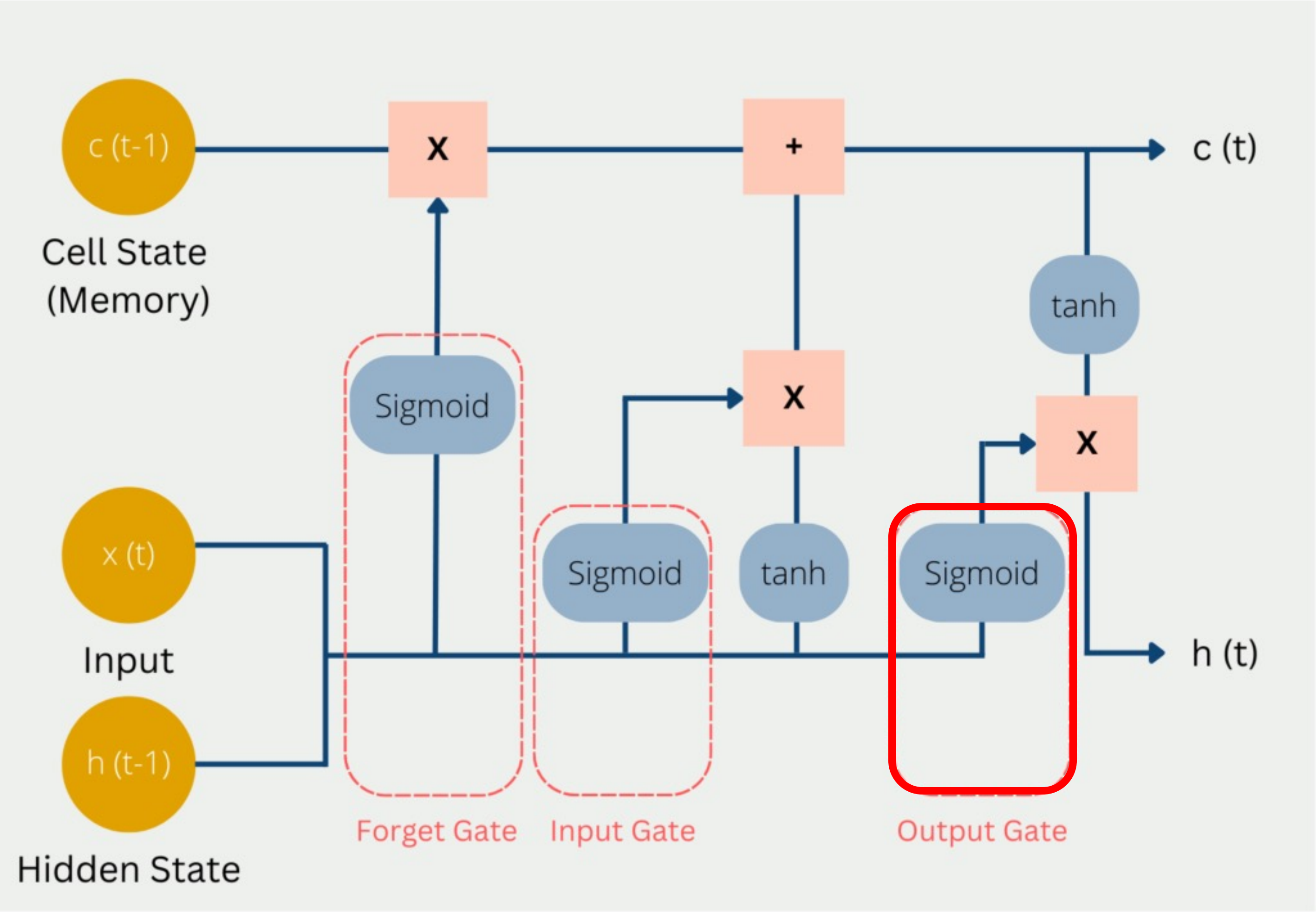


Long Short-Term Memory Networks

- Output gate: controls which information from the current input and previous hidden state should be outputted from the cell state

o(t) = σ(W(o) * [h(t-1), x(t)] + b(o))

h(t) = o(t) * tanh(C(t))

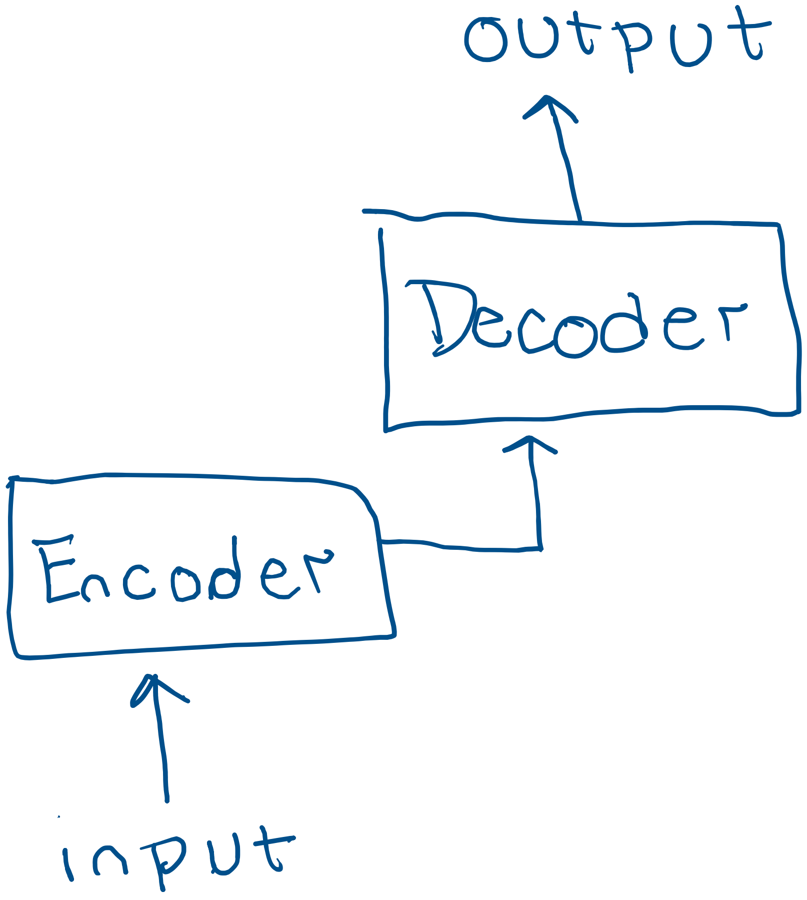


LSTMs vs vanilla RNNs

- Ability to handle long(er)-term dependencies by maintaining a cell state
- Ability to selectively remember or forget information through the gates
- Ability to output information based on the current input and previous hidden state

Sequence to Sequence Models

- The architecture of a Seq-2-Seq model consists of two main components: an encoder and a decoder
- **Encoder:** recurrent neural network (RNN) such as LSTM or GRU, which produces a fixed-length vector representation of the input sequence
- **Decoder:** recurrent neural network (RNN) such as LSTM or GRU, which uses the vector representation produced by the encoder to generate the output sequence one element at a time



Later: Transformers for text generation

- These seq-2-seq models are typically used for chatbots
  - For now, we will use a pre-trained models

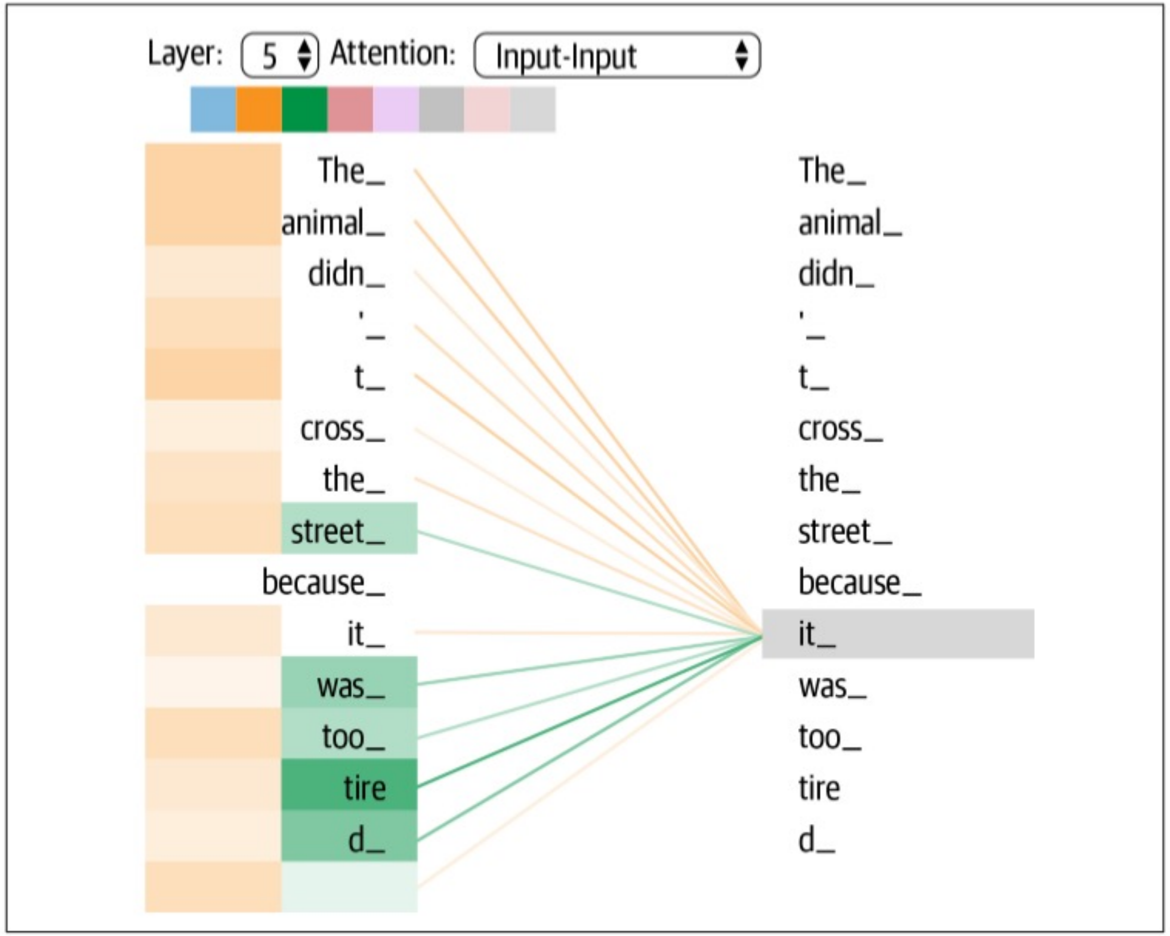


Figure 1-17. Self-attention mechanism in a transformer []

Next time

- Building a chatbot
    - Rule-based
    - Open-ended