ALGONQUIN COLLEGE

CST8507: NATURAL LANGUAGE PROCESSING

WEEK#5

INTRODUCTION TO LANGUAGE MODEL

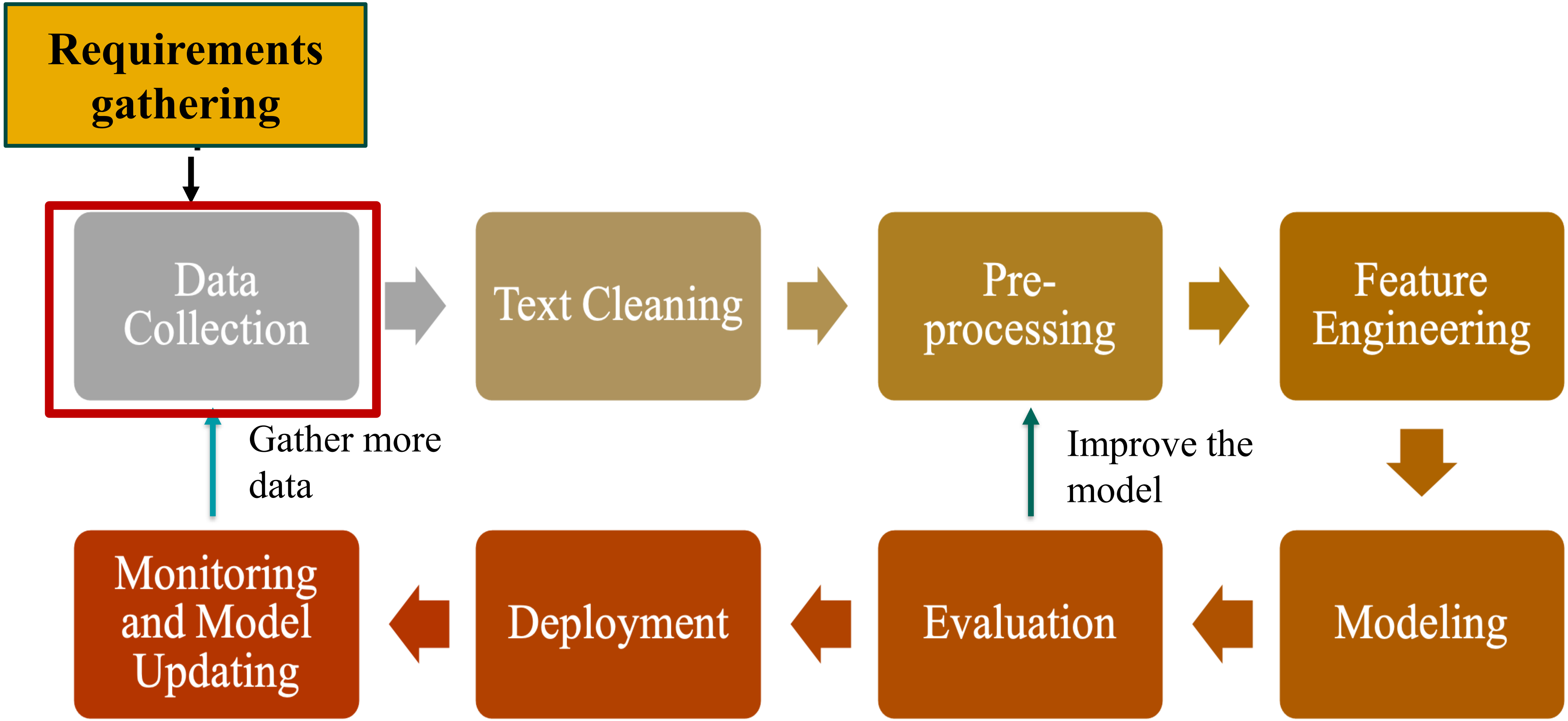
DEVELOPED BY

HALA OWN, PH.D.

Lesson Agenda

- Lab 3
- Text Collection(overview)
- Language Model
  - N-gram
  - NN Language model
    - Recurrent Neural Networks RNN
    - LSTMs

NLP Development Life Cycle



Data generated in one minute on various social platforms

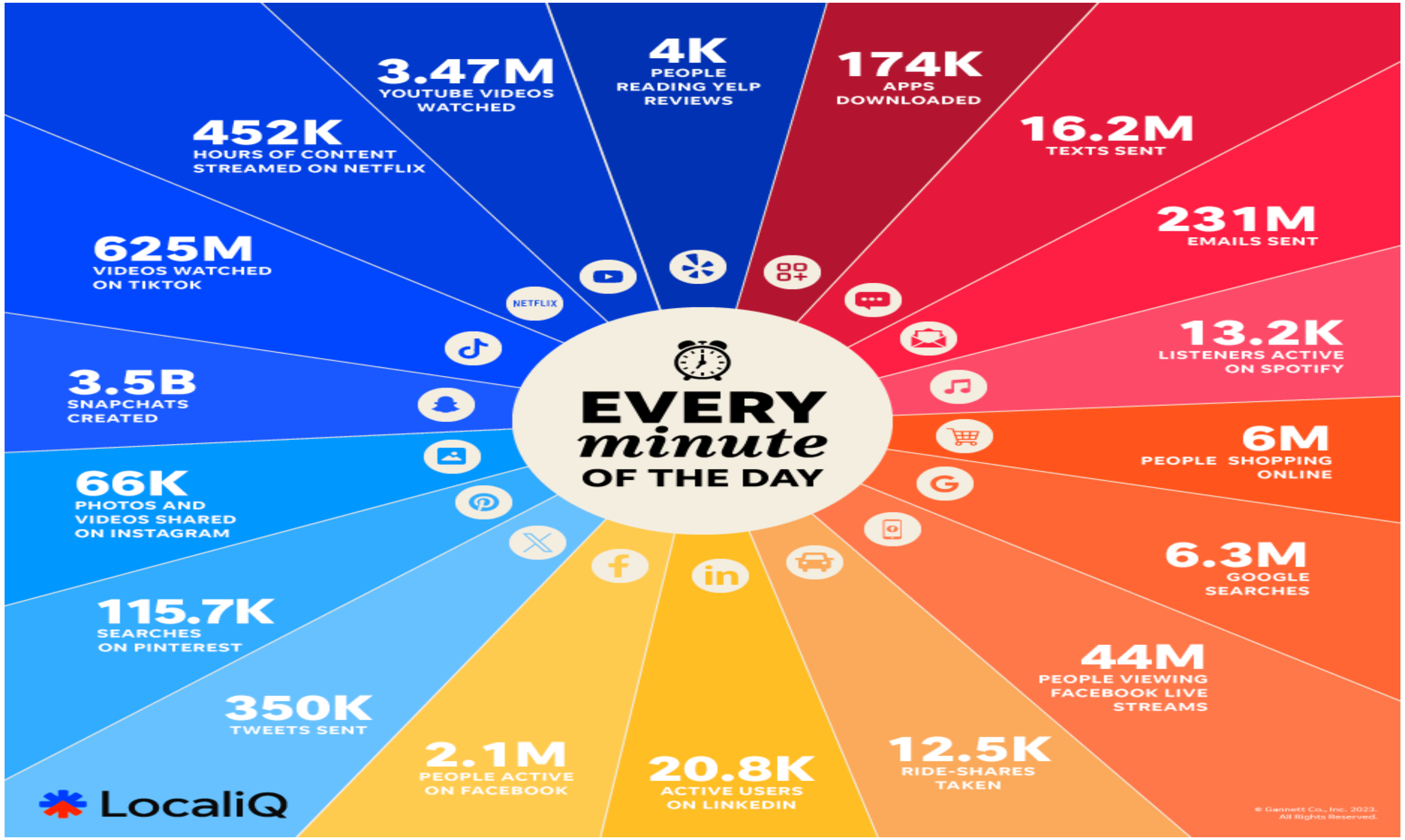
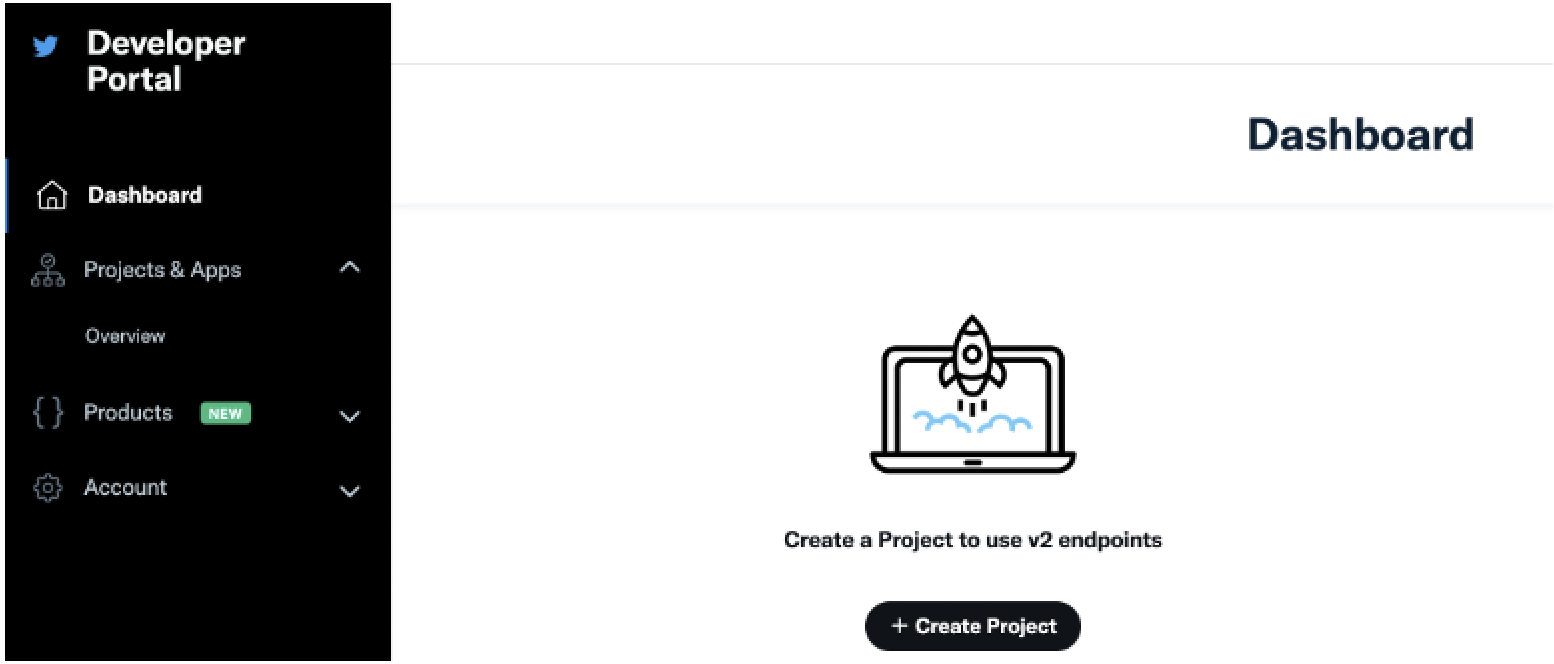


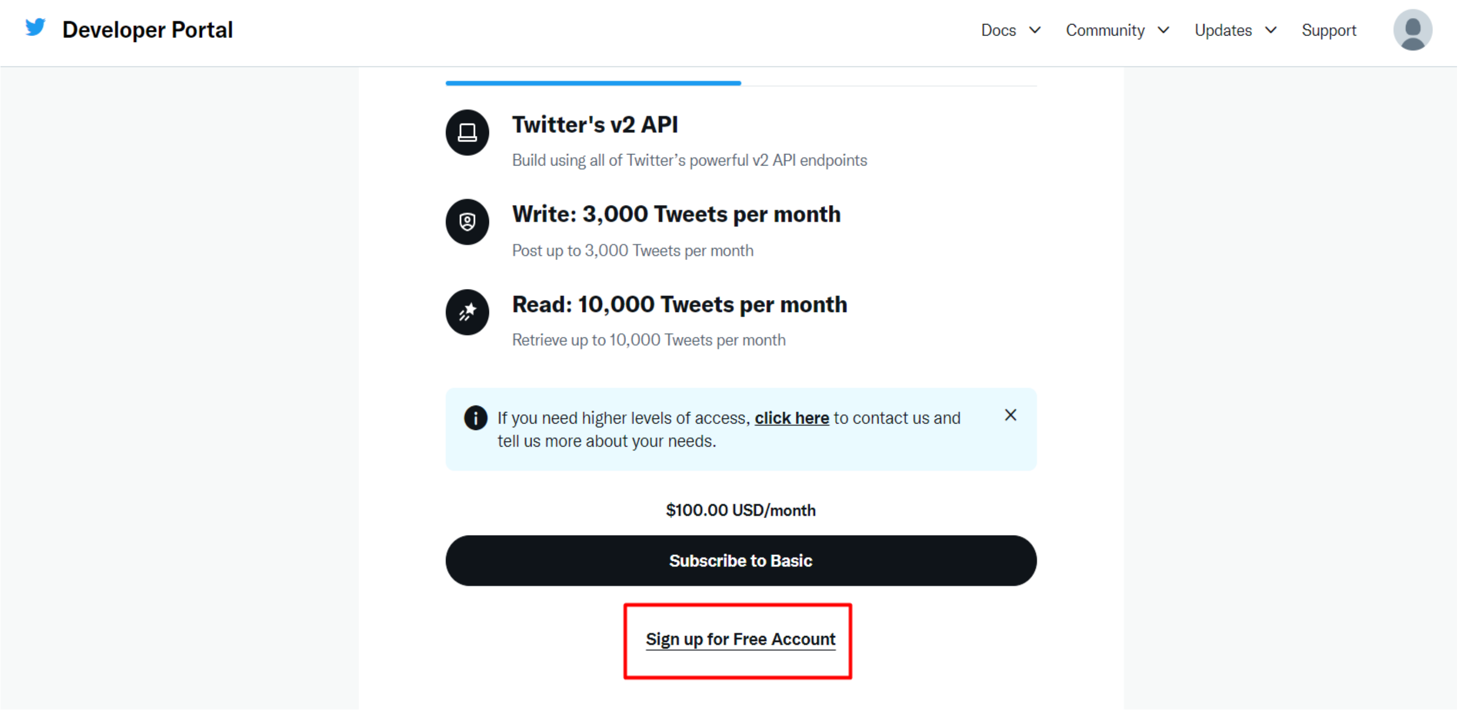
Image source: HTTPs://localiq.com/blog/what-happens-in-an-internet-minute/

Text Collection

- Tweet Collecting
  - X API



Create X Developer Account



https://help.rssground.com/articles/233141-how-to-create-x-twitter-developer-app

Web Scraping:Extraction of data from a website

Python libraries are widely used for parsing HTML:

1. **Beautiful Soup**: A popular library for parsing HTML and XML documents. It simplifies extracting data from web pages and has an active community with detailed documentation.
2. **lxml**: Known for its speed, lxml is one of the fastest parsing libraries available. It receives regular updates, with the latest released in July 2023.
3. **html5lib**: A pure-Python library designed to conform to the WHATWG (**Web Hypertext Application Technology Working Group**) HTML specification, ensuring compatibility with major web browsers.

Demo

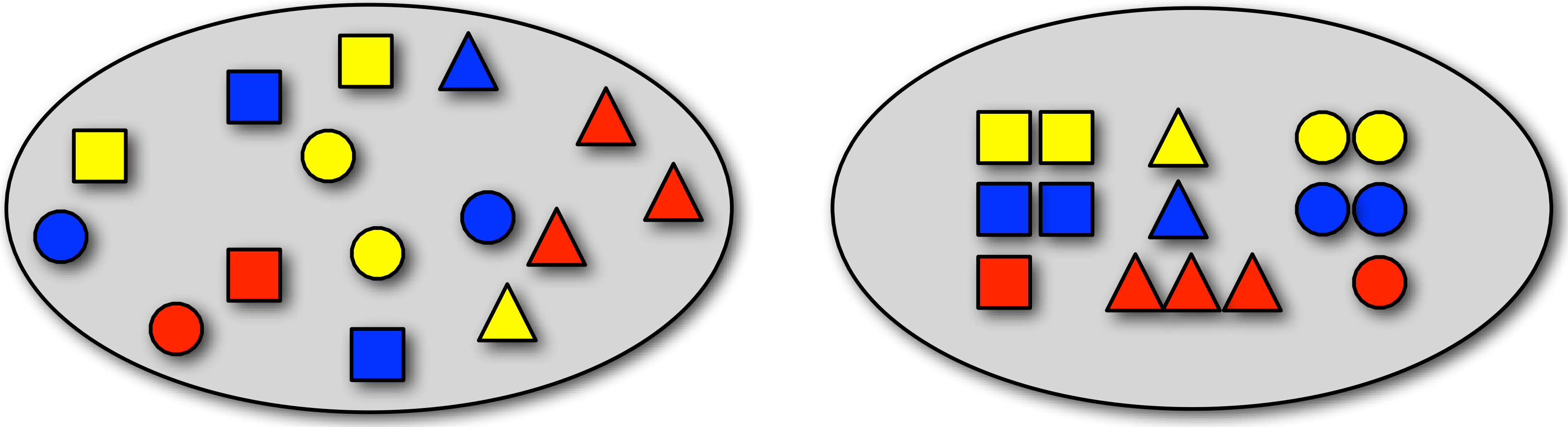
- Inclass code

Reminder

PROBABILITY THEORY

Basic Probability Theory: Sampling with replacement

Pick a random shape, then put it back in the bag.



$P(\color{blue}{\blacksquare}) = 2/15$

$P(\color{blue}{\text{blue}}) = 5/15$

$P(\color{blue}{\text{blue}} \mid \square) = 2/5$

$P(\color{red}{\blacksquare}) = 1/15$

$P(\color{red}{\text{red}}) = 5/15$

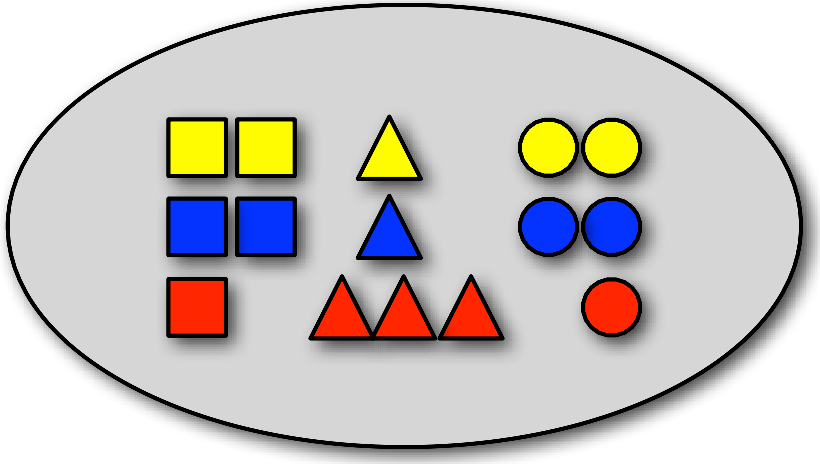
$P(\square) = 5/15$

$P(\color{red}{\blacksquare} \text{ or } \color{blue}{\blacktriangle}) = 2/15$

$P(\triangle \mid \color{red}{\text{red}}) = 3/5$

Sampling with replacement

Pick a random shape, then put it back in the bag.
What sequence of shapes will you draw?



$P({\color{red}\bullet}{\color{yellow}\blacktriangle}{\color{blue}\blacktriangle}{\color{blue}\blacksquare})$
$= 1/15 \times 1/15 \times 1/15 \times 2/15$
$= 2/50625$

$P({\color{red}\blacktriangle}{\color{yellow}\bullet}{\color{blue}\bullet}{\color{red}\blacktriangle})$
$= 3/15 \times 2/15 \times 2/15 \times 3/15$
$= 36/50625$

$P({\color{blue}\blacksquare}) = 2/15$
$P(\text{blue}) = 5/15$
$P(\text{blue} \mid \square) = 2/5$

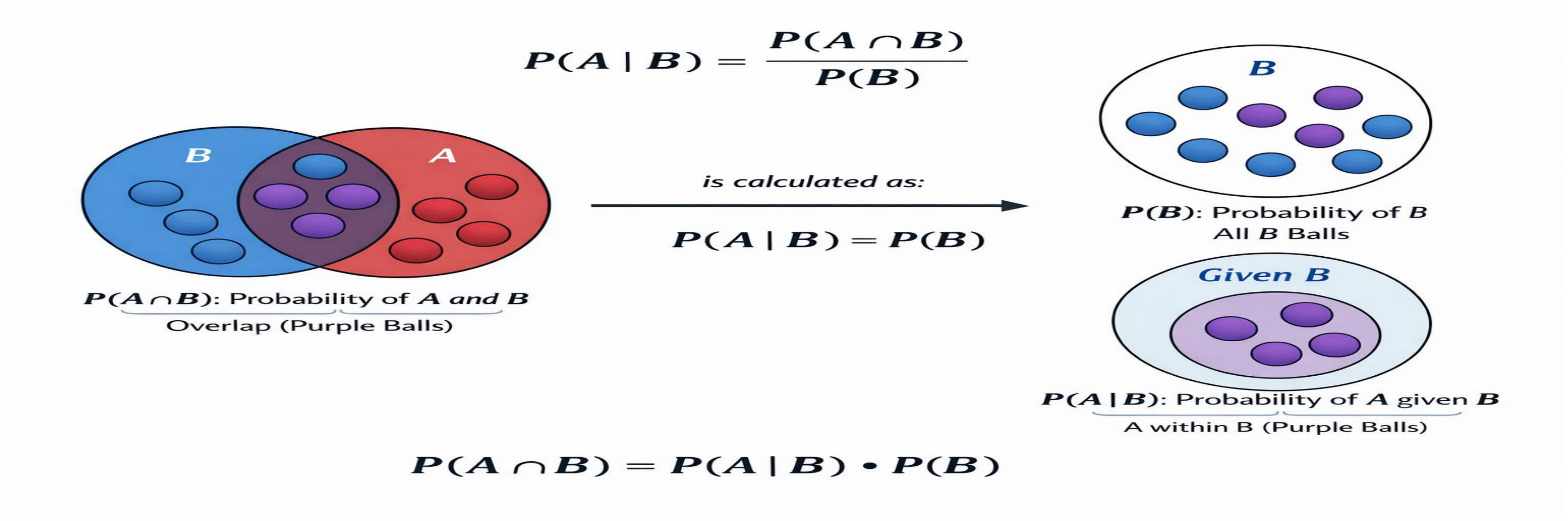
$P({\color{red}\blacksquare}) = 1/15$
$P(\text{red}) = 5/15$
$P(\square) = 5/15$

$P({\color{red}\blacksquare} \text{ or } {\color{blue}\blacktriangle}) = 2/15$
$P(\triangle \mid \text{red}) = 3/5$

Conditional Probability

$$P(X|Y) = \frac{P(X, Y)}{P(Y)}$$

The conditional probability of X given Y, Probability that one event occurs given that another event has already occurred.



Chain Rule of Probability

The **chain rule** expresses a joint probability as a **product** of conditional probabilities.

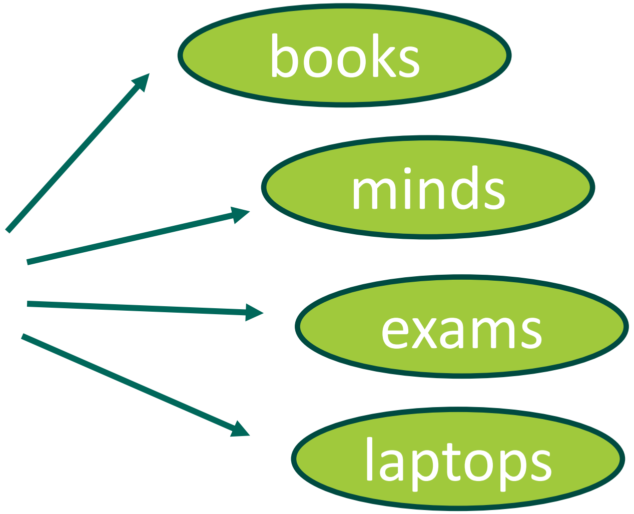
For a sequence of events $\boldsymbol{X_1, X_2, \ldots, X_n}$

$$\boxed{P(X_1, X_2, \ldots, X_n) = P(X_1)\, P(X_2 \mid X_1)\, P(X_3 \mid X_1, X_2) \cdots P(X_n \mid X_1, \ldots, X_{n-1})}$$

LANGUAGE MODELING

Language Modeling: the task of **predicting** what word comes next

the students opened their-------

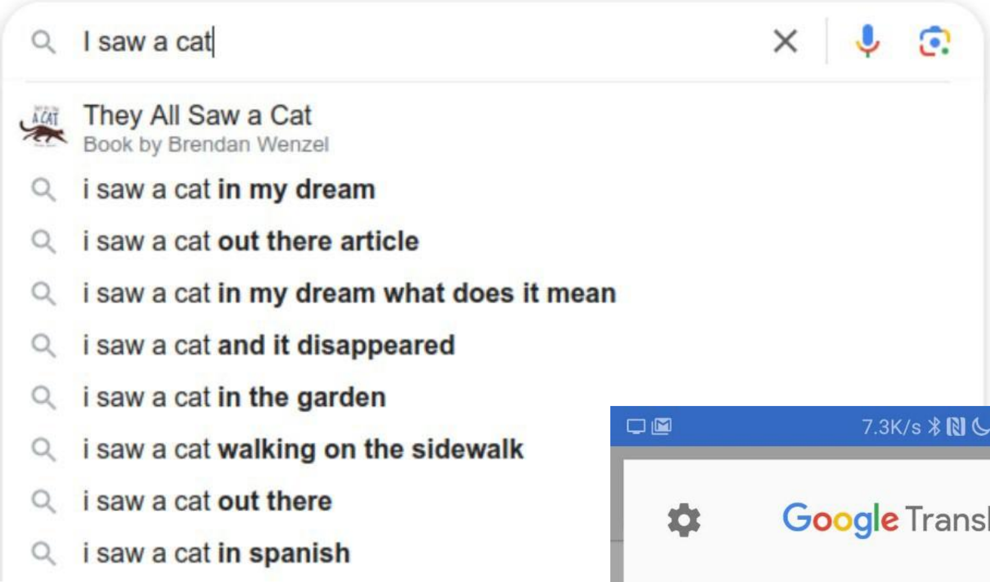


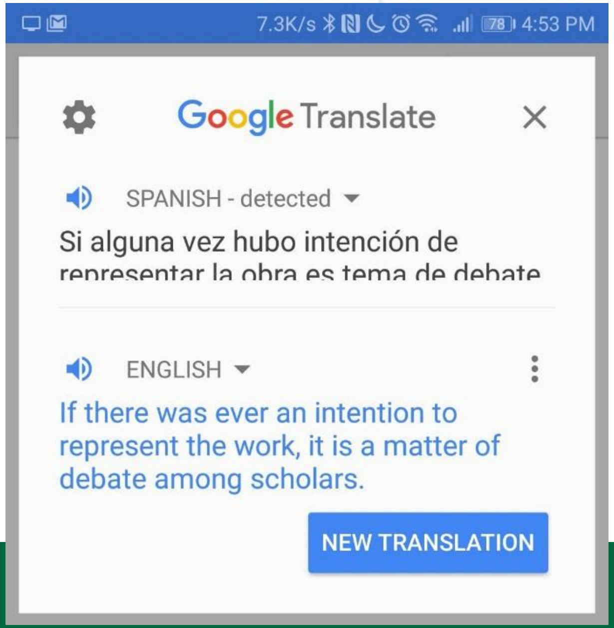
Given a sequence of words $\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, \ldots, \boldsymbol{x}^{(t)}$ $\boldsymbol{x}^{(t+1)}$
compute the probability distribution of the next word $P(\boldsymbol{x}^{(t+1)} | \boldsymbol{x}^{(t)}, \ldots, \boldsymbol{x}^{(1)})$

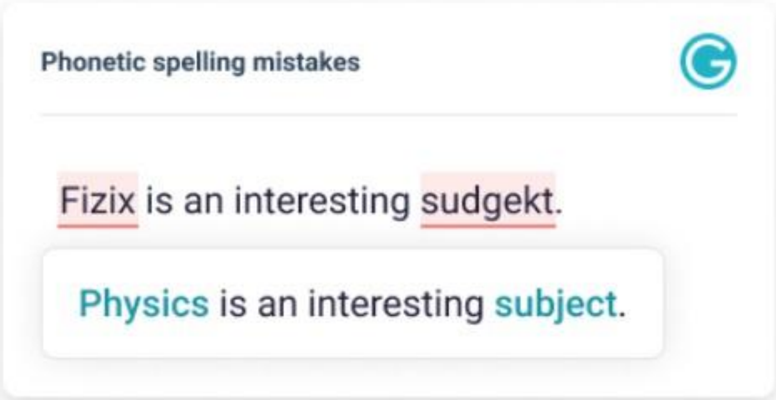
Where $\boldsymbol{x}^{(t+1)}$ can be any word in the vocabulary $V = \{\boldsymbol{w}_1, ..., \boldsymbol{w}_{|V|}\}$

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

Popular Usages





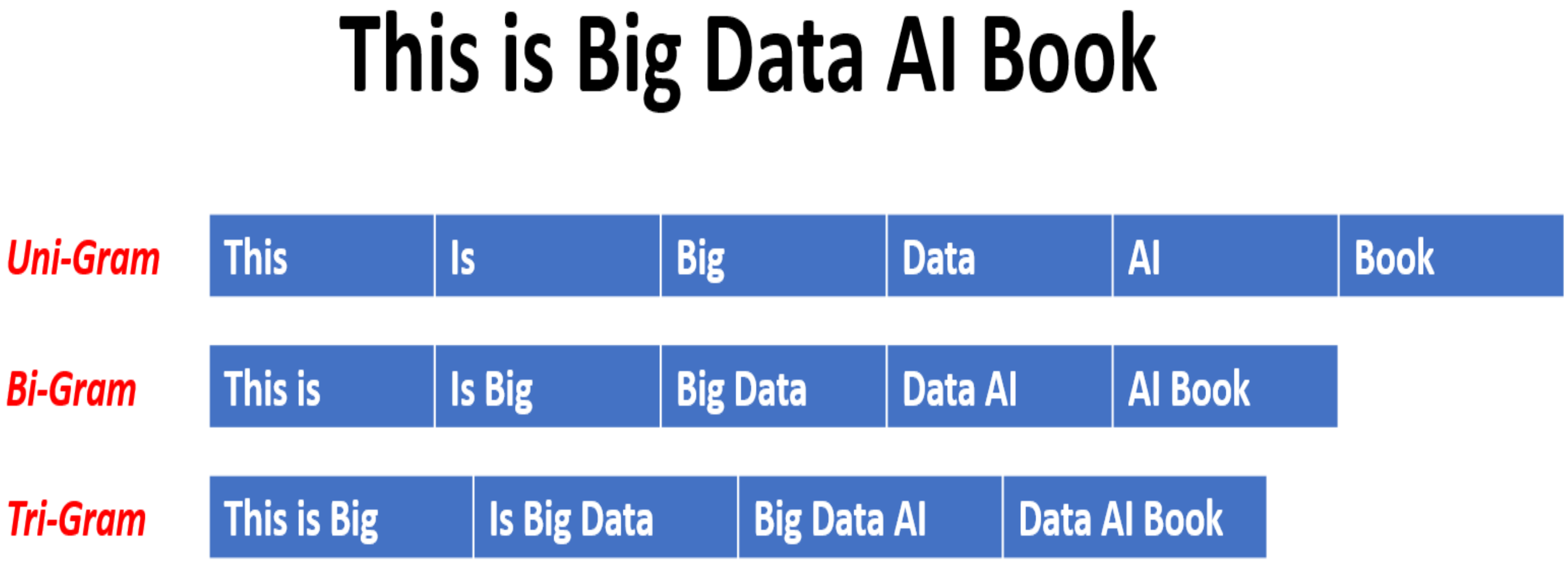


Felix is an interesting subject	0.03
Felix is an interesting dialect	0.005
Physics is an interesting dialect	0.0034
Physics is an interesting subject	0.01

Goal of Language Modeling

learn patterns in text and predict the next word (or sequence of words) based on prior context.

N-gram Language Modeling



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

Image source: https://devopedia.org/n-gram-model

N-gram Language Modeling…

- For example, if we have sequence of tokens $\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(T)}$, then the probability to see these tokens in this order is:

Using chain Rule

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

Language Modeling: n gram…

Our assumption $$P(\boldsymbol{x}^{(t+1)}|\boldsymbol{x}^{(t)},\dots,\boldsymbol{x}^{(1)}) = P(\boldsymbol{x}^{(t+1)}|\underbrace{\boldsymbol{x}^{(t)},\dots,\boldsymbol{x}^{(t-n+2)}}_{n\text{-1 words}})$$

Recall the definition of conditional probabilities

```
p(B|A) = P(A,B)/P(A)

P(A,B) = P(A)P(B|A)
```

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

n-gram Language Models: Example using 4- gram

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

discard

fixed window

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

For example, suppose that in the corpus:

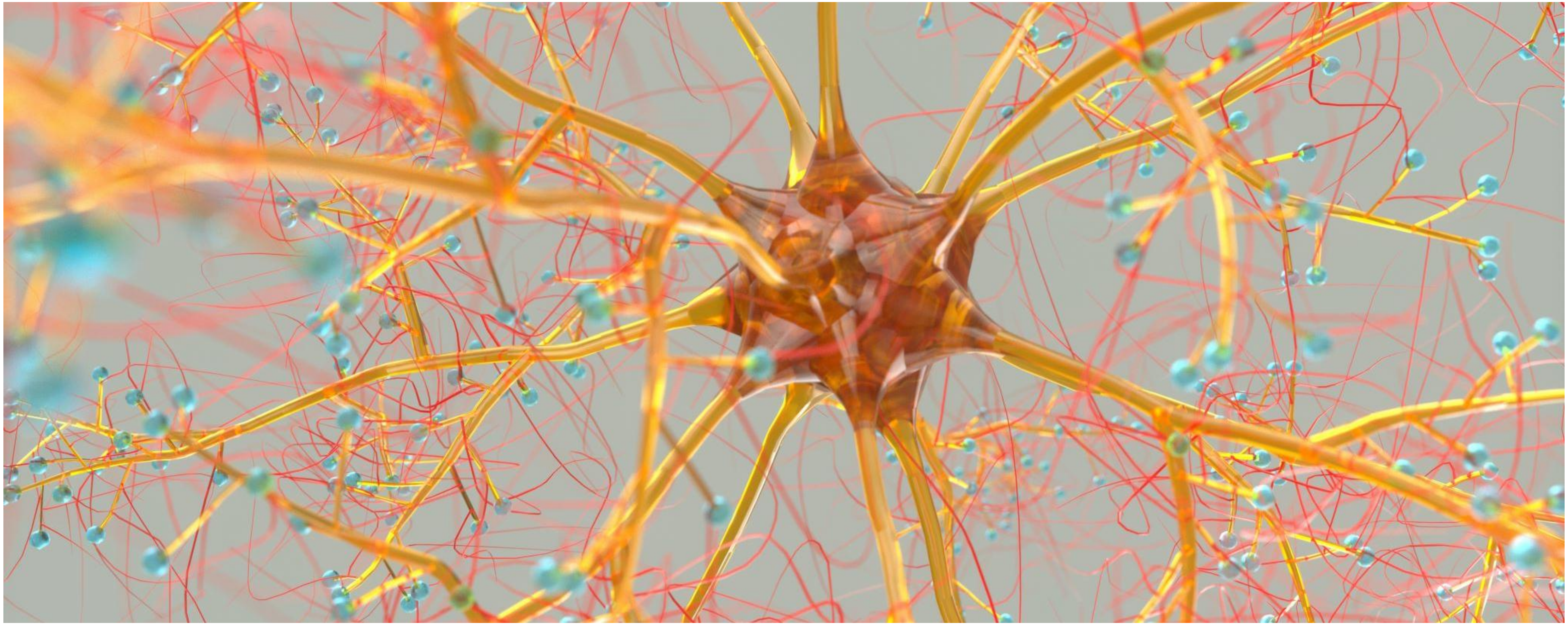
- "students opened their" occurred 1000 times
- "students opened their books" occurred 400 times
  - → `P(books | students opened their) = 0.4`
- "students opened their exams" occurred 100 times
  - → `P(exams | students opened their) = 0.1`

ALGONQUIN COLLEGE

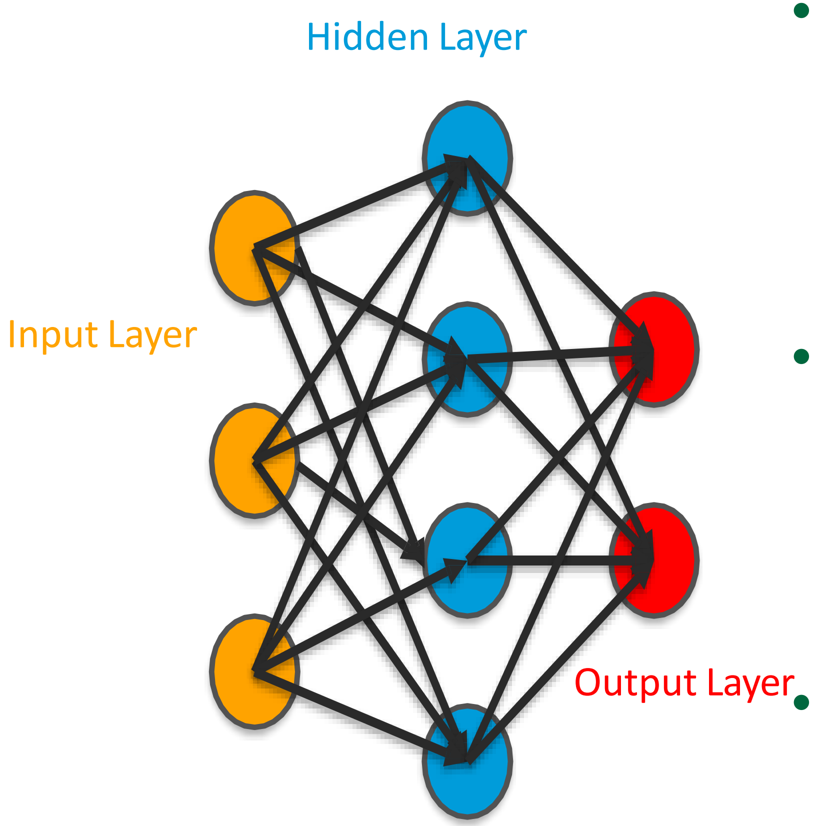
N-grams : limitations and challenges

- Data Sparsity
- Computational Complexity
- Context Limitations

Neural Network Based Language Models

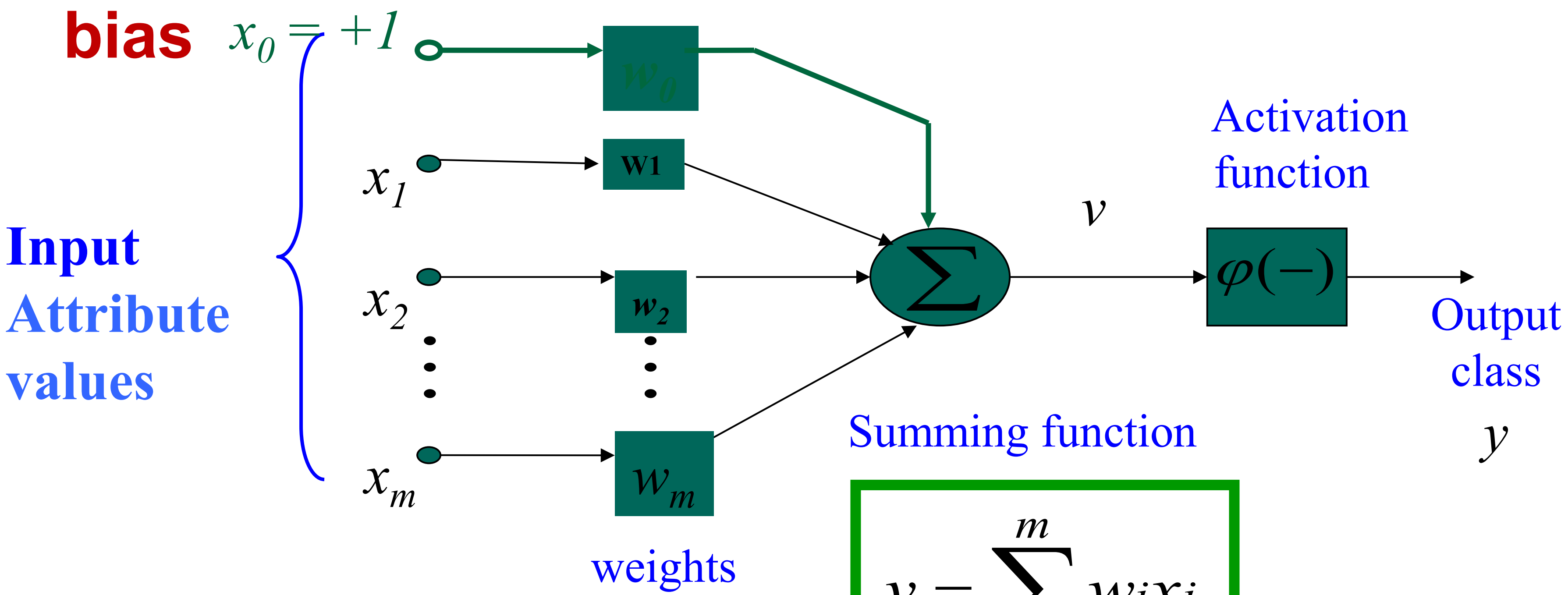


A Quick Review Of Neural Nets



- Input layer is a set of features; each arrow represents a weight (float number) that tells us how much each input contributes to each following step.
- Each node in the hidden layer is some combination of all the inputs. The hidden layer acts as the ‘input’ for the output layer.
- Backpropagation allows us to adjust the weights to improve accuracy and find the ‘correct’ way to combine the inputs and hidden layers to get the best possible results.

NN basic element: Perceptron or Neuron



$$v = \sum_{j=0}^{m} w_j x_j$$

$$w_0 = b$$

Language Model: Neural Nets

~~*as the proctor started the clock*~~ *the students opened their* -----?-----

discard

fixed window

Language Model: Neural Nets …

output distribution

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

hidden layer

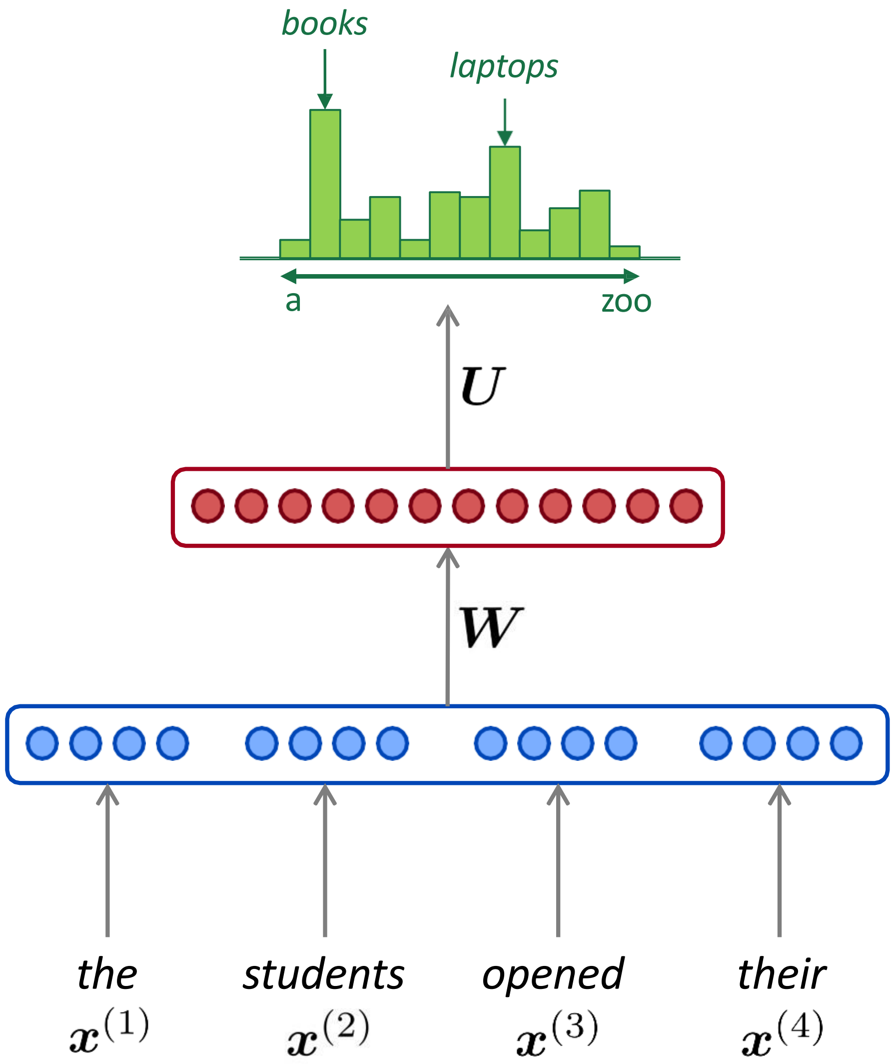
$$\boldsymbol{h} = f(\boldsymbol{W}\boldsymbol{e} + \boldsymbol{b}_1)$$

concatenated word embeddings

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

words / one-hot vectors

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



Feed Forword NN: Limitation...

~~*as the proctor started the clock*~~ *the students opened their* ------------

discard

fixed window

Feed Forword NN: Limitation

"The food was good, not bad at all"

"The food was bad, not good at all"

Feed Forword NN: Limitation…

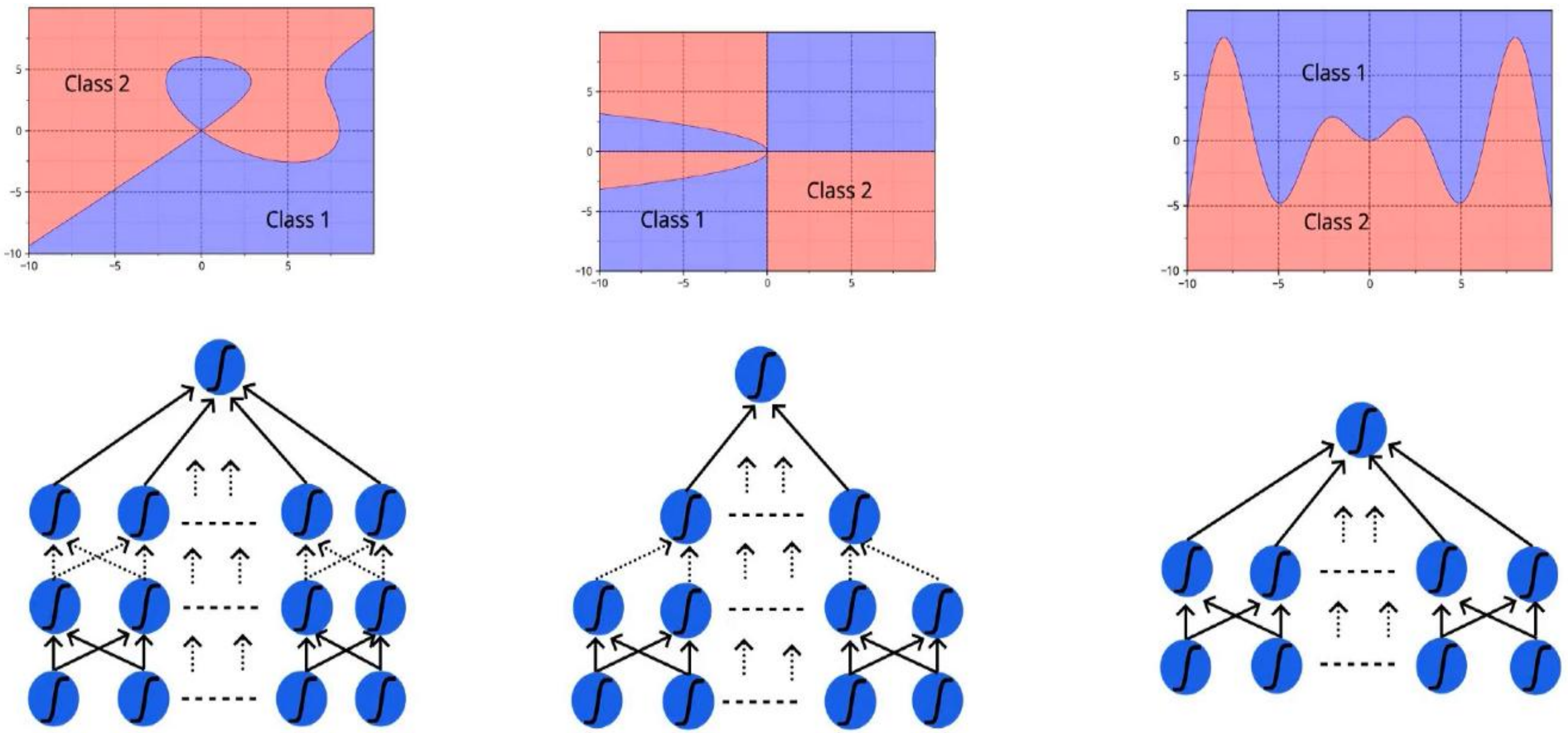
“Just watched the new movie. Loved it! 🍿 #entertained”

“The storyline was captivating, the characters were well-developed, and the cinematography was impressive. Overall, a fantastic movie night! 🎬 👍 #movienight #recommend”

Sequence Modeling: Motivations

- Handle **variable length** sequence data
- Track long **term dependency**
- Maintain information about **order**
- **Share information** across the sequence

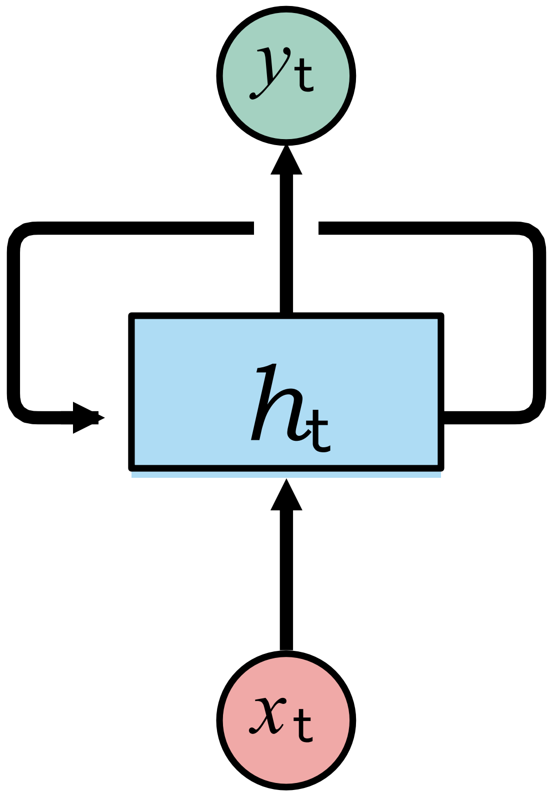
DNN: Universal Approximation Theorem (UAT)



proven by George Cybenko in 1989

Core idea of Recurrent Neural Networks (RNNs) RNNs

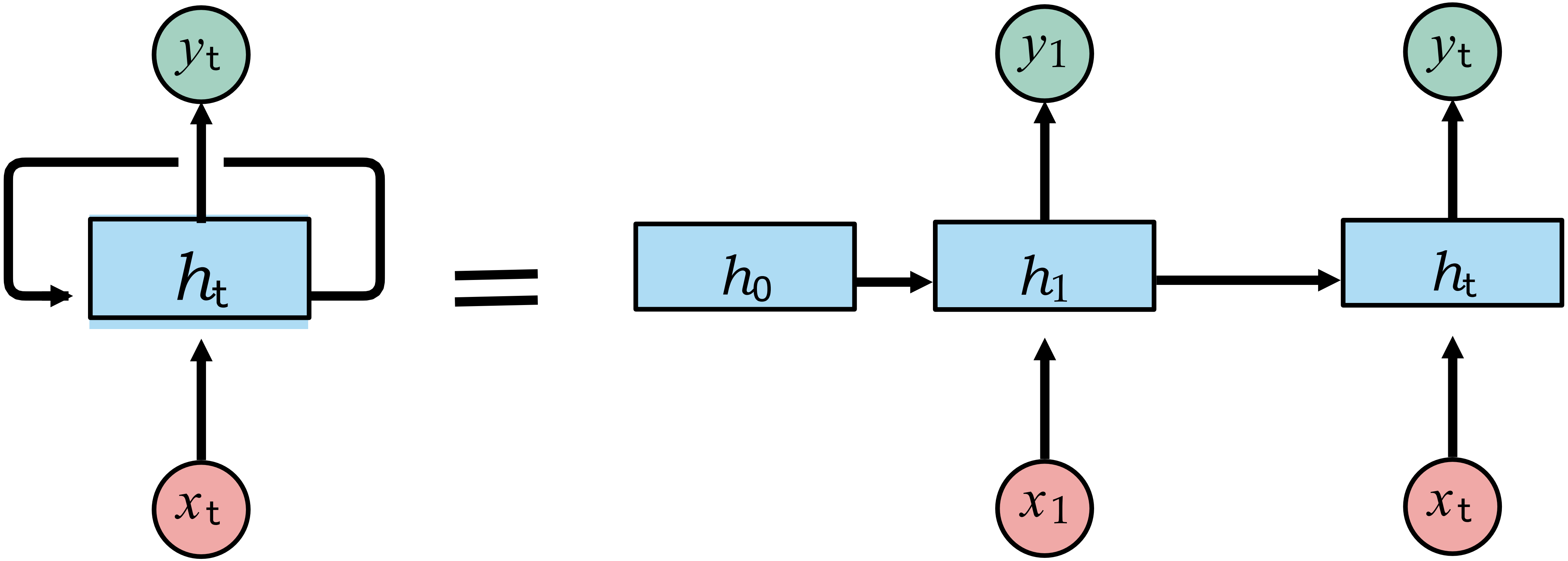
Stateful computation



$$y_t, h_t = f(x_t, h_{t-1})$$

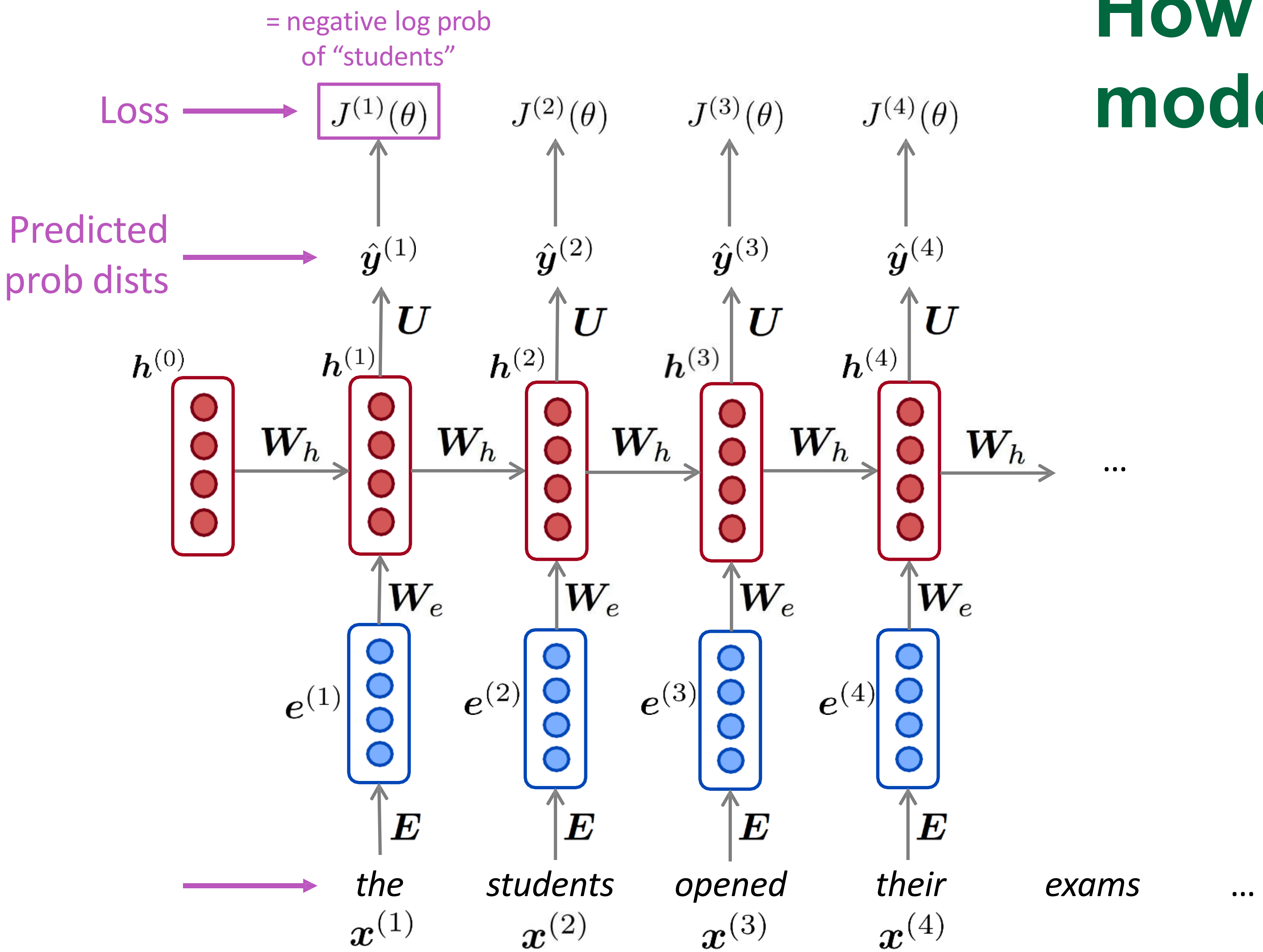
Core idea of RNNs …

Stateful computation

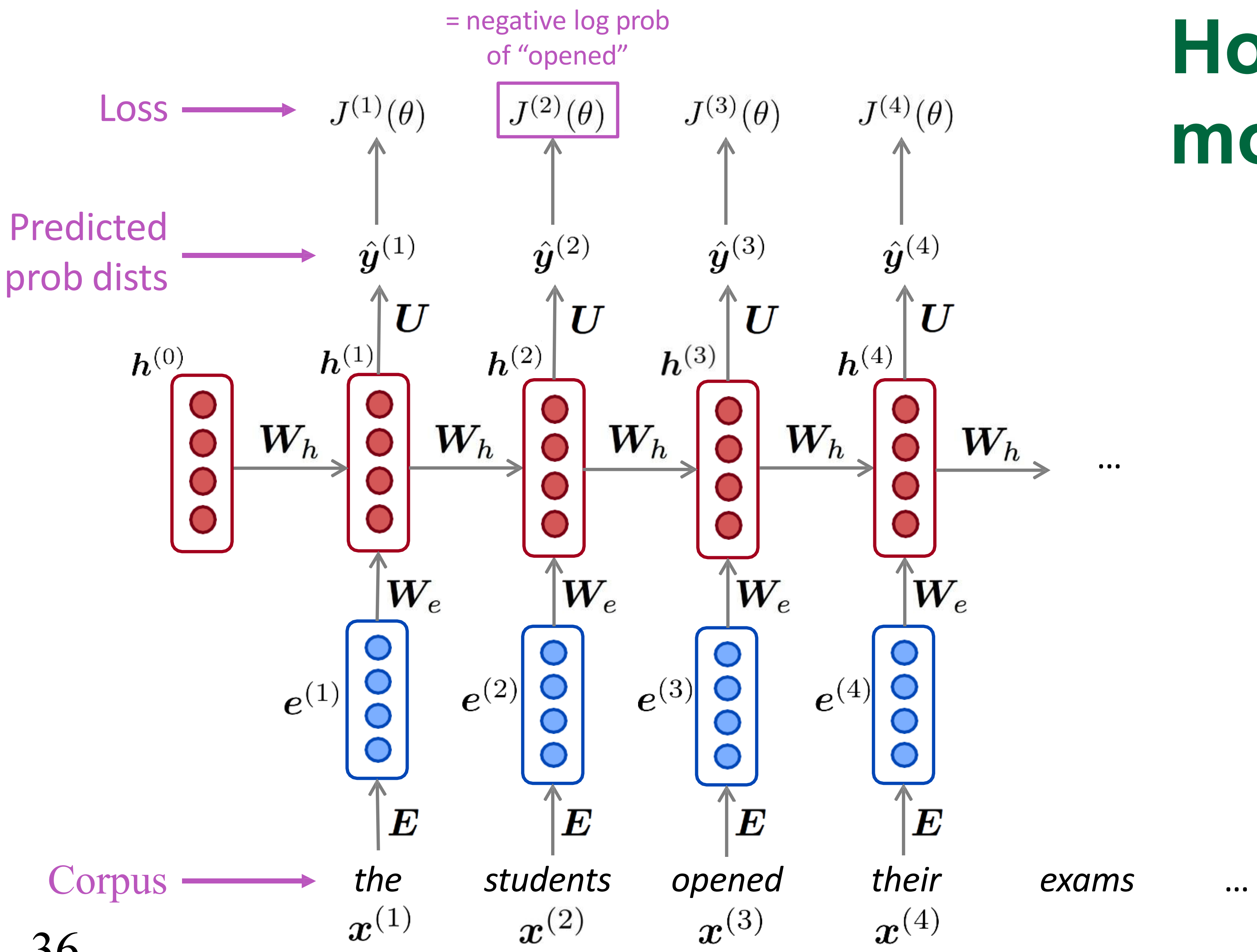


$$h_t = W_x x_t + W_h h_{t-1} + b$$

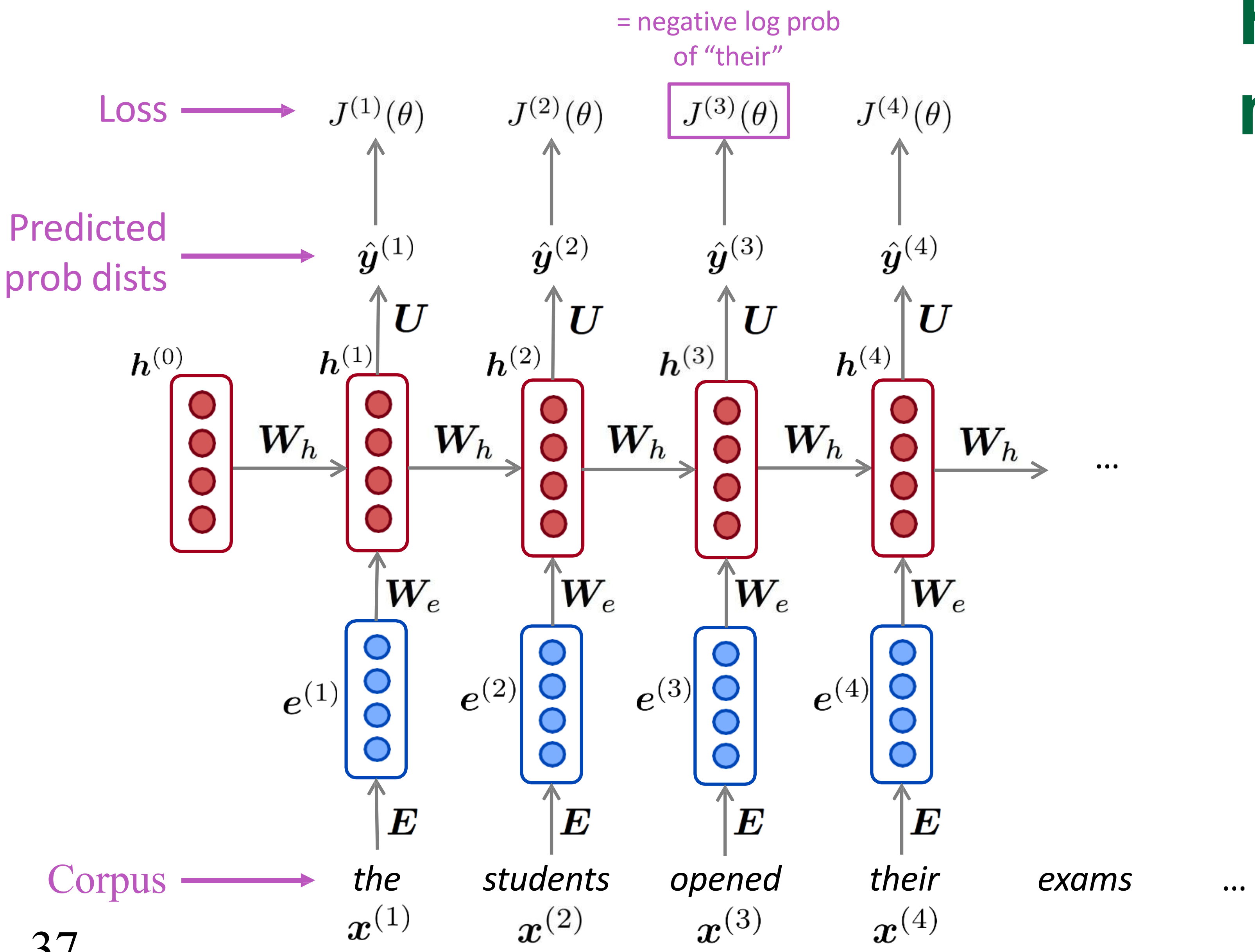
How we train the model



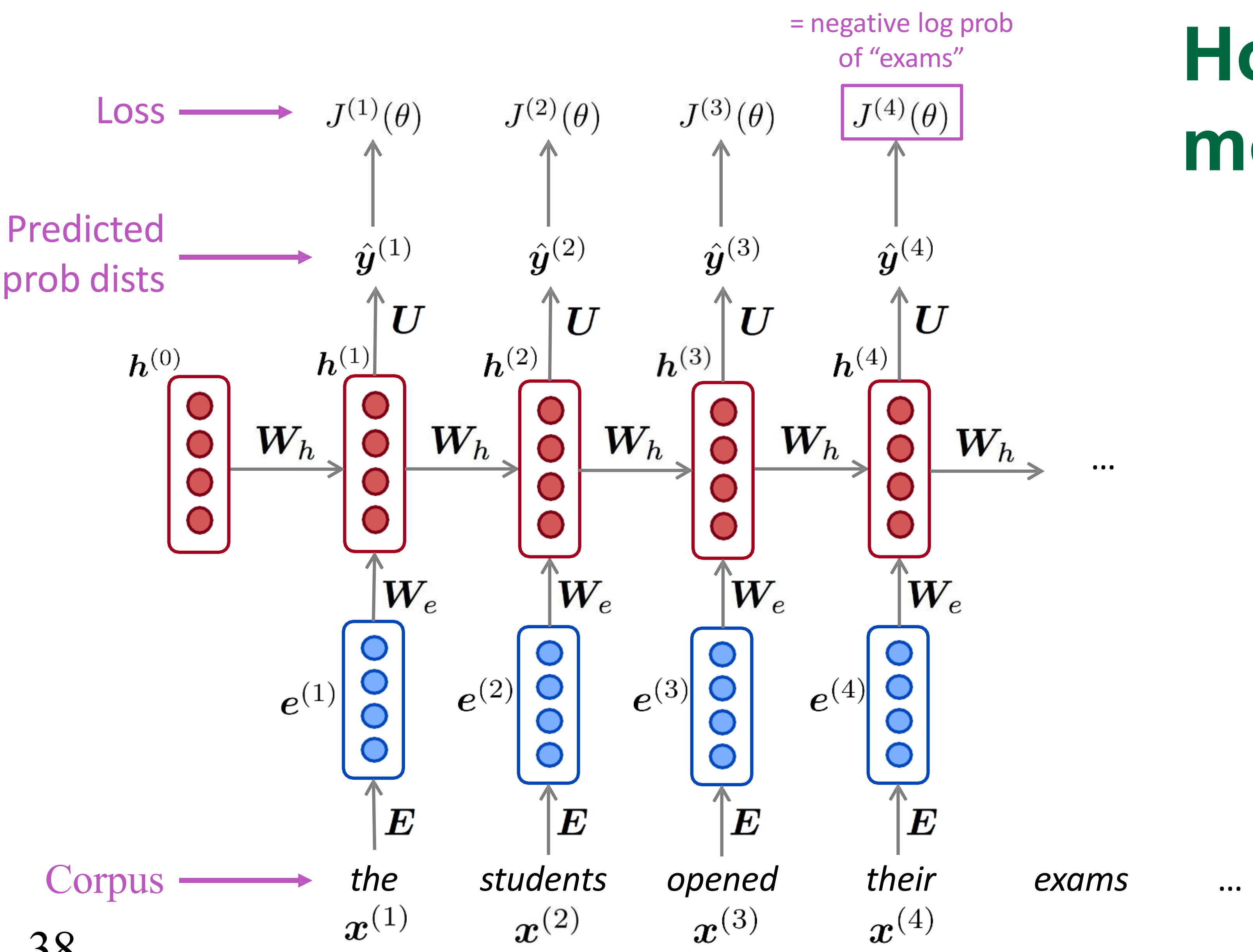
How we train the model



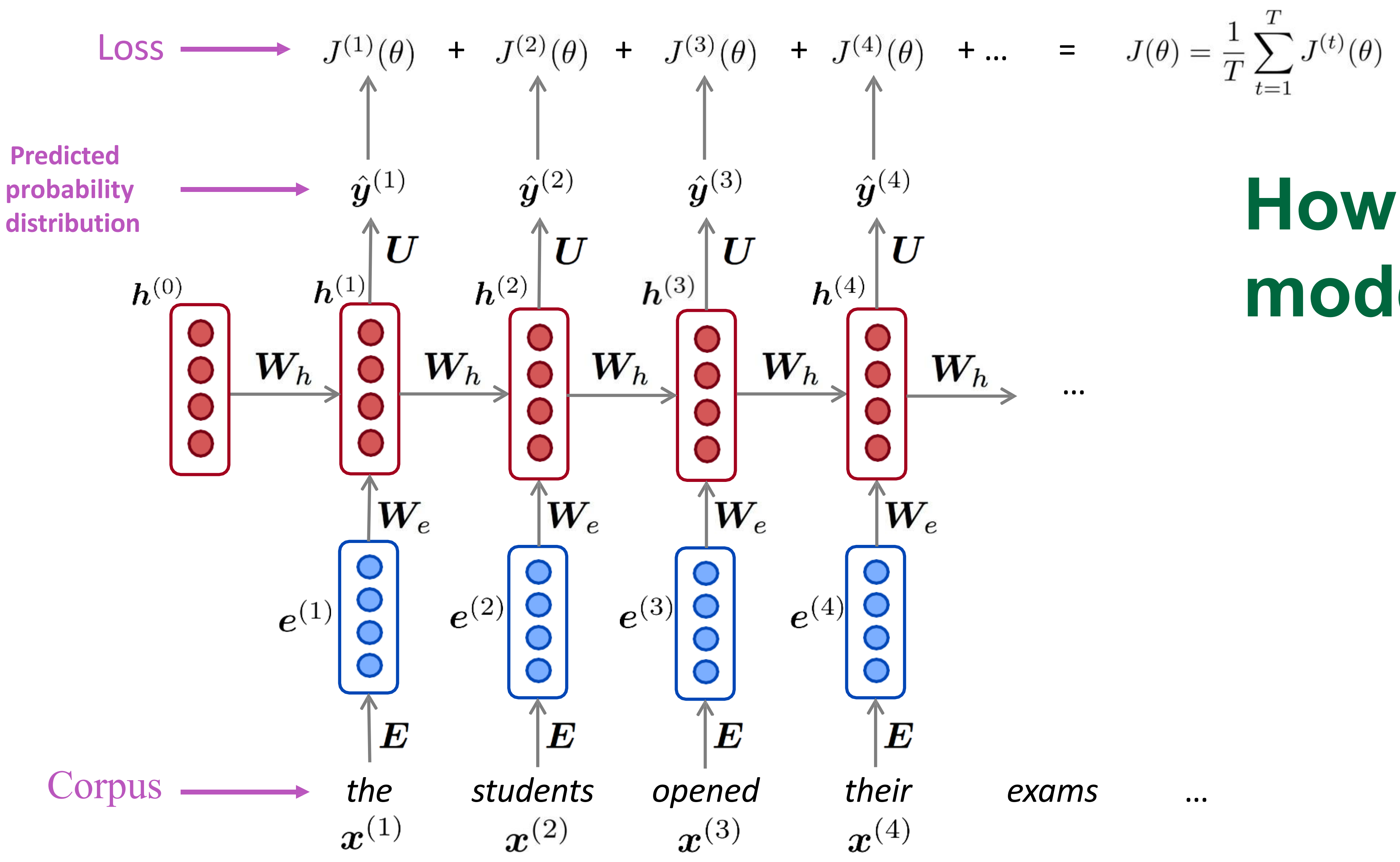
How we train the model



How we train the model



How we train the model



Language Model: RNN

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

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

hidden states

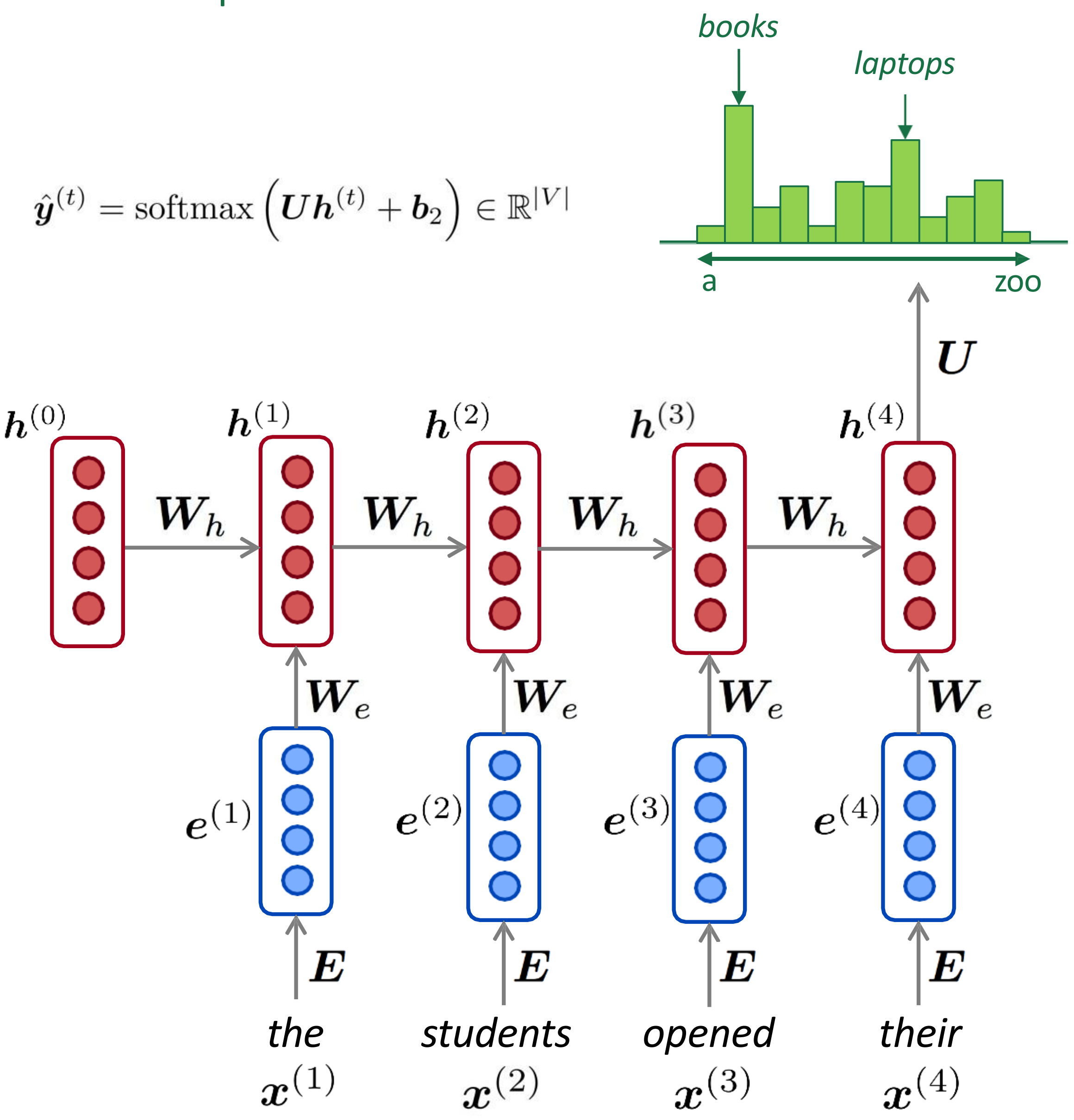
$$\boldsymbol{h}^{(t)} = \sigma\left(\boldsymbol{W}_h\boldsymbol{h}^{(t-1)} + \boldsymbol{W}_e\boldsymbol{e}^{(t)} + \boldsymbol{b}_1\right)$$

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

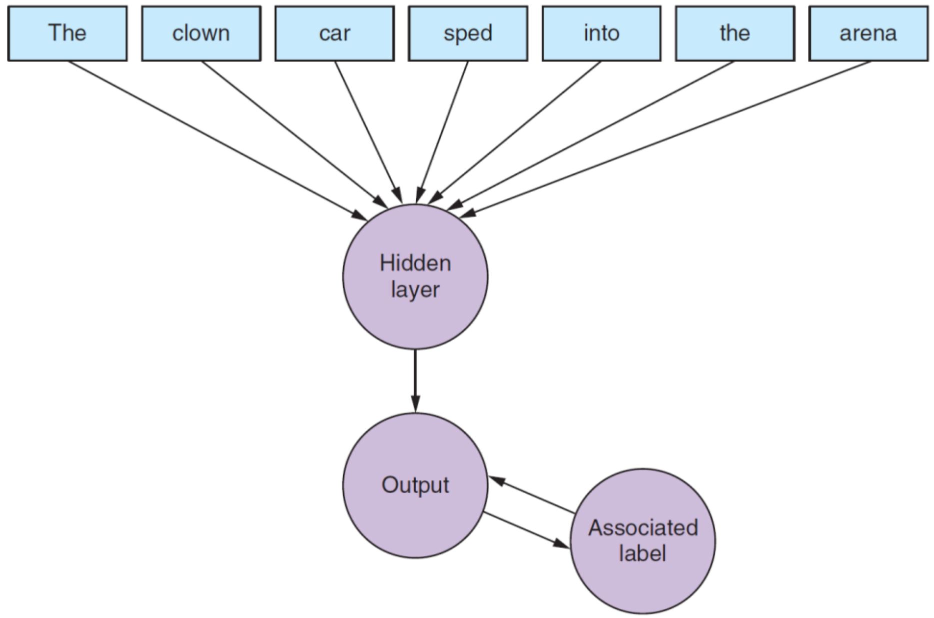
word embeddings

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

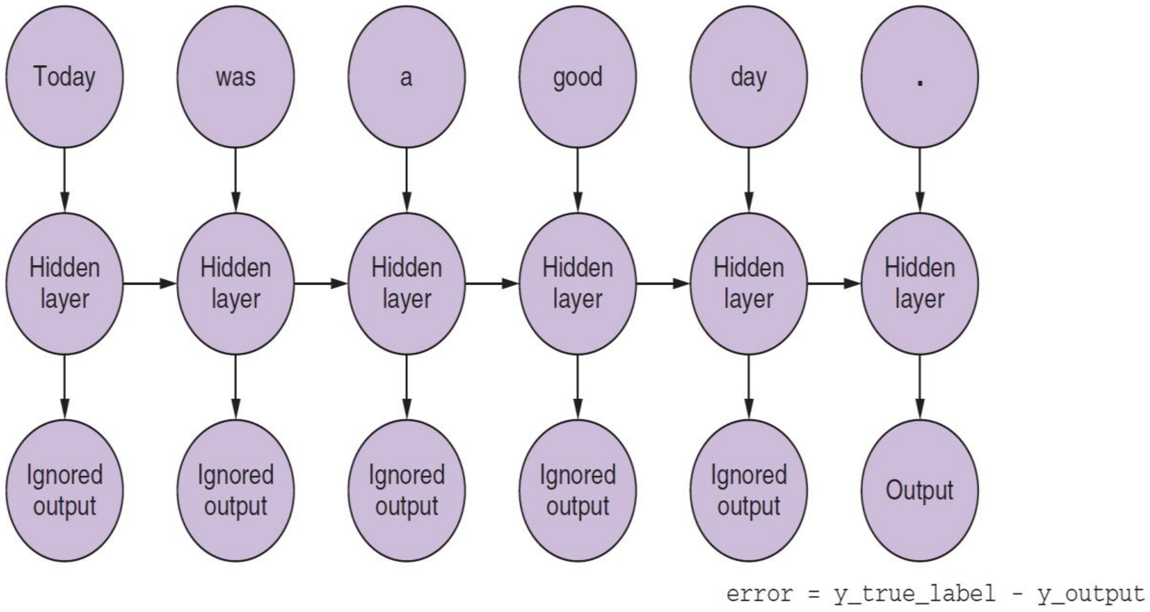
$$\boldsymbol{x}^{(t)} \in \mathbb{R}^{|V|}$$



Difference between NN and RNN



Traditional NN for LM



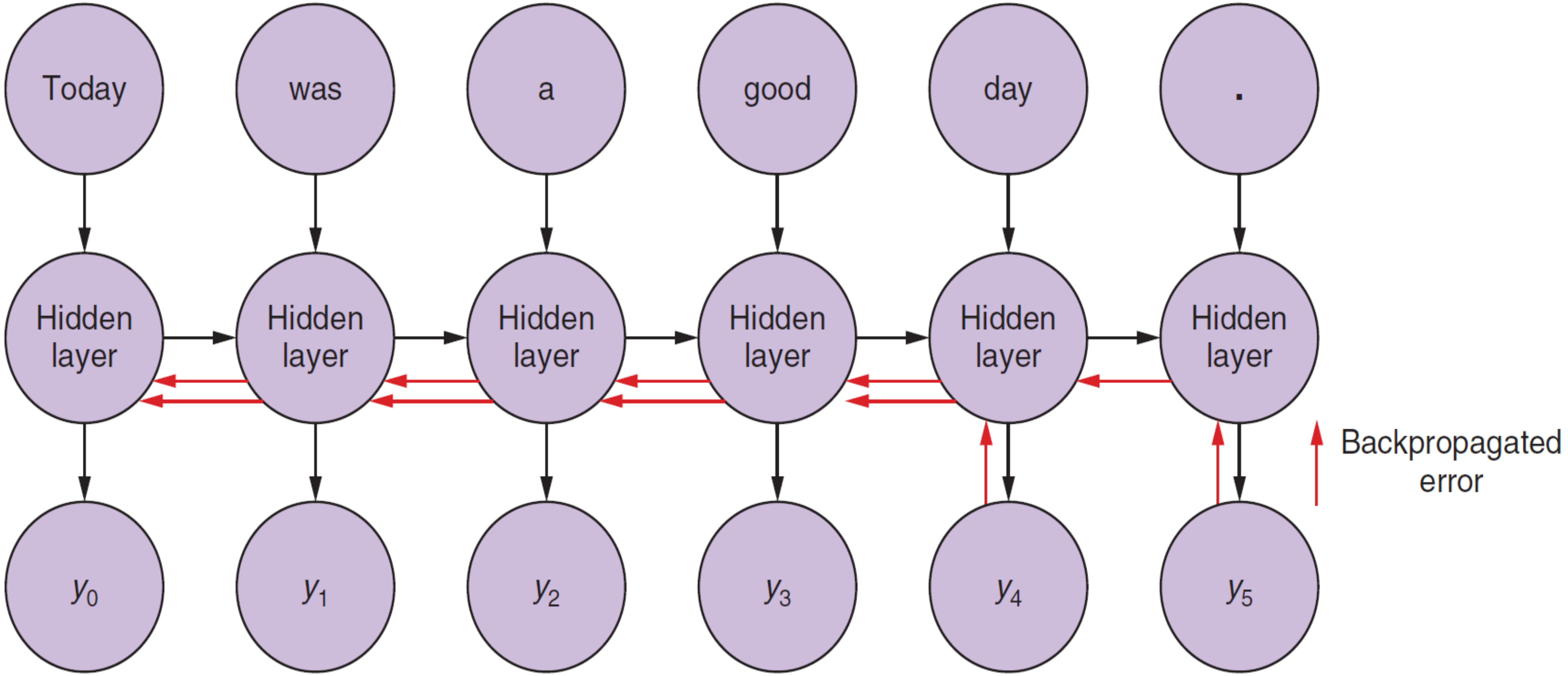
RNN for LM

Image source: NLP in Action text book, O'Reilly

Fun With RNN Language Model

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

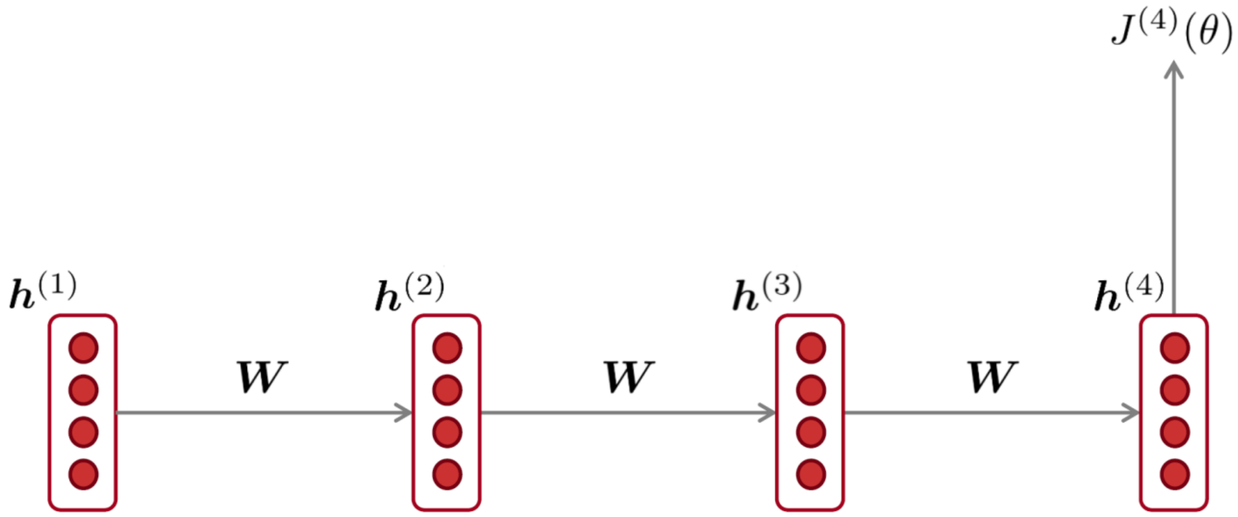
Back Propagation in RNN



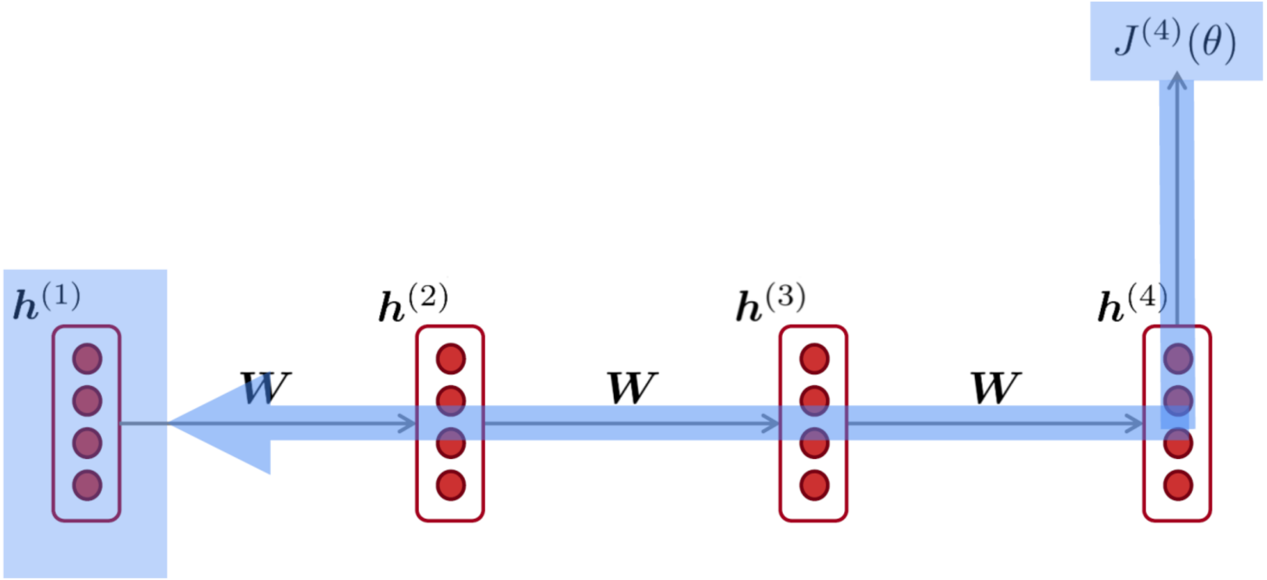

```
error = sum([y_true_label[i] - y[i] for i in range(6)])
```

Backpropagation Through Time (BPTT).

RNN Vanishing Gradient Intuition

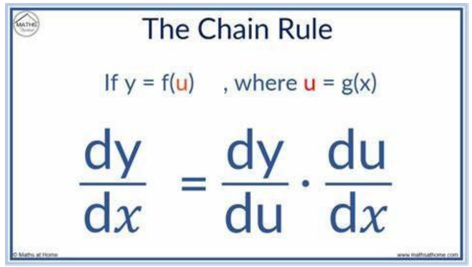


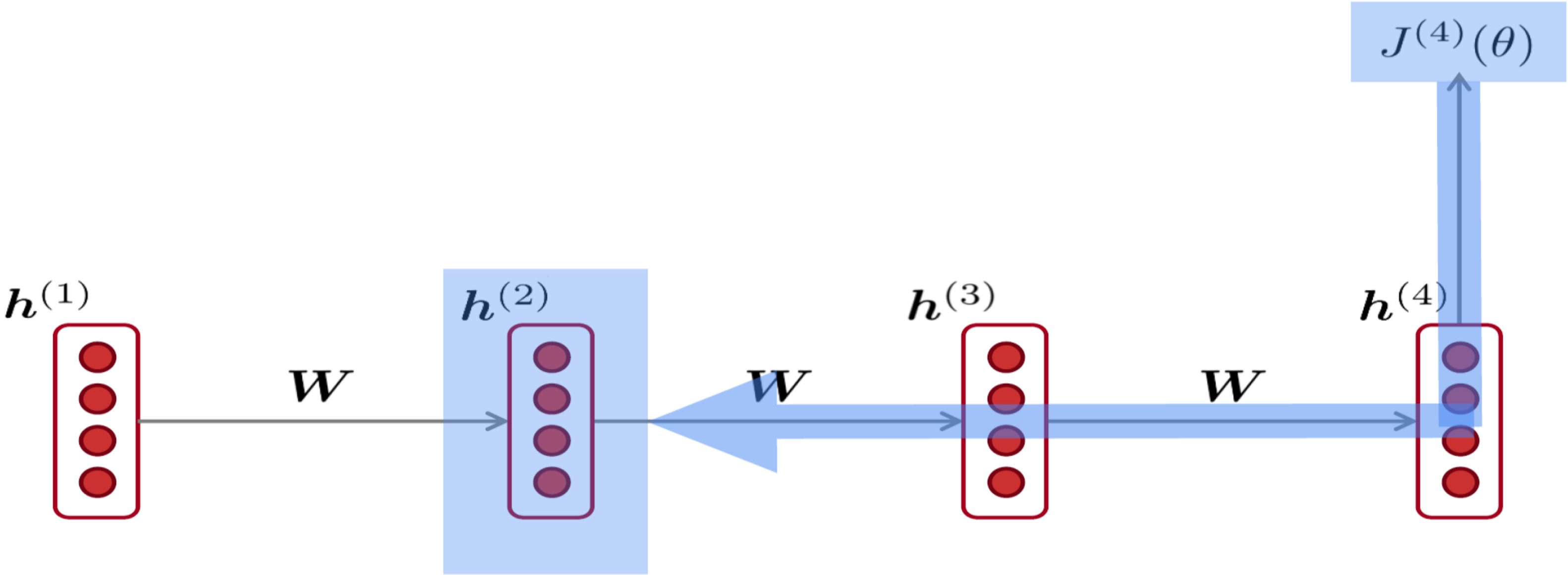
Vanishing gradient intuition



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

Vanishing gradient intuition

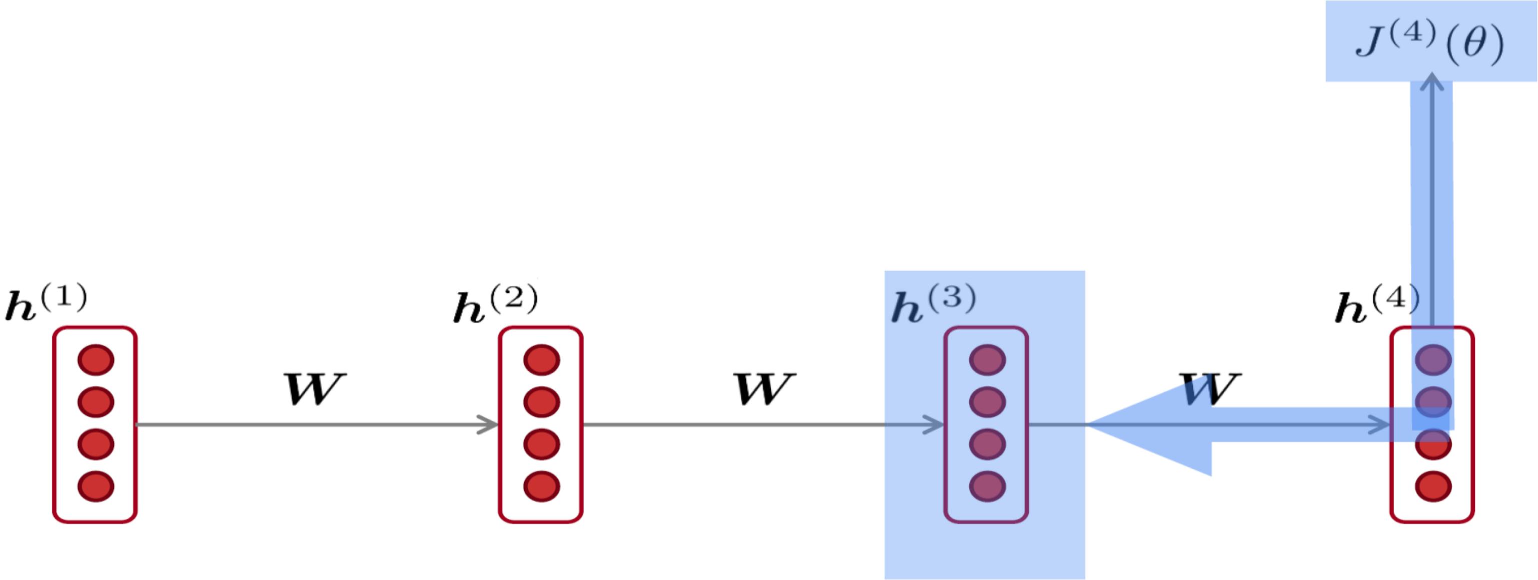




$$\frac{\partial J^{(4)}}{\partial \boldsymbol{h}^{(1)}} = \frac{\partial \boldsymbol{h}^{(2)}}{\partial \boldsymbol{h}^{(1)}} \times \frac{\partial J^{(4)}}{\partial \boldsymbol{h}^{(2)}}$$

chain rule!

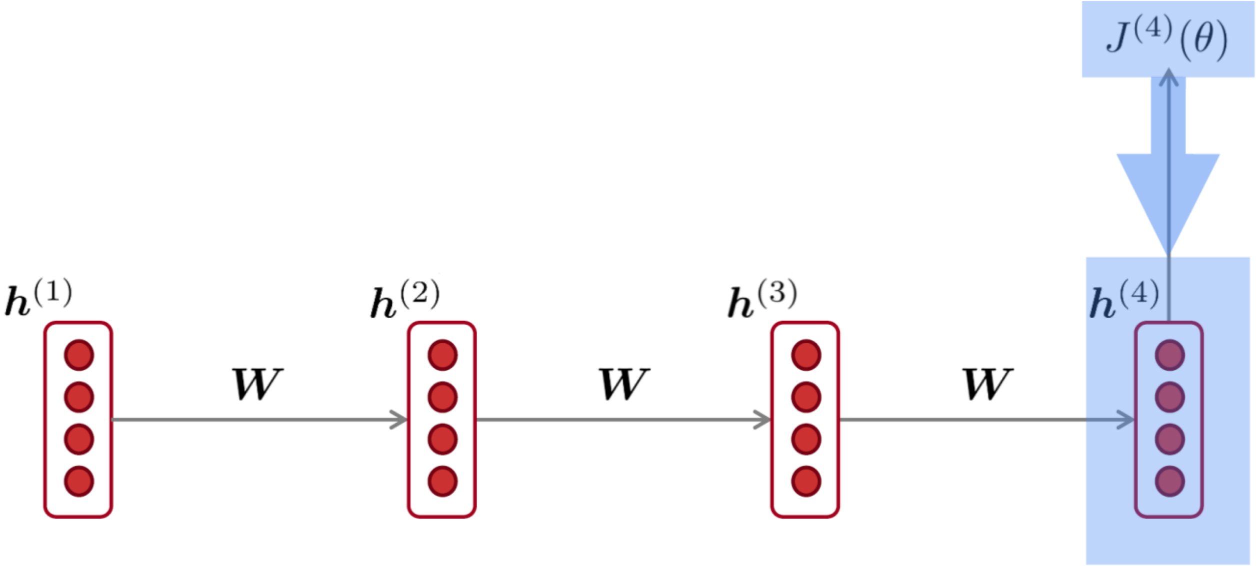
Vanishing gradient intuition



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

chain rule!

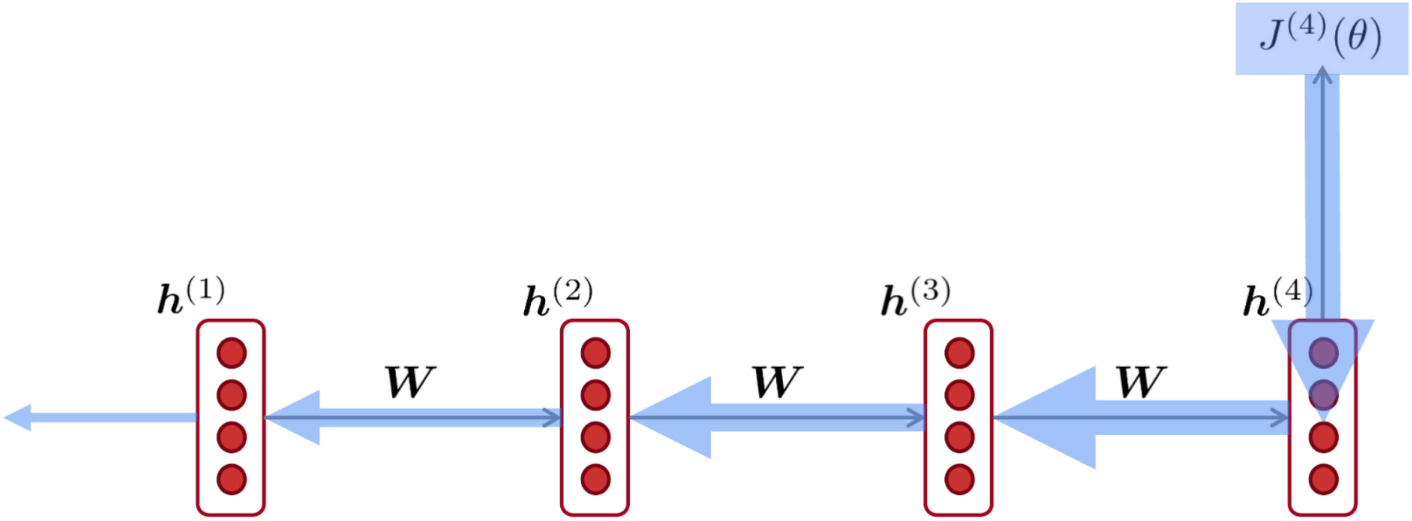
Vanishing gradient intuition



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

chain rule!

Vanishing gradient intuition



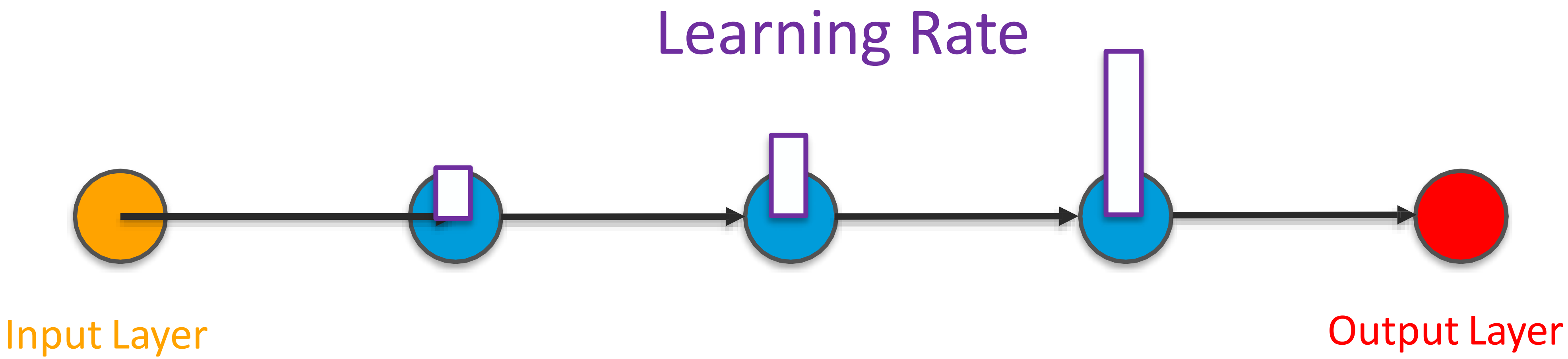
$$\frac{\partial J^{(4)}}{\partial \boldsymbol{h}^{(1)}} = \boxed{\frac{\partial \boldsymbol{h}^{(2)}}{\partial \boldsymbol{h}^{(1)}}} \times \boxed{\frac{\partial \boldsymbol{h}^{(3)}}{\partial \boldsymbol{h}^{(2)}}} \times \boxed{\frac{\partial \boldsymbol{h}^{(4)}}{\partial \boldsymbol{h}^{(3)}}}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

What happens if these are small?

<u>Vanishing gradient problem:</u>
When these are small, the gradient signal gets smaller and smaller as it backpropagates further

Why Vanishing Gradients is Problem

Vanishing gradients occur when the values of a gradient are too small and the model stops learning or takes way too long as a result



Vanishing Gradients Problem…

Example

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-------------------

RNN-LM needs to **model the dependency** between *“tickets”* on the 7th step and the target word *“tickets”* at the end

Long Short-Term Memory (LSTM)

- **Hochreiter & Schmidhuber (1997)** solved the problem of getting an RNN to remember things for a long time.
  - At each timestep t, the LSTM maintains two key components:
    - **Hidden state** – captures short-term dependencies.
    - **Cell state** – acts as a memory unit, storing long-term information.

Long Short-Term Memory (LSTM)

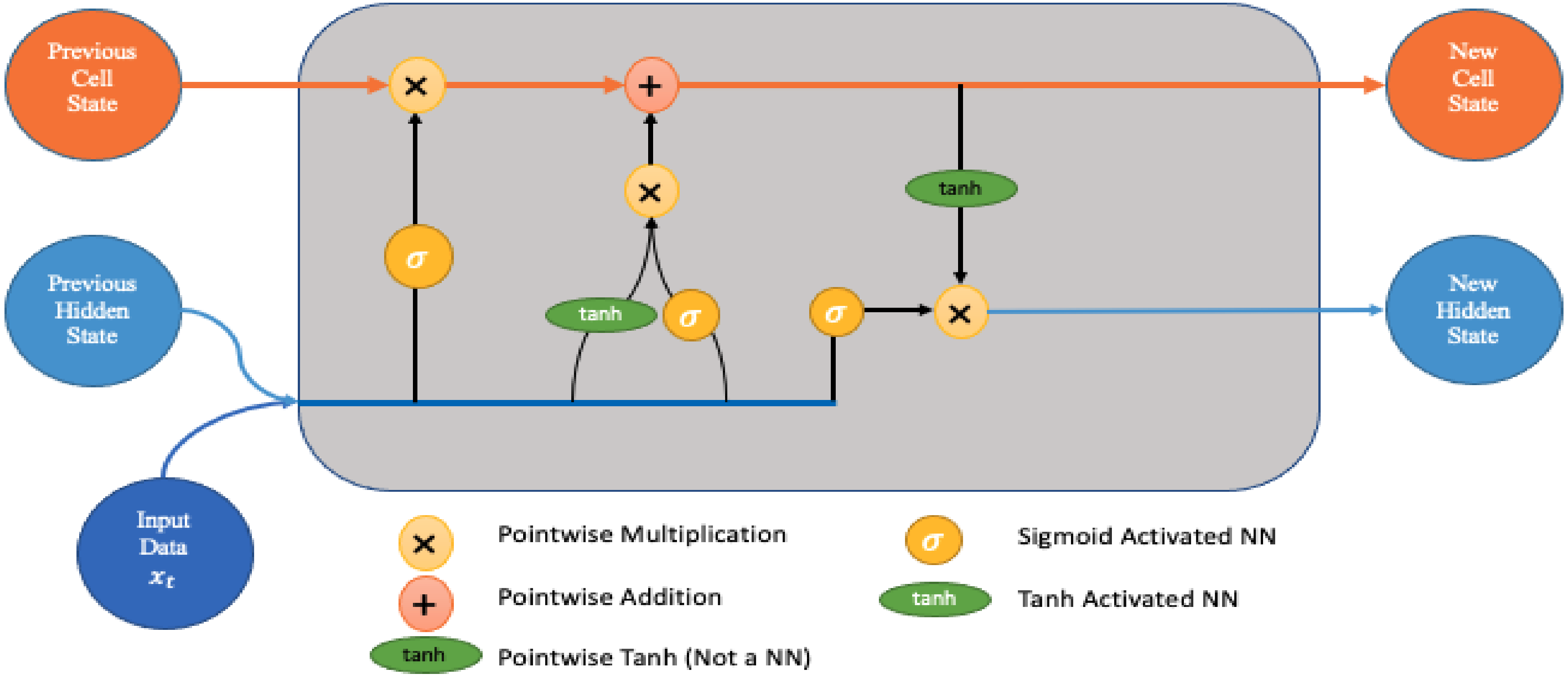
Key Concepts:

•Unlike standard RNNs, LSTMs can **control the flow of information** through three specialized gates:

- **Forget gate** – decides which information to erase.
- **Input gate** – determines what new information should be stored.
- **Output gate** – regulates what information is passed to the next timestep.

•Each gate is represented as a vector of size n and can take values between **0** (**closed**) and **1** (**open**) dynamically, based on the current context.

Long Short -Term Memory (LSTM)



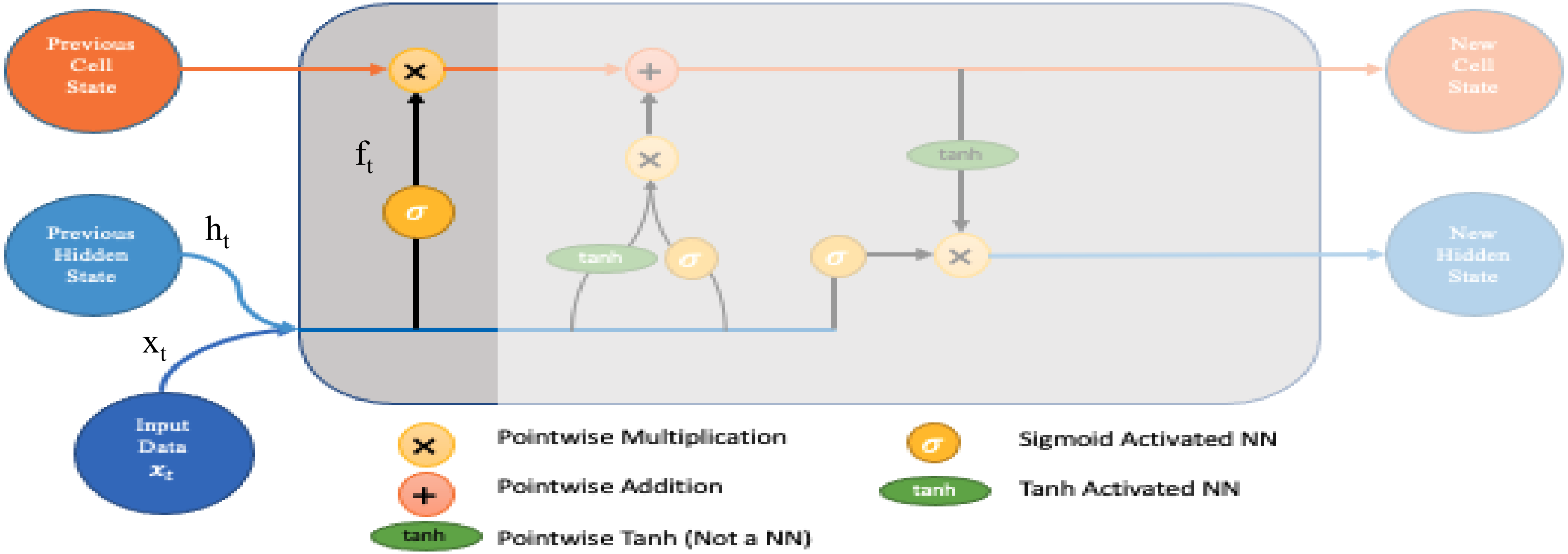
LSTM at time stamp T

Image source: https://towardsdatascience.com/lstm-networks-a-detailed-explanation-8fae6aefc7f9

Long Short Term Memory (LSTM): step 1

Forget Gate:

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

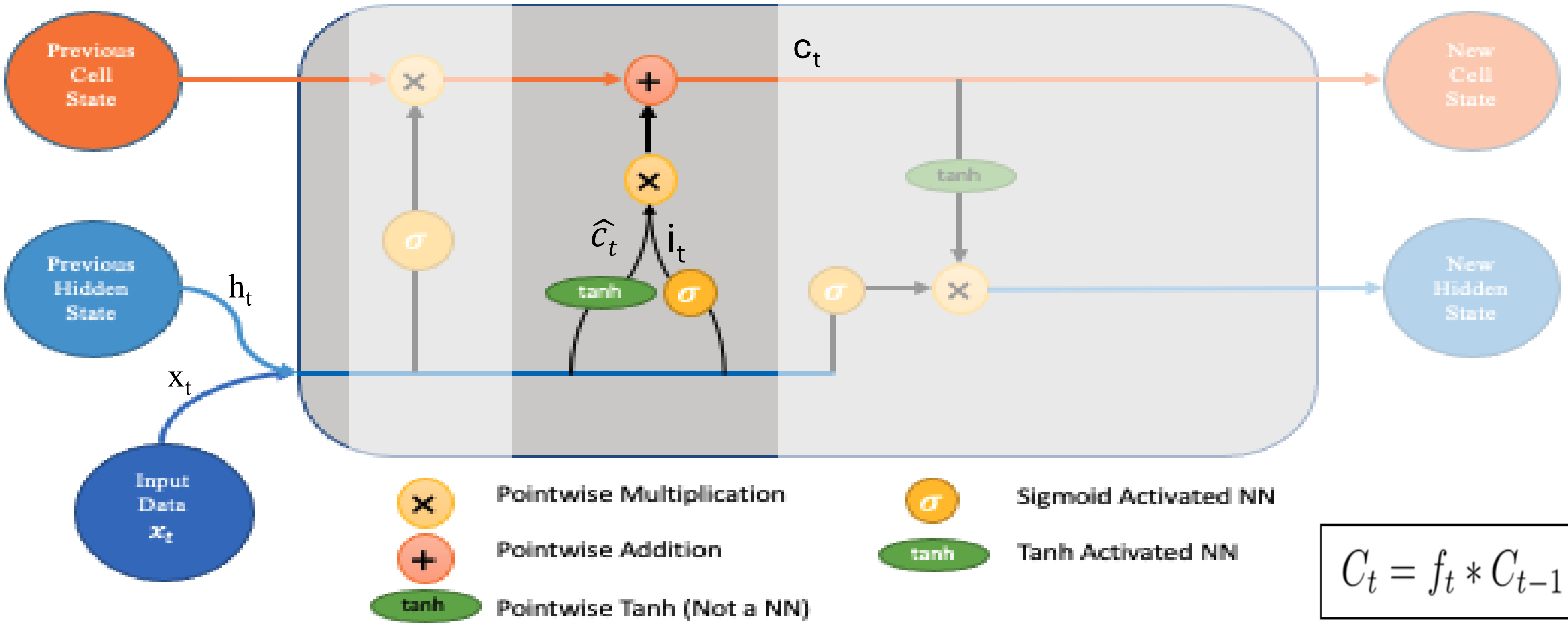


Forget gate: decide what parts of old state to forget

Long Short Term Memory (LSTM):step2

Input Gate:

$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \ + \ b_i\right)$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \ + \ b_C)$$



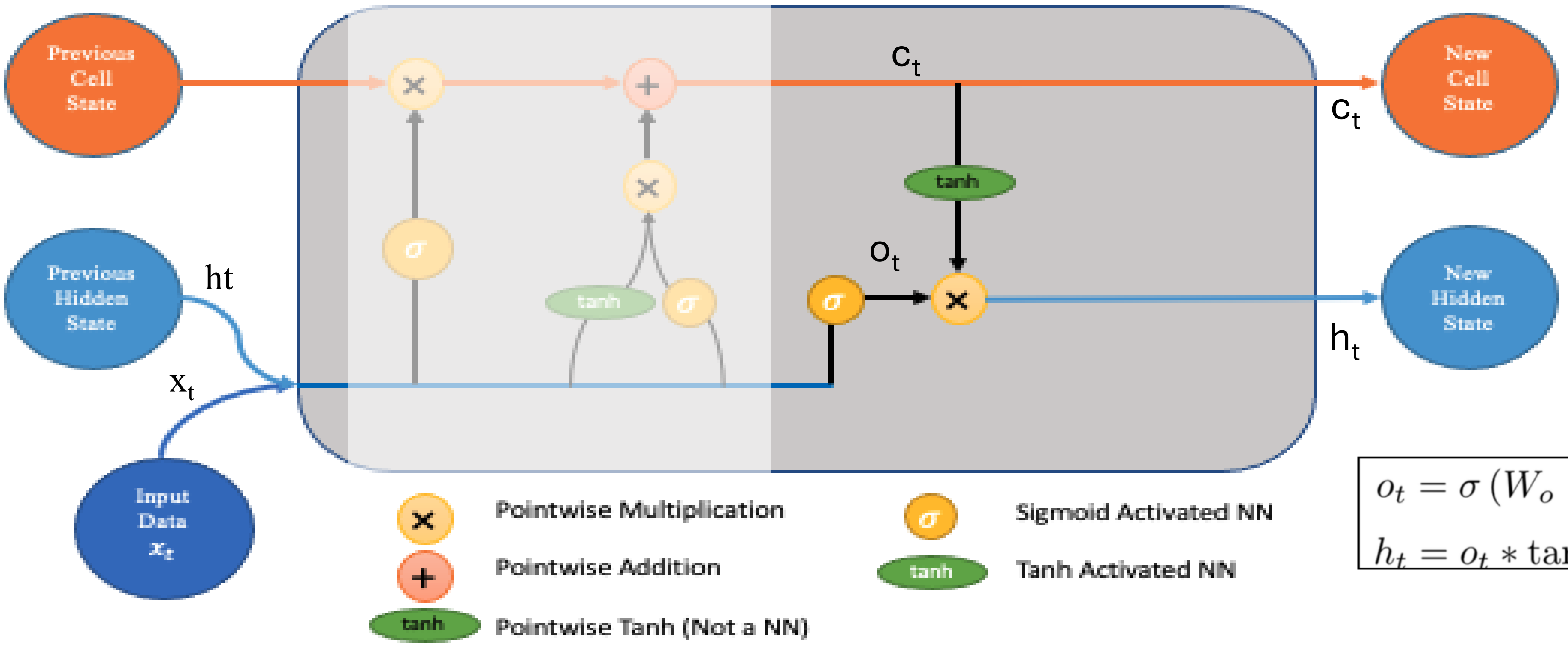
$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

Input gate: decide how to update the cell state

Long Short Term Memory (LSTM):step3

Output Gate:

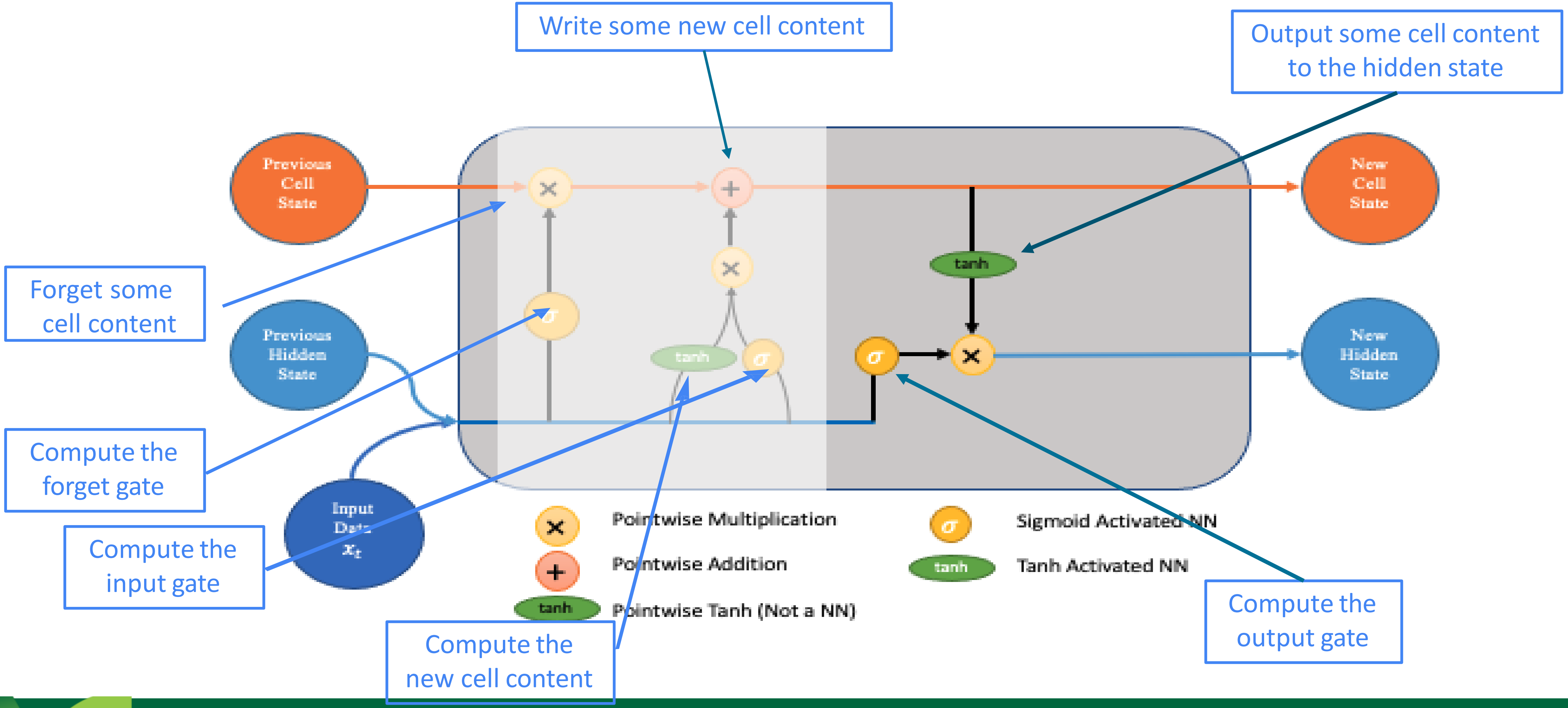
$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$



$$o_t = \sigma\left(W_o\left[h_{t-1}, x_t\right] + b_o\right)$$
$$h_t = o_t * \tanh\left(C_t\right)$$

Finally, decide what to output as hidden state

Long Short Term Memory (LSTM)



LSTM Great resources

- https://colah.github.io/posts/2015-08-Understanding-LSTMs/

Keras – Simplifying LSTMs in Python

Keras is a Python package that makes building and training TensorFlow neural networks really simple. We'll be working with the "Sequential" model which lets you add layers one at a time. As an example, let's see how to build a 1-layer LSTM model with 10 hidden nodes.

```
from keras.models import Sequential
from keras.layers import Dense, Activation, LSTM

model = Sequential()
model.add(LSTM(10, input_shape=(TIMESTEPS, FEATURE_LENGTH)))
model.add(Dense(NUMBER_OF_OUTPUT_NODES))
model.add(Activation('softmax'))
```

Evaluating Language Models

- The standard evaluation metric for Language Models is perplexity.

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

Normalized by number of words

Low perplexity → the model predicts the text well

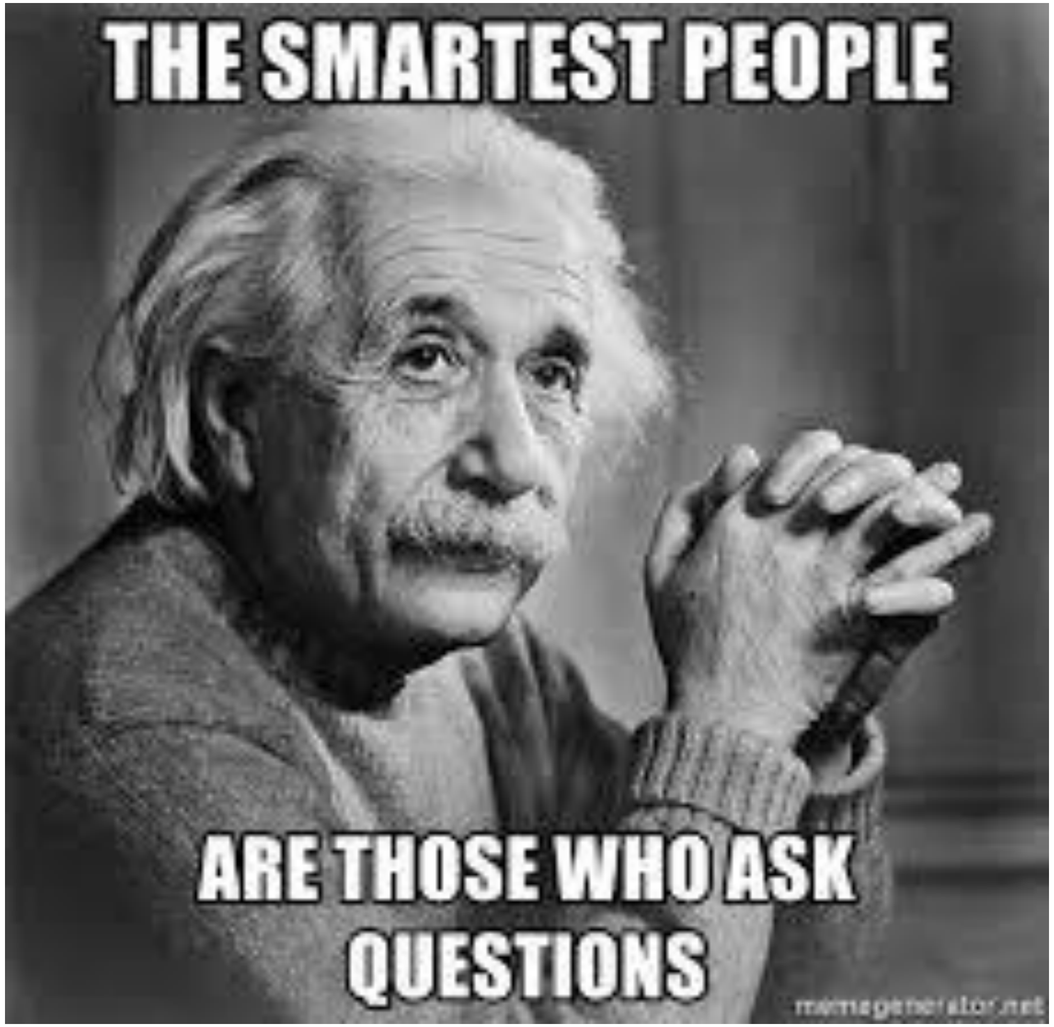
High perplexity → the text is unexpected for the model

Perplexity (PPL) measures **how confused** a language model is when predicting the next word in a sentence.

Summary

- We introduced the concepts of recurrent neural networks and how it can be applied to language problems.
- RNNs can be trained with a straightforward extension of the backpropagation algorithm.
- How LSTM used for text generation
- Applications of LSTM for sequence-to-sequence modeling

Q&A


THE SMARTEST PEOPLE
ARE THOSE WHO ASK QUESTIONS
memegenerator.net