

Deep Learning for Language, Part 1: Recurrent Neural Networks

Robin Jia
USC CSCI 467, Fall 2023
October 3, 2023

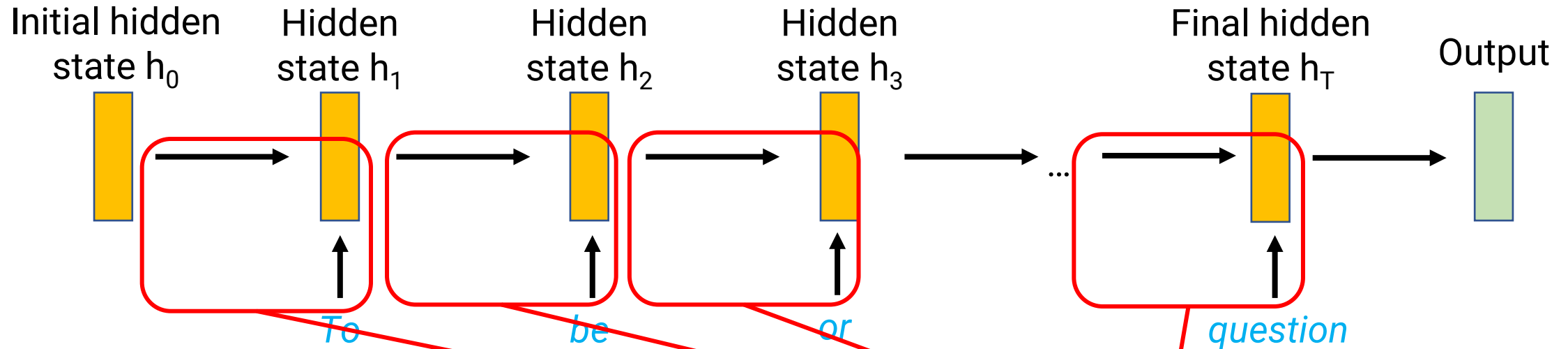
Outline

- Recurrent Neural Networks (RNNs) for sequential data
- Language modeling and Long-range dependencies
- Vanishing gradients and Gated RNNs

Handling textual data

- Images: We assume inputs are fixed dimensional
 - Can crop/rescale as needed
- Text: Inputs are naturally variable-sized!
 - Example 1: *Amazing!*
 - Example 2: *There are many issues with this movie, such as...*
- Challenge: How can we use the **same** set of model parameters to handle inputs of any size?

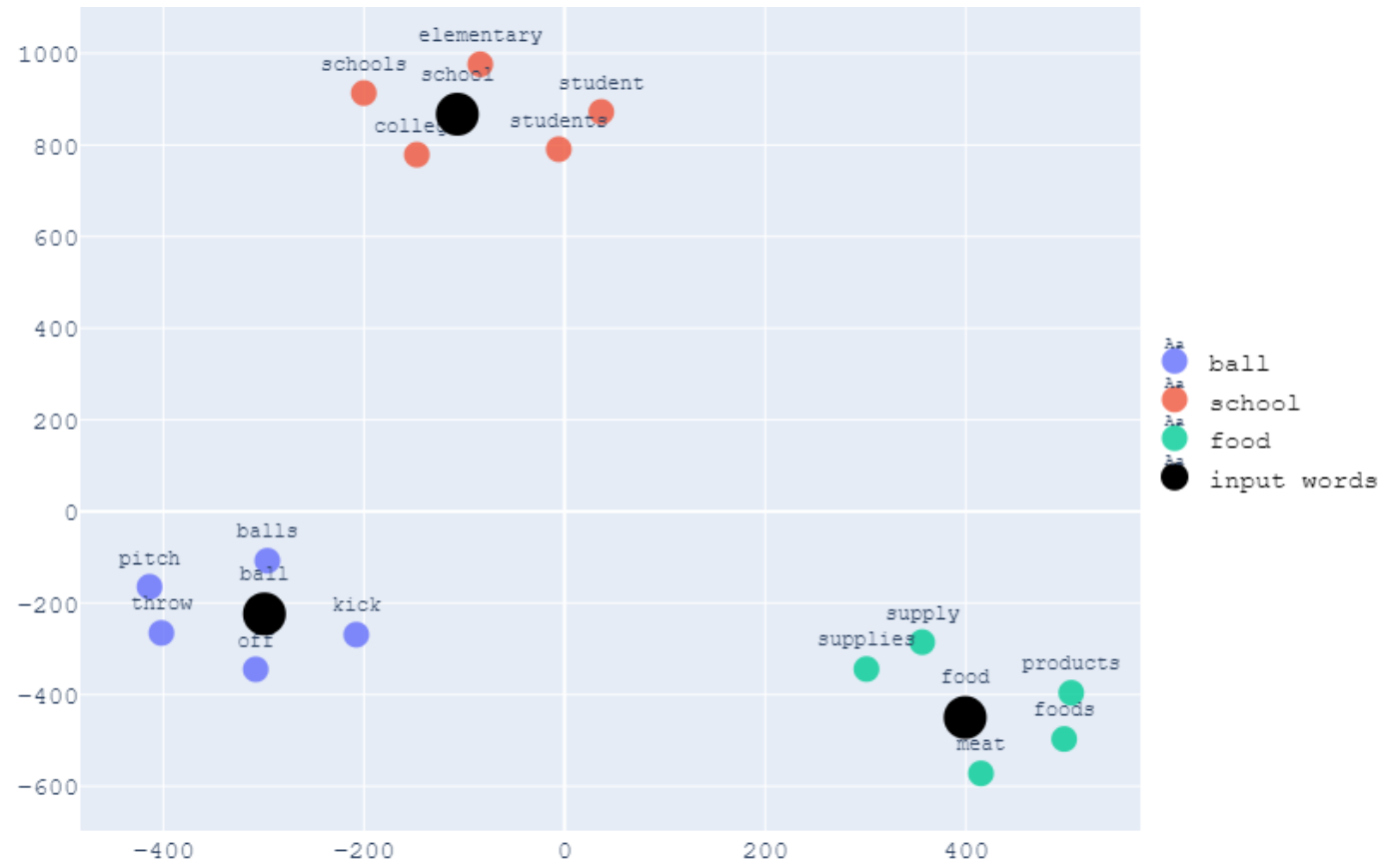
Recurrent Neural Networks (RNNs)



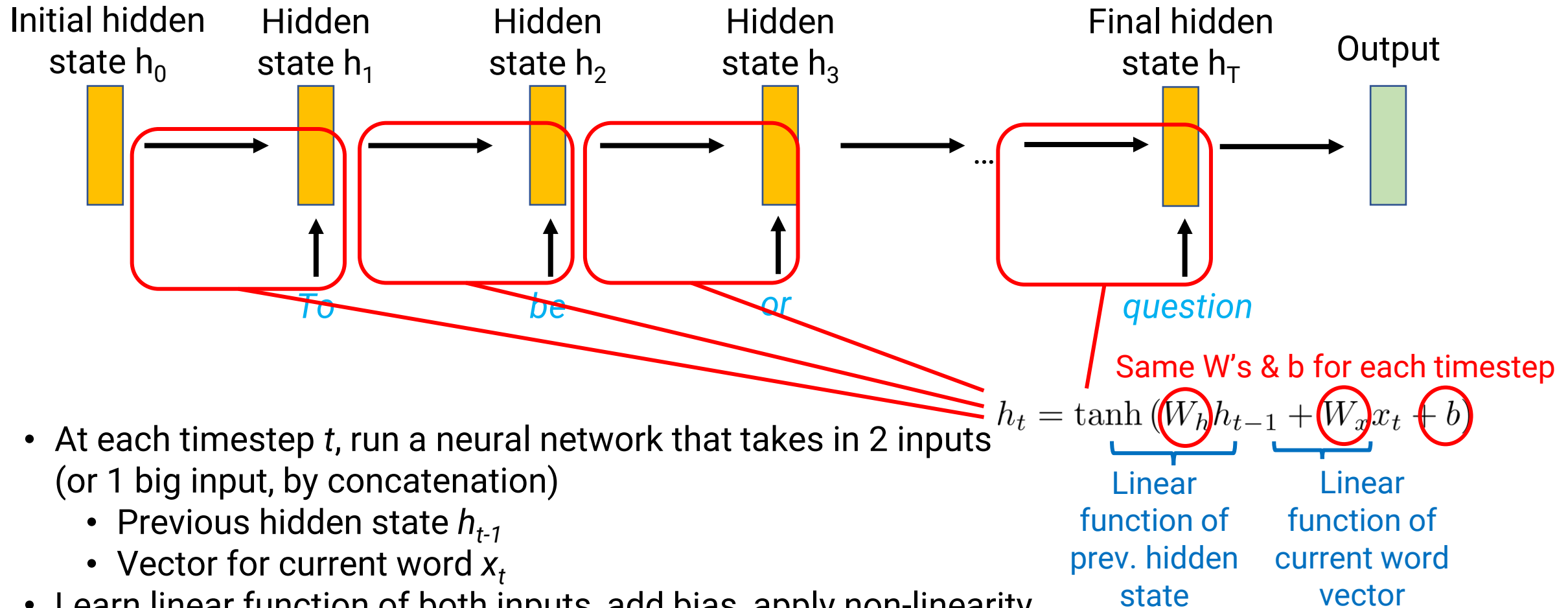
- Idea: Recurrence!
 - “Read” the input one word at a time
 - At each step, update the hidden state of the network
 - **Model parameters to do this update are same for each step**
- Each step is an application of the **same** neural network

Word Embeddings

- How do we “feed” the next word to the RNN?
- Want to learn a vector that represents each word
 - For each word w in vocabulary V , have vector v_w of size d
 - $|V| * d$ parameters needed
- Intuition: Similar words get similar vectors
 - More on learning word vectors later in the class!

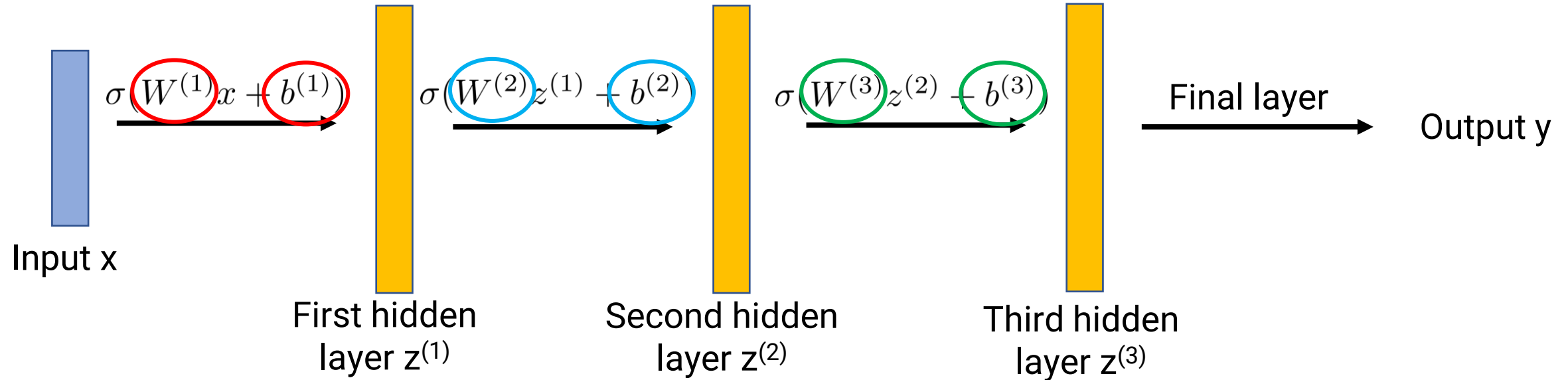


A “Vanilla”/”Elman” RNN



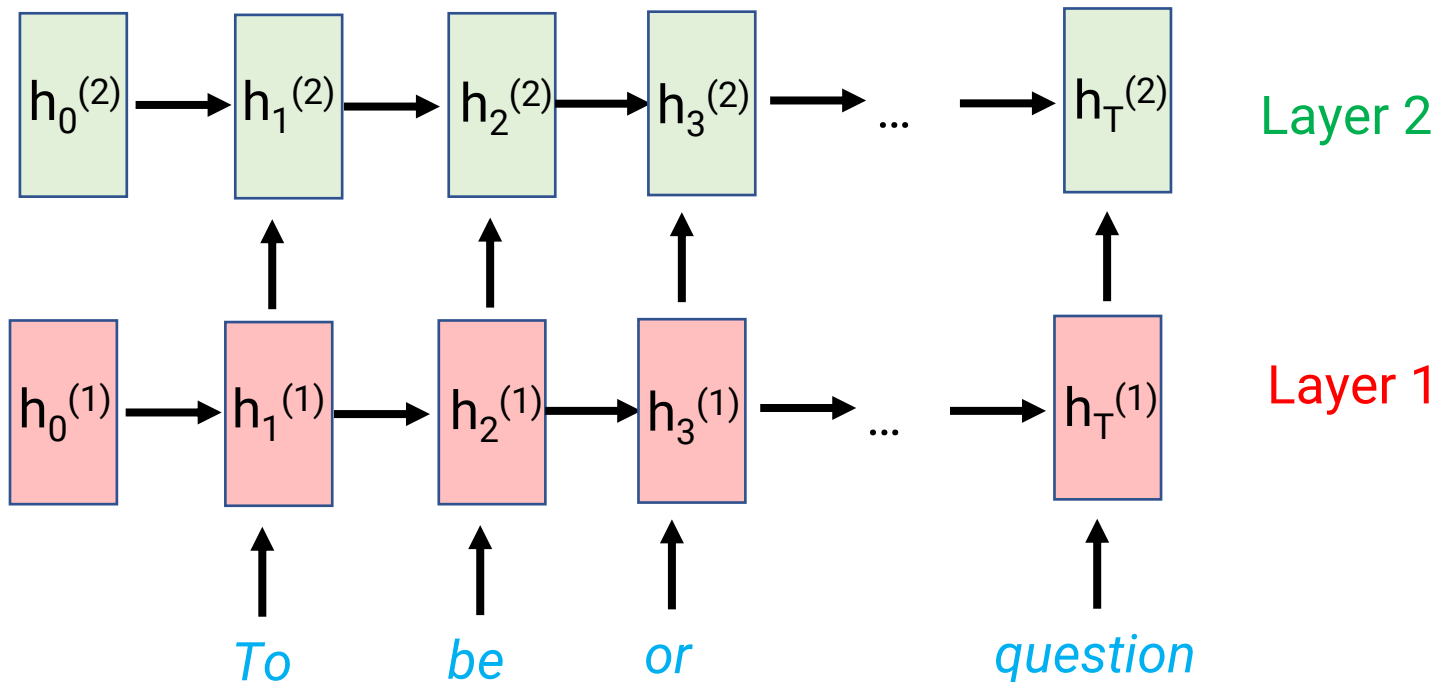
- At each timestep t , run a neural network that takes in 2 inputs (or 1 big input, by concatenation)
 - Previous hidden state h_{t-1}
 - Vector for current word x_t
- Learn linear function of both inputs, add bias, apply non-linearity
- Parameters: Recurrence params (W_h, W_x, b), initial hidden state h_0 , word vectors

Recurrence vs. Depth



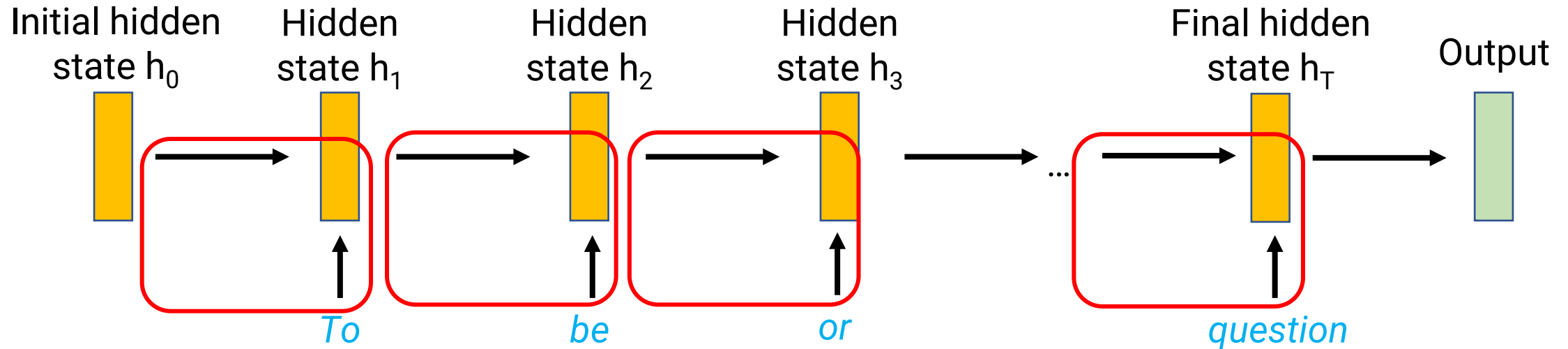
- Deep networks (i.e., adding more layers)
 - Computation graph becomes longer
 - Number of parameters also grows; each step uses new parameters
- Recurrent neural networks
 - Computation graph becomes longer
 - Number of parameters **fixed**; each step uses **same parameters**

Recurrence and Depth



- You can have multiple layers of recurrence too!
 - Left-to-right axis (“time dimension”): Length is size of input, same parameters in each step
 - Top-to-bottom axis (“depth dimension”): Length is depth of network, different parameters in each row

Training an RNN

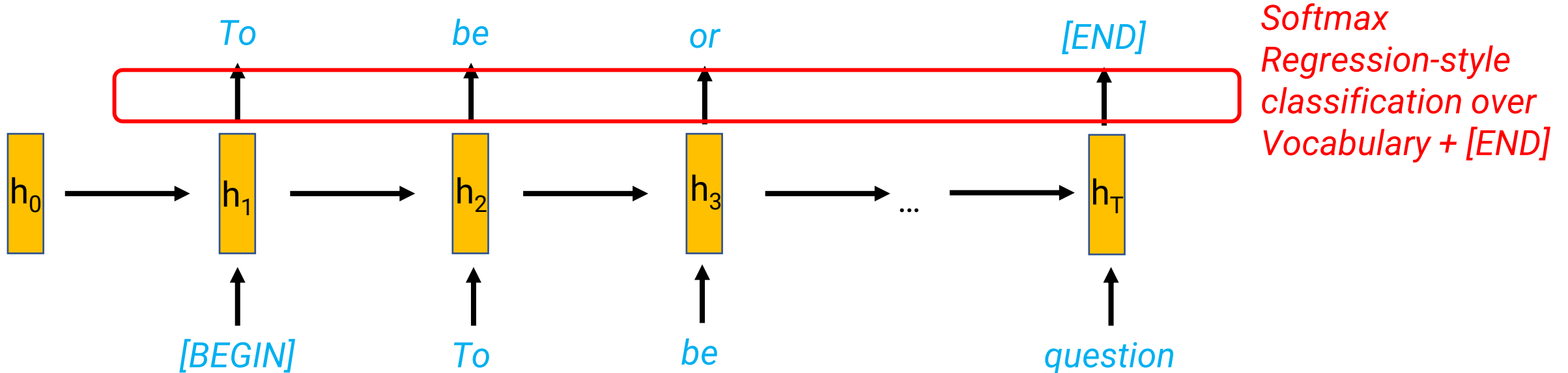


- Same recipe: Backpropagation to compute gradients + gradient descent
- Must backpropagate through whole computation graph
 - “Backpropagation through time”
 - Same weights for recurrence used at every time step; total change to weights is added up for each timestep

Outline

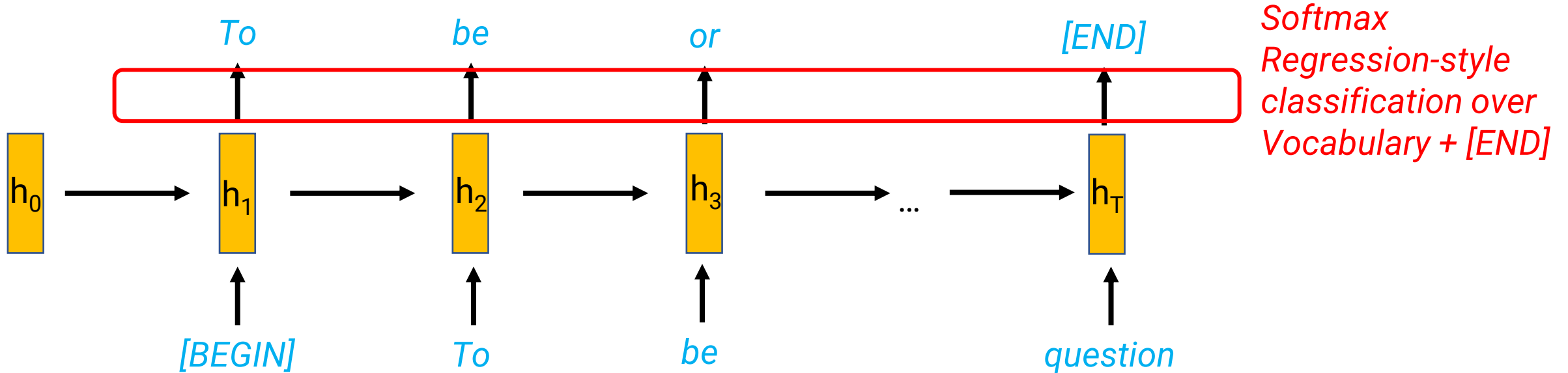
- Recurrent Neural Networks (RNNs) for sequential data
- Language modeling and Long-range dependencies
- Vanishing gradients and Gated RNNs

Language Modeling (“Decoder only”)



- At each step, predict the next word given current hidden state
 - Essentially a softmax regression “head”—takes in hidden state, outputs distribution over Vocabulary + [END]
- Start with special $[BEGIN]$ token (so the first word model generates is first real word)
- One step’s output becomes next step’s input (“autoregressive”)
- To mark end of sequence, model should predict the $[END]$ token
- Called a “Decoder” because it looks at the hidden state and “decodes” the next word

Language Modeling (“Decoder only”)



- Training a language model
 - Input sequence is a real human-written document
 - For each word, compute classification loss (like softmax regression) for model, using the actual human-written next word as the correct “label”
 - Sum up loss for whole document, then backpropagate & update parameters with gradient descent

Long-Range Dependencies

- Every step, you update the hidden state with the current word
- Over time, information from many words ago can easily get lost!
- This means RNNs can struggle to model **long-range dependencies**

The keys to the cabinet ___ (on the table)
plural singular

Long-Range Dependencies

- Every step, you update the hidden state with the current word
- Over time, information from many words ago can easily get lost!
- This means RNNs can struggle to model **long-range dependencies**

The *keys* to the cabinet *are* (on the table)
plural singular

Long-Range Dependencies

- Every step, you update the hidden state with the current word
- Over time, information from many words ago can easily get lost!
- This means RNNs can struggle to model **long-range dependencies**

*The **keys** to the cabinet by the door **are** (on the table)*

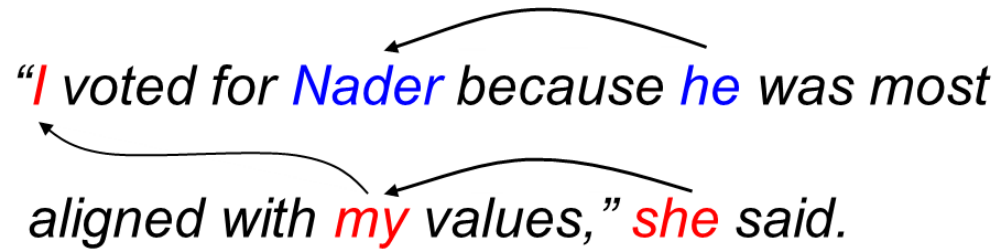
Long-Range Dependencies

- Every step, you update the hidden state with the current word
- Over time, information from many words ago can easily get lost!
- This means RNNs can struggle to model **long-range dependencies**

*The **keys** to the cabinet by the door on the left **are** (on the table)*

Long-Range Dependencies

*“I voted for **Nader** because **he** was most aligned with **my** values,” **she** said.*



The diagram illustrates coreference relationships in the sentence. Two curved arrows are drawn above the text: one from the word "Nader" to the pronoun "he", and another from the pronoun "she" to the pronoun "I".

- “Coreference”: When two words refer to the same underlying person/place/thing
 - Pronouns typically **corefer** to an **antecedent** (something mentioned earlier in the text)
- Coreference relationships can even span multiple sentences

Even longer-range dependencies



- Imagine trying to generate a novel...
 - Same set of characters
 - Characters have to behave in consistent ways
 - Sensible ordering of events

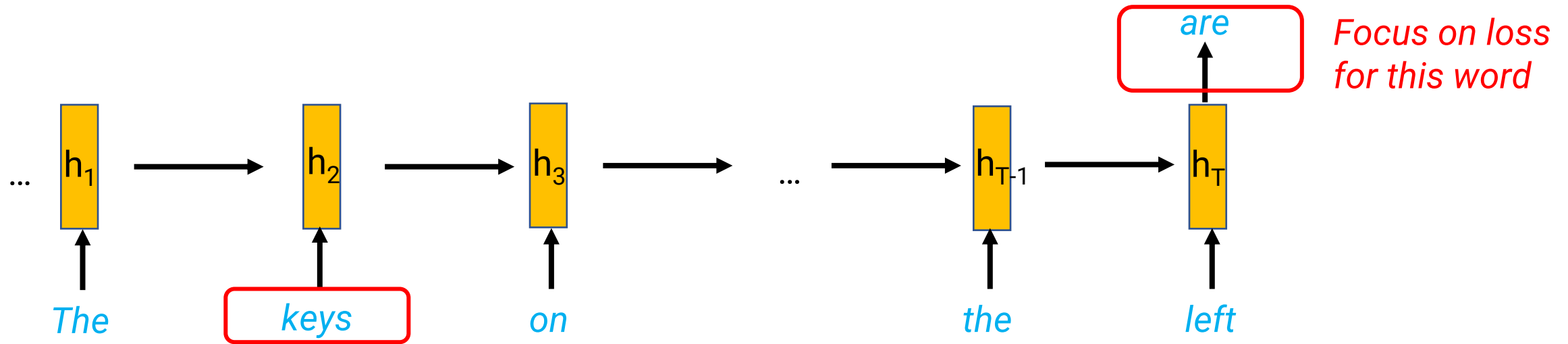
Announcements

- HW2 due this Thursday
- Thursday class: A bit more on RNNs + first half review
- Section Friday: Midterm Review (practice exam + questions)
- Midterm exam next Tuesday, October 10
 - In-class, 80 minutes, one double-sided 8.5x11 sheet of notes
 - Practice exam posted
 - Room assignments (also on Piazza)
 - Last name A-O: LVL 17 (this room)
 - Last name P-Z: THH 116

Outline

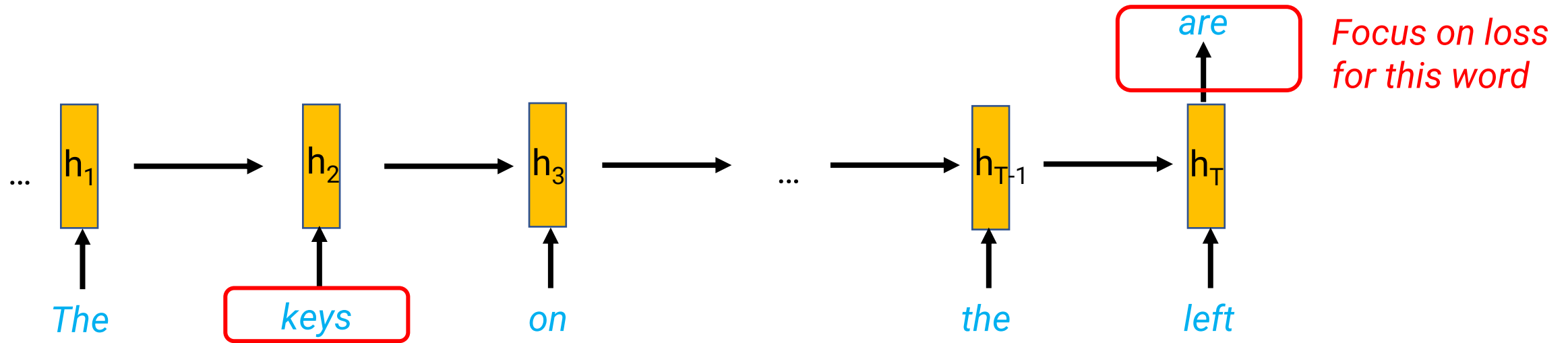
- Recurrent Neural Networks (RNNs) for sequential data
- Language modeling and Long-range dependencies
- **Vanishing gradients and Gated RNNs**

Backpropagation through time, revisited



- Model needs to know that the correct word is *are* because of the word *keys*!
- Let's backpropagate the loss on generating *are* to the word vector parameters for *keys*
 - For simplicity, let's assume all the hidden states are just 1-dimensional
 - Step 1: Compute $\delta\text{Loss}/\delta(h_T)$
 - Step 2: Compute $\delta\text{Loss}/\delta(h_{T-1}) = \delta\text{Loss}/\delta(h_T) * \delta(h_T)/\delta(h_{T-1})$
 - Step 3: Compute $\delta\text{Loss}/\delta(h_{T-2}) = \delta\text{Loss}/\delta(h_T) * \delta(h_T)/\delta(h_{T-1}) * \delta(h_{T-1})/\delta(h_{T-2})$
 - ...
 - Gradient through "keys" hidden state: $\delta\text{Loss}/\delta(h_T) * \delta(h_T)/\delta(h_{T-1}) * \delta(h_{T-1})/\delta(h_{T-2}) * \dots * \delta(h_3)/\delta(h_2)$
 - Gradient through "keys" word vector: $\delta\text{Loss}/\delta(h_T) * \delta(h_T)/\delta(h_{T-1}) * \delta(h_{T-1})/\delta(h_{T-2}) * \dots * \delta(h_3)/\delta(h_2) * \delta(h_2)/\delta(x_2)$

The Vanishing Gradient Problem



- Gradient through “keys” word vector: $\delta\text{Loss}/\delta(h_T) * \delta(h_T)/\delta(h_{T-1}) * \delta(h_{T-1})/\delta(h_{T-2}) * \dots * \delta(h_3)/\delta(h_2) * \delta(h_2)/\delta(x_2)$

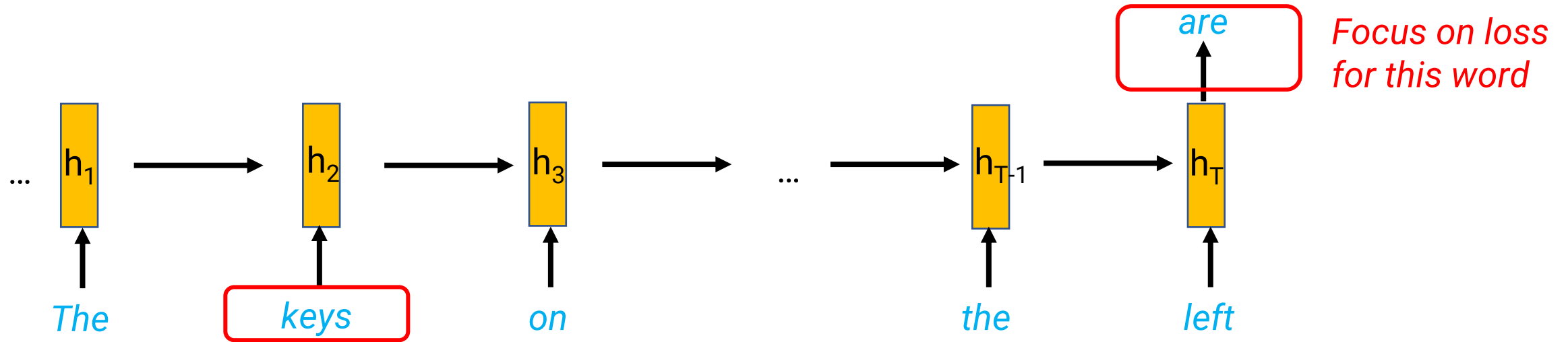
- What is each individual $\delta(h_t)/\delta(h_{t-1})$ term ?

- Elman network: $h_t = \tanh(W_h h_{t-1} + W_x x_t + b)$, $\frac{\delta h_t}{\delta h_{t-1}} = \underbrace{\tanh'(W_h h_{t-1} + W_x x_t + b)}_{\text{Ignore for now}} \cdot \underbrace{W_h}_{\text{The same parameter over and over!}}$

- After t timesteps, have a factor of $(W_h)^t$ (to the t -th power)!

- If $W_h \ll 1$, this quickly becomes 0 (“vanishes”)

The Vanishing Gradient Problem

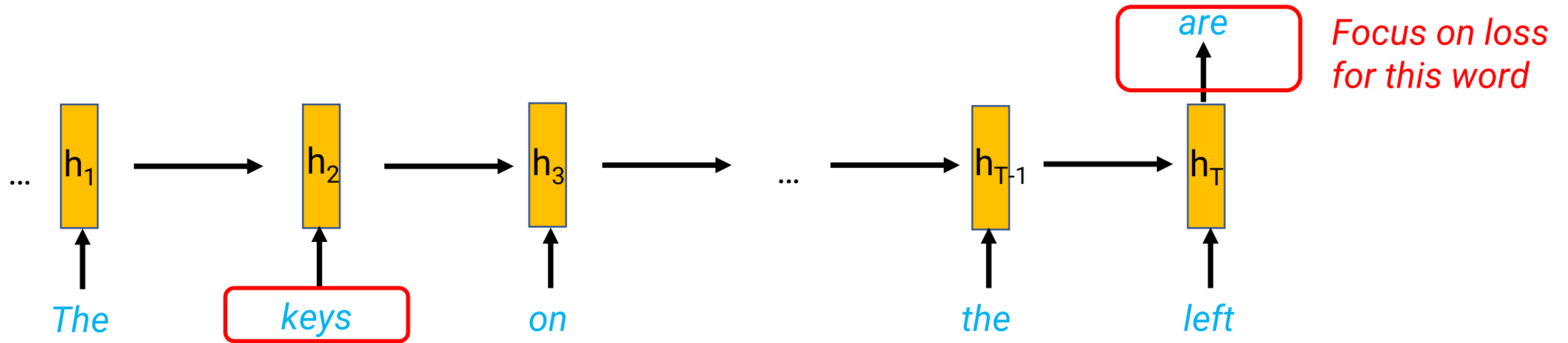


- Vanishing Gradients: Updates to one word/hidden state not influenced by loss on words many steps in the future
 - Illustrated only for 1-dimensional hidden states, but same thing happens when states are vectors/parameters are matrices
- Result: Hard for model to learn long-range dependencies!

Vanishing and Exploding

- Vanishing gradient occurs because
 - Gradient w.r.t. words t steps in the past has $(W_h)^t$
 - And when $W_h \ll 1$ (e.g., at initialization time)
- What if $W_h > 1$?
 - Gradients get bigger as you go backwards in time: Exploding gradients!
 - Vanishing gradients more usual, but explosion can happen too
- Quick fix: Gradient Clipping
 - If gradient is super large, “clip” it to some maximum amount
 - Rescale the total vector to some maximum norm
 - Clip each entry to be within some minimum/maximum value
- Outside of RNNs, vanishing/exploding gradients can happen whenever you have long computation graphs with lots of multiplications

Avoiding Vanishing Gradients



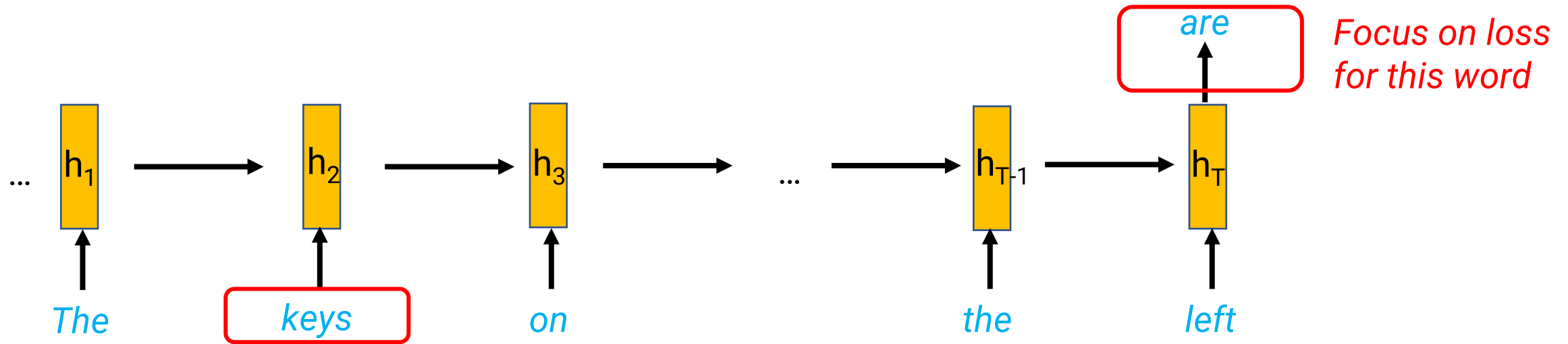
- Where did we go wrong?

$$h_t = \tanh(W_h h_{t-1} + W_x x_t + b), \quad \frac{\delta h_t}{\delta h_{t-1}} = \tanh'(W_h h_{t-1} + W_x x_t + b) \cdot W_h$$

Multiplicative
relationship between previous state and next state

Leads to repeated multiplication by W_h

Avoiding Vanishing Gradients

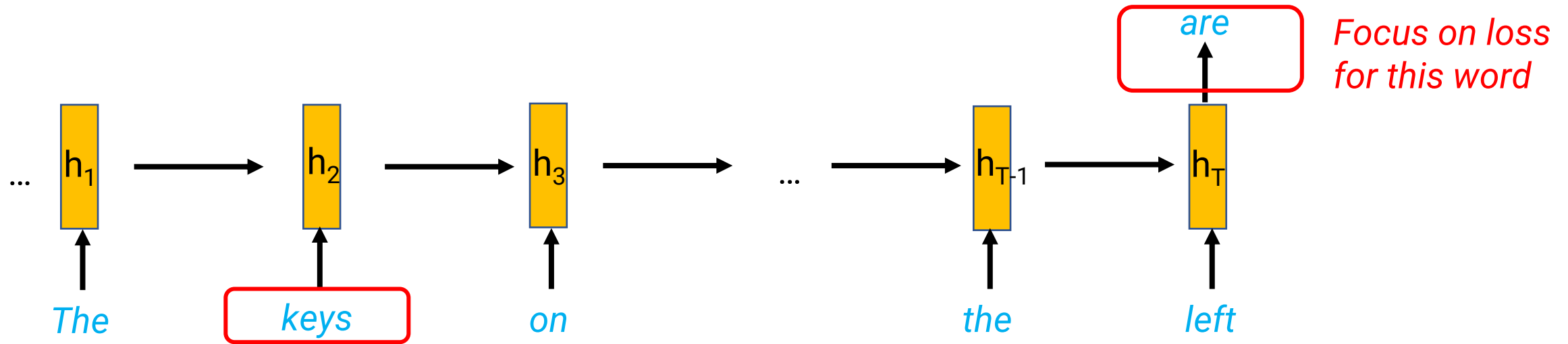


- Extreme idea: A purely additive relationship
 - Pro: No vanishing gradients
 - Pro: Old hidden state carried through to all future times
 - Con: May be good to “forget” irrelevant information about old states

$$h_t = h_{t-1} + \underbrace{g(h_{t-1}, x_t)}_{\text{Additive relationship}}$$

$$\frac{\delta h_t}{\delta h_{t-1}} = 1 + \underbrace{\frac{\delta}{\delta h_{t-1}} g(h_{t-1}, x_t)}_{\text{Gradients also add, not multiply}}$$

Avoiding Vanishing Gradients



- Middle-ground: **Gated** recurrence relationship

- Additive component makes gradients add, not multiply = less vanishing gradients
- Forget gate allows for selectively “forgetting” some neurons within hidden state
- When forget gate is all 1’s, becomes the purely additive model (no vanishing)

Elementwise multiplication

$$h_t = h_{t-1} \odot \underbrace{f(h_{t-1}, x_t)}_{\substack{\text{“forget gate”} \\ \text{in } [0, 1]}} + \underbrace{g(h_{t-1}, x_t)}_{\text{Additive relationship}}$$

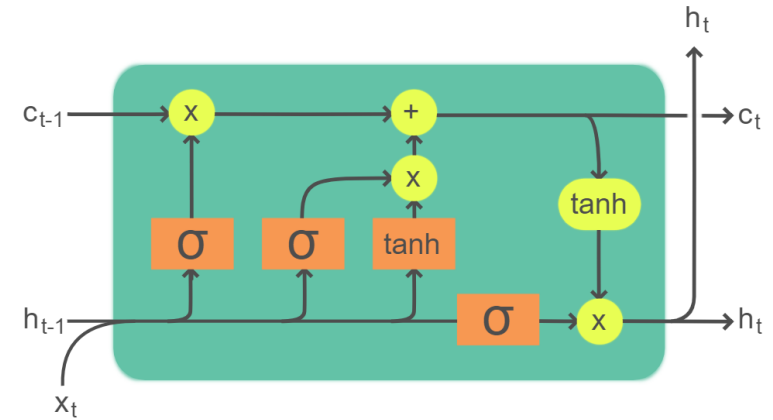
Gated Recurrent Units (GRUs)

- One type of gated RNN
 - Here z is the “forget gate” vector
 - Where $z_i = 0$:
 - Forget this neuron
 - Allow updating its value
 - Where $z_i = 1$:
 - Don’t forget this neuron
 - Do not allow updating its value
- Parameters: W , U , plus parameters of g
 - (g has a slightly complicated form not shown, has its own parameters)

$$h_t = h_{t-1} \odot z + \overbrace{g(x_t, h_{t-1}) \odot (1 - z)}^{\text{Additive relationship}}$$
$$z = \underbrace{\sigma(Wx_t + Uh_{t-1})}_{\text{Sigmoid ensures gate is between 0 and 1}}$$

Long Short-Term Memory (LSTM)

- Another, more complicated gated RNN
- Commonly used in practice
- What's the idea?
 - Split the hidden state into normal hidden state h_t and "cell" state c_t
 - Cell state uses gated recurrence
 - Hidden state is gated function of cell state



Legend:

Layer	Componentwise	Copy	Concatenate

What do LSTMs learn?

t t p : / / w w w . y n e t n e w s . c o m /] E n g l i s h - l a n g u a g e w e b s i t e o f I s r a e l ' s l a r
 t p : / / w w w b a c a h e t s . c o m / - x g l i s h l i n g u a g e s a i r s i t e o f t s l a e l i s s i n g
 d : x n e . w a e a . . a w a t o a . s & n t i a c a - s a r d e e l h o a n t b i s a n f a n r e i f ' a a t d
 m w - 2 p i i i s o e s s i s . / e r n . c] (d c e e n e p e s a a i k i i e e l e d h , i r t h r a o n s e , c o s e
 d r . < : a h b - n p t w t . x i g h / m a) T v d r y z i c o u e d l s u : t h a - o o t u , s t u i f l v e p e r y
 s t p , t c o a 2 d r u l w o c l e n s r] p . l l v a o d , , e y t c - n d m - o i b u v s] b b i m s u l t a t l y b n

g e s t n e w s p a p e r ' ' [[Y e d i o t h A h r o n o t h]] ' ' ' ' H e b r e w - l a n g u a g e p e r i o d
 e t a a w s p a p e r s o ' [[T e l t i (f e a n e m t i) ' ' * ' ' [e r r e w s l e n g u a g e : a r o s o d i
 i r s c o e e n a i T T h A o a i n n h S r m u w] e y s [' i n e i a ' s i w d d e ' h s o l r i f r :
 u s . . s e t l g o r s . a s a t C a r e e g ' a C l r i s z] i e ' : : , # : T A a a a a t B a s e e i l o ' i a n f v l
 - t u a e v r t i d , t B A m S u s y u t]] A s a o i g s]] , . : s M B o l o u s : T o u a - n : d w o a p n u
 a , d , i i u i t i c p .] (l S v H v t u s u i e D n o e g a n o . ,] : { C C u i b o h e C y b k s l s : r - e p c n t s

i c a l s : ' ' ' * ' ' [[G l o b e s]] ' ' [h t t p : / / w w w . g l o b e s . c o . i l /] b u s i n e s s d a
 c a l : ' ' ' * ' ' [T a a b a] ' ' ([t t p : / / w w w . b u o b a l . c o m u n / s A - y t i n e s s a e t
 s t l ' [h A e o v e l t s a h a d : x g e . w a o i r . r t o a . e l . i T & a i e g e o o y
 t t ' ' ' & [& & m C o e r o n e ' : : , i ' o d w . , : n i i i s a a u e . e n i / o m l c C . (e f t g i r i i u
 a ' n : , C : & : # * : a f D r u s u] l , . o m e l p < , d h a ; d e u o o t / i h n c s i f S , u r h o s t , t u n
 n k i <] : & 1 1 s T G u i t r s i , : b a c m r - x t p o b - g r e s i s l e r l n a f a D] l o s p t a d , i f r m

i l y * ' ' [[H a a r e t z | H a ' A r e t z]] ' ' [h t t p : / / w w w . h a a r e t z . c o . i l /] R e l a t i v
 l y * ' ' [[T e r r d n F e r a n t a h]] ' ' ([t t p : / / w w w . b o n m d s t . c o m u n / s - e s a t e o i
 r e ' ' h A i l n n t t e H a l s r c n o l ' s a h a d : x n e . w a a m r t d h e o h . o l . c & o p i n i v e
 k i . : * s C O S a n l t h i T i m ' l i] e : , i m c d w - 2 p h i i s e r d i t . i n a / c m f i . (a f l c a n a
 d s - ! [t B T C o m m g d]] W o n a a e , : . b a e r r . < t a i b - d u l c n n c / a r n e s i] l i c e y s t o
 n d s # & : G l D u v c c s a o S u c l t e l] z | , : o ' o m t] , : e o a 2 n i v f s r o o e i u n a l a) u v v r o

- Here: a character-level LSTM (not word-level)
- Blue/Green: Low/high values of 1 neuron
- Below: Top-5 predictions for next character
- This neuron seems to detect whether we're inside a URL

What do LSTMs learn?

* * * [[Jerusalem Report]] * * * [http://www.jrep.com/] Left-of-center En
 * * * [hToausal m aogurt]] * * * (http://www.bsinioom/ -iat af t enter (ng
 [' [Cassmene] Beaonds s a [ad : xne. waaaaoca. s &ato--nfhhlsu m-ouc
 ' s mFurnl s iaetal lsa ' : : , i ' cdw-2tpi i isoeg. er/. a) (oseswr - ci ddrs [mt
 : * : AqDenebiut n | Ci pre e | , . b1emr . 9 : ahb- npumughnmp) Teir et u : eoseodsald
 # T & T f Si wr p e]] aluvel ru , s : - mpr ts < moa2deys h i l r] c . Augl , 1p , larc : fae

g l i s h [[weekly newspaper]] * * * [[YNet News]] * * * [http://www.ynetnews.c
 l i s h c [Caakly] cawspaper]] * * * [hTaA at]] * * * (http://www.bacahets.co
 i aci - l hSoip] i sec] enp] s . ' ' [Co * wess] s a [ad : xne. waea . . awat oa
 een a , pCci et nedlox] g i c i | | s ' [sAmFeSahon] t ' : : , i momw-2 pi i i soessi s . / er
 syz . s f penn a | ruel | rra . ' # * : oDuFreiuep , : b1edr . < : ahb- nptwt . xi gh
 a dpe amAr bdeor pi tee] dt s - | T { [BaAvTpoSwa o , . . oacstp , tcoa2drulwoclens

om/] English-language website of israel's largest newspaper ' ' [[Yed
 m/] - xgl ish languages a i r s i t e o f t s l a e l i s s i n g e t a a w s p a p e r s o ' [[Tel
 . s & n t i a c a - s a r d e e l h o a n t b i s a n f a n r e i f ' a a t d i r s c o e e n a i T T h A o a i
 n . c] (d c e e n e p e s a a i k i i e e l e d h , i r t h r a o n s e , c o s e u s . . s e t l g o r s . a s a t C a r e
 / ma) T v d r y z i c o u e d l s u : t h a - o o t u , s t u i f l v e p e r y - t u a e v r t i d , t B A m S u s y
 r] p . l l v a o d , , e y t c - n d m - o i b u v s] b b i m s u l t a t l y b n a , d , i i u i t i c p .] (l S v H v t u

i o t h A h r o n o t h]] * * * * Hebrew-language periodicals : ' ' * * * [[Globes]] *
 t i (f e a n e m t i]] * * * [e r r e w s l e n g u a g e : a r o s o d i c a l : ' ' * * * [T a a b a]] * *
 n n h S r m u w] e y] s [' i n e i a ' s i w d d e ' h s o l r i f r : s t l ']] [h A e o v e l t]] s
 e g ' a C l r i s z] i e ' : : , # : T A a a a a t B a s e e i l o ' i a n f v l t t ' ' ' & [& m C o e r o n e ' : :
 u t]] A s a o i g s]] , . : s M B o l o u s : T o u a - n : d w o a p n u a ' n : , C : & # * : a f D r u s u] l ,
 s u i e D n o e g a n o . ,] : { C C u i b o h e C y b k s l s : r - e p c n t s n k i <] : & 1 1 s T G u i t r s i ,

- Here: a character-level LSTM (not word-level)
- Blue/Green: Low/high values of 1 neuron
- Below: Top-5 predictions for next character
- This neuron fires only inside a Markdown `[[link]]` (so it knows when to close the square brackets?)

Conclusion

- Deep Learning for Language must deal with possibly long inputs
- RNNs handle arbitrarily long inputs with fixed number of parameters
- Need to handle long-range dependencies, but hard to learn due to vanishing gradients
- Gated RNNs (GRUs, LSTMs) can reduce vanishing gradient problems