6. Problems with RNNs, LSTMs

LING-581-Natural Language Processing 1

Instructor: Hakyung Sung

September 30, 2025

*Acknowledgment: These course slides are based on materials from CS224N @ Stanford University; Dr. Kilho Shin @ Kyocera

Table of contents

Review

Review

- Language modeling

Review

- Language modeling
    - Definition

Review

- Language modeling
  - Definition
  - Applications

Review

- Language modeling
    - Definition
    - Applications
- approach 1?

Review

- Language modeling
  - Definition
  - Applications
- approach 1?
- approach 2?

Review

- Language modeling
  - Definition
  - Applications
- approach 1?
- approach 2?
- approach 3?

Review: n-gram language models

$$P(x_{t+1} \mid x_t, \ldots, x_1) \;\approx\; P(x_{t+1} \mid x_t, \ldots, x_{t-n+2})$$

- t: position of the current token in the sequence
- n: size of the n-gram (the model looks back $n-1$ tokens)

Only the last $(n-1)$ words matter.

Review: Conditional probability

- Definition:

$$P(A \mid B) = \frac{P(A, B)}{P(B)}.$$

- Apply to Markov assumption:

$$P(x_{t+1} \mid x_t, \dots, x_{t-n+2}) = \frac{P(x_{t+1}, x_t, \dots, x_{t-n+2})}{P(x_t, \dots, x_{t-n+2})}.$$

Review: Example

Every morning, my neighbor yelled at the _____

Review: Example

Every morning, my neighbor yelled at the _____

(**4-gram**) Conditioning only on the **last three words**:

Review: Example

Every morning, my neighbor yelled at the _____

(**4-gram**) Conditioning only on the **last three words**:

~~*Every morning, my neighbor*~~ *yelled at the* _____

Review: Example

Every morning, my neighbor yelled at the _____

(**4-gram**) Conditioning only on the **last three words**:

~~*Every morning, my neighbor*~~ *yelled at the* _____

$$\hat{P}(w \mid \text{yelled at the}) = \frac{\text{count(yelled at the } w\text{)}}{\text{count(yelled at the)}}.$$

Every morning, my neighbor yelled at the _____

(**4-gram**) Conditioning only on the **last three words**:

~~*Every morning, my neighbor*~~ *yelled at the* _____

$$\hat{P}(w \mid \text{yelled at the}) = \frac{\text{count(yelled at the } w\text{)}}{\text{count(yelled at the)}}.$$

Suppose in the corpus:

- *`yelled at the`* occurs 600 times,

Review: Example

Every morning, my neighbor yelled at the _____

(**4-gram**) Conditioning only on the **last three words**:

~~*Every morning, my neighbor*~~ *yelled at the* _____

$$\hat{P}(w \mid \text{yelled at the}) = \frac{\text{count(yelled at the } w\text{)}}{\text{count(yelled at the)}}.$$

Suppose in the corpus:

- ***yelled at the*** occurs 600 times,
- ***yelled at the dog*** occurs 250 times, so

Review: Example

Every morning, my neighbor yelled at the _____

(**4-gram**) Conditioning only on the **last three words**:

~~*Every morning, my neighbor*~~ *yelled at the* _____

$$\hat{P}(w \mid \text{yelled at the}) = \frac{\text{count(yelled at the } w\text{)}}{\text{count(yelled at the)}}.$$

Suppose in the corpus:

- `yelled at the` occurs 600 times,
- `yelled at the dog` occurs 250 times, so

Review: Example

Every morning, my neighbor yelled at the _____

(**4-gram**) Conditioning only on the **last three words**:

~~*Every morning, my neighbor*~~ *yelled at the* _____

$$\hat{P}(w \mid \text{yelled at the}) = \frac{\text{count(yelled at the } w)}{\text{count(yelled at the)}}.$$

Suppose in the corpus:

- `yelled at the` occurs 600 times,
- `yelled at the dog` occurs 250 times, so

$$P(\text{dog} \mid \text{yelled at the}) = 0.42,$$

- `yelled at the kids` occurs 180 times, so

Review: Example

Every morning, my neighbor yelled at the _____

(**4-gram**) Conditioning only on the **last three words**:

~~*Every morning, my neighbor*~~ *yelled at the* _____

$$\hat{P}(w \mid \text{yelled at the}) = \frac{\text{count(yelled at the } w\text{)}}{\text{count(yelled at the)}}.$$

Suppose in the corpus:

- `yelled at the` occurs 600 times,
- `yelled at the dog` occurs 250 times, so

$$P(\text{dog} \mid \text{yelled at the}) = 0.42,$$

- `yelled at the kids` occurs 180 times, so

Review: Example

Every morning, my neighbor yelled at the _____

(**4-gram**) Conditioning only on the **last three words**:

~~*Every morning, my neighbor*~~ *yelled at the* _____

$$\hat{P}(w \mid \text{yelled at the}) = \frac{\text{count(yelled at the } w\text{)}}{\text{count(yelled at the)}}.$$

Suppose in the corpus:

- `yelled at the` occurs 600 times,
- `yelled at the dog` occurs 250 times, so

$$P(\text{dog} \mid \text{yelled at the}) = 0.42,$$

- `yelled at the kids` occurs 180 times, so

$$P(\text{kids} \mid \text{yelled at the}) = 0.30.$$

Review: Window-based neural language model

output distribution

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

hidden layer

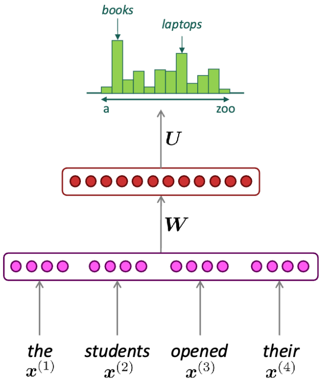
$\boldsymbol{h} = f(\boldsymbol{W}\boldsymbol{e} + \boldsymbol{b}_1)$

concatenated word embeddings

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

words / one-hot vectors

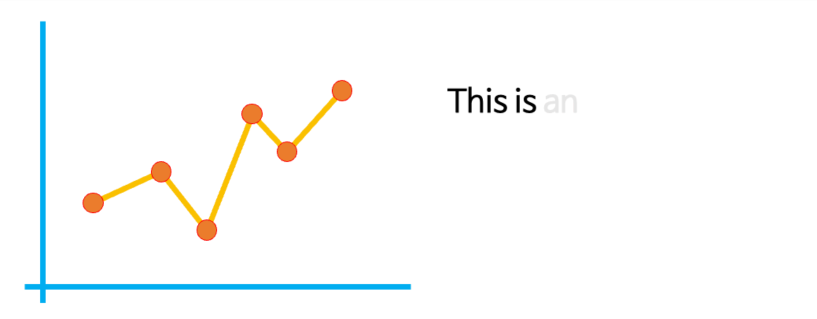
$\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, \boldsymbol{x}^{(3)}, \boldsymbol{x}^{(4)}$



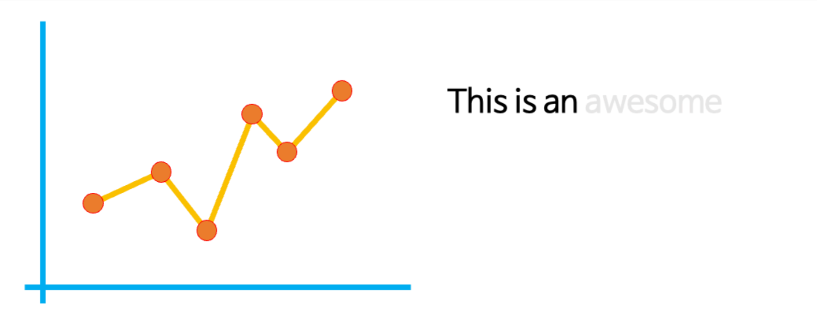
Review: RNNs

Good for processing continuous (time series) dataset like words in a sentence.

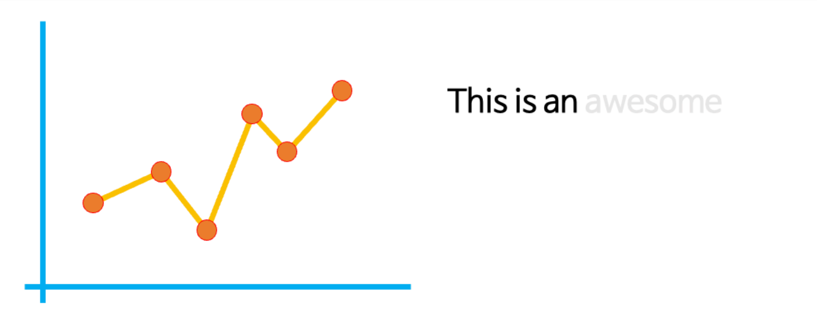
Good for processing continuous (time series) dataset like words in a sentence.



Good for processing continuous (time series) dataset like words in a sentence.

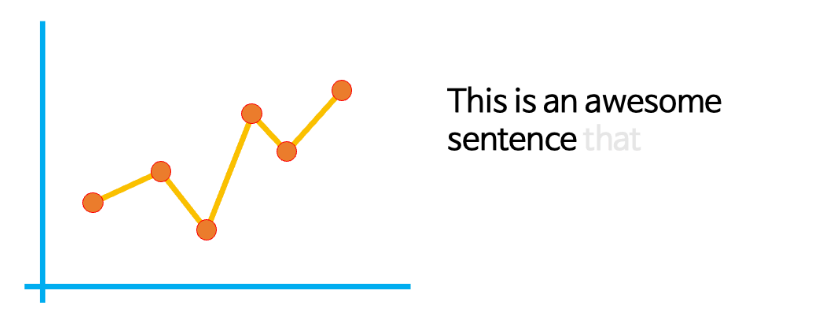


Good for processing continuous (time series) dataset like words in a sentence.

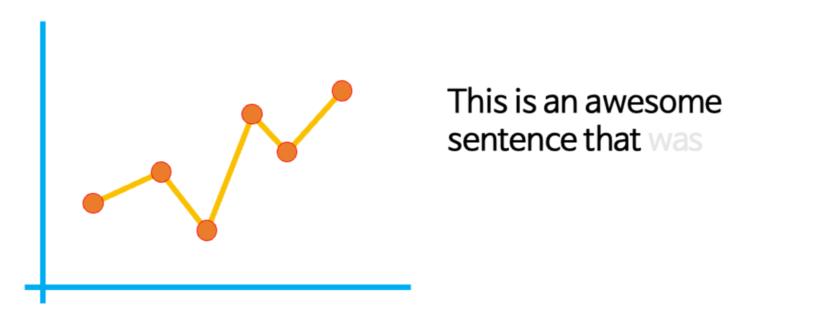


Good for processing continuous (time series) dataset like words in a sentence.

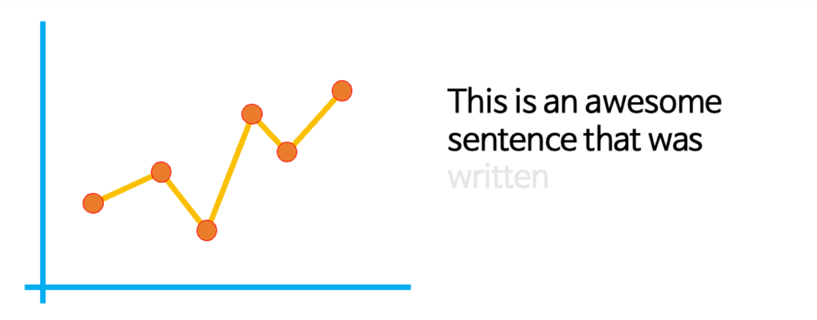
Good for processing continuous (time series) dataset like words in a sentence.



Good for processing continuous (time series) dataset like words in a sentence.

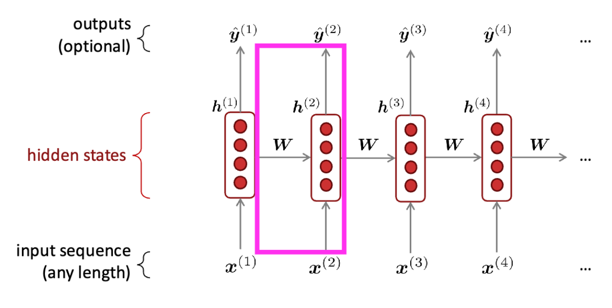


Good for processing continuous (time series) dataset like words in a sentence.



Review: RNNs

- Idea: Repeatedly apply the same weight matrix W at each time step
- Maintain a hidden state over time, feeding it back into the network to capture temporal dependencies



1. Start with a corpus, represented as a sequence of words $w_1, \dots, w_{T-1}, w_T$.

1. Start with a corpus, represented as a sequence of words $w_1, \dots, w_{T-1}, w_T$.
2. Feed this sequence into the RNN-based language model.

Review: RNNs

1. Start with a corpus, represented as a sequence of words $w_1, \ldots, w_{T-1}, w_T$.
2. Feed this sequence into the RNN-based language model.
3. At each time step t, the model outputs a probability distribution $\hat{\mathbf{y}}_t$ over the vocabulary.

Review: RNNs

1. Start with a corpus, represented as a sequence of words $w_1, \dots, w_{T-1}, w_T$.
2. Feed this sequence into the RNN-based language model.
3. At each time step t, the model outputs a probability distribution $\hat{\mathbf{y}}_t$ over the vocabulary.
    - Internally, the RNN updates its hidden state $\mathbf{h}_t$, then applies a linear layer followed by softmax:

$$\hat{\mathbf{y}}_t = \mathrm{softmax}(W_o\,\mathbf{h}_t + b_o).$$

Review: RNNs

1. Start with a corpus, represented as a sequence of words $w_1, \ldots, w_{T-1}, w_T$.
2. Feed this sequence into the RNN-based language model.
3. At each time step t, the model outputs a probability distribution $\hat{\mathbf{y}}_t$ over the vocabulary.
    - Internally, the RNN updates its hidden state $\mathbf{h}_t$, then applies a linear layer followed by softmax:

$$\hat{\mathbf{y}}_t = \mathrm{softmax}(W_o\,\mathbf{h}_t + b_o).$$

    - Each component of $\hat{\mathbf{y}}_t$ corresponds to

$$P(w_{t+1} = v_i \mid w_1, \ldots, w_t),$$

      i.e., the probability that the next word is v_i.

Review: RNNs

1. Start with a corpus, represented as a sequence of words $w_1, \ldots, w_{T-1}, w_T$.
2. Feed this sequence into the RNN-based language model.
3. At each time step t, the model outputs a probability distribution $\hat{\mathbf{y}}_t$ over the vocabulary.
    - Internally, the RNN updates its hidden state $\mathbf{h}_t$, then applies a linear layer followed by softmax:

    $$\hat{\mathbf{y}}_t = \mathrm{softmax}(W_o\,\mathbf{h}_t + b_o).$$

    - Each component of $\hat{\mathbf{y}}_t$ corresponds to

    $$P(w_{t+1} = v_i \mid w_1, \ldots, w_t),$$

      i.e., the probability that the next word is v_i.
    - Put simply, at every step t, the model predicts the likelihood of each possible next word **given all preceding words**.

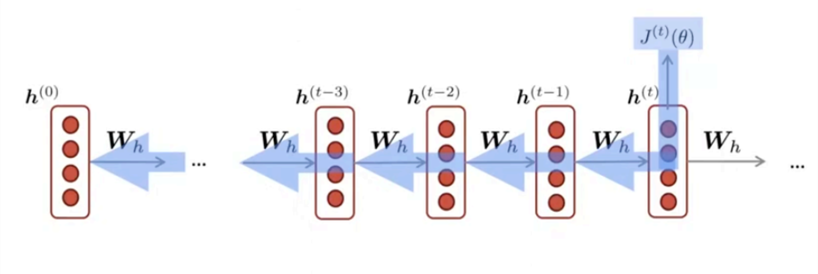
- **Loss** at step t:

$$\mathcal{J}^{(t)} = -\sum_{i=1}^{|V|} y_i^{(t)} \log \hat{y}_i^{(t)} = -\log \hat{y}_{w_{t+1}}^{(t)},$$

where:

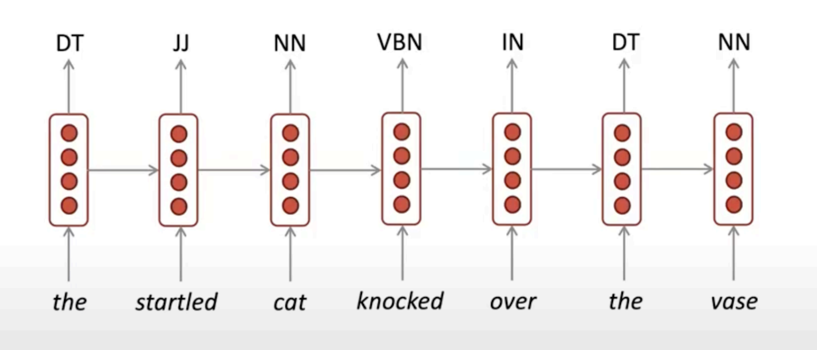
  - $y^{(t)}$: one-hot vector for the true next word w_{t+1}.
  - $\hat{y}^{(t)}$: predicted probability distribution over the vocabulary from the softmax layer.
  - This is the cross-entropy **loss** between the predicted distribution and the true label.
  - higher loss? lower loss?
  - *https://www.desmos.com/calculator*

Review: RNNs+Backpropagation

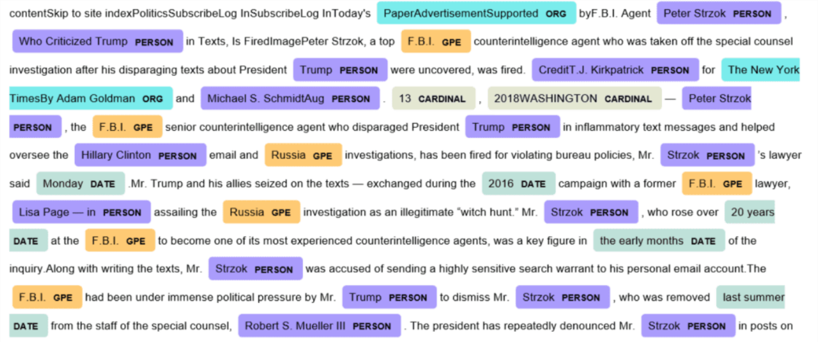


Review: NLP applications

POS tagging

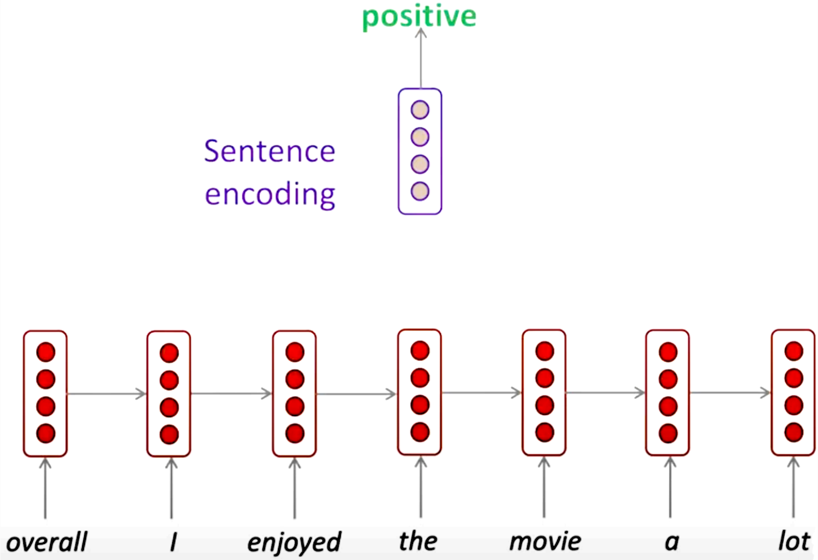


NER tagging

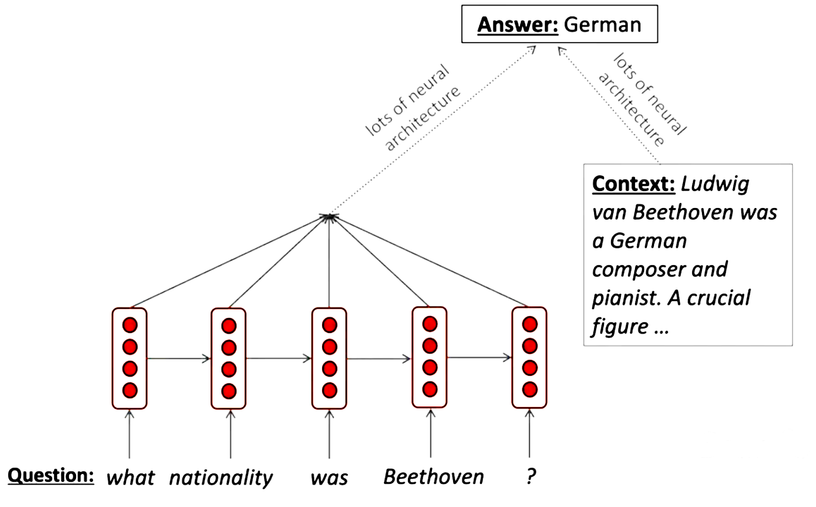


* A `named entity` is a specific word or phrase that refers to a particular person, place, organization, money, time or other real-world values. *`https://www.wisecube.ai/blog/named-entity-recognition-ner-with-python/`*

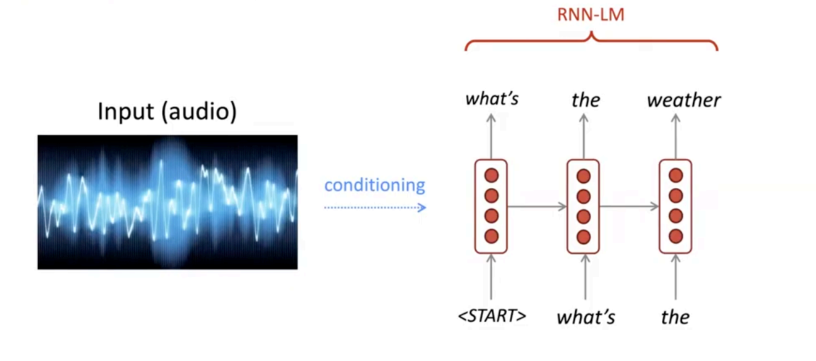
Sentiment classification



Question answering



Speech recognition



Lesson plan

Lesson plan

- Problems with RNNs

Lesson plan

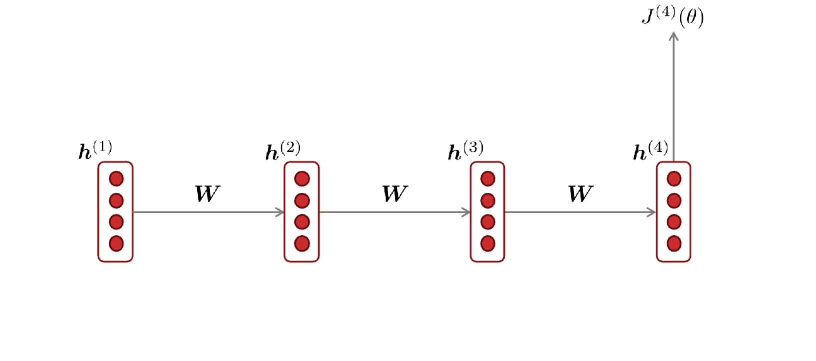
- Problems with RNNs
- LSTMs

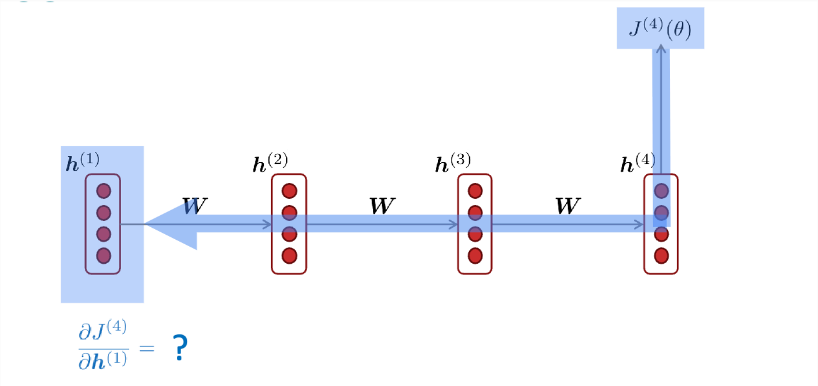
Lesson plan

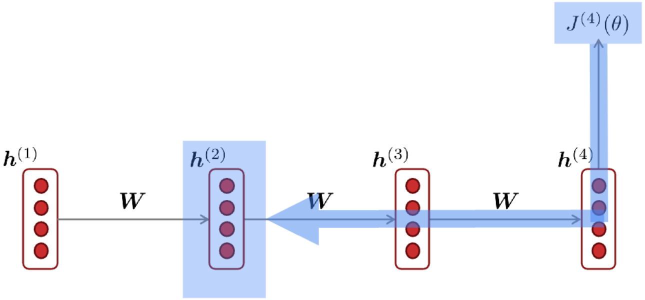
- Problems with RNNs
- LSTMs
- Bidirectional/multi-layer models

Problems with RNNs

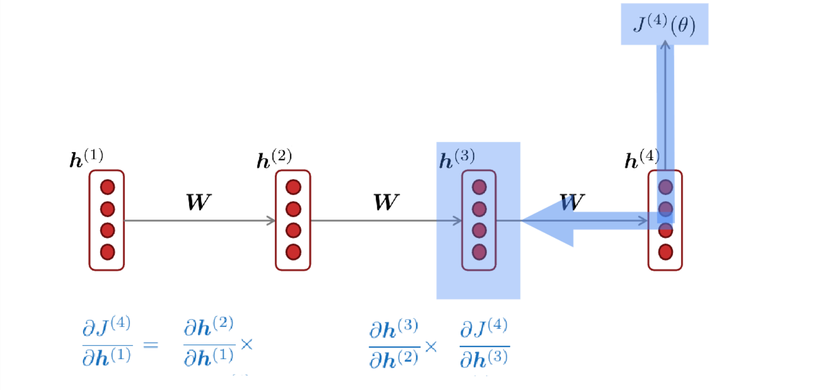
Problem with RNN 1: Vanishing gradient

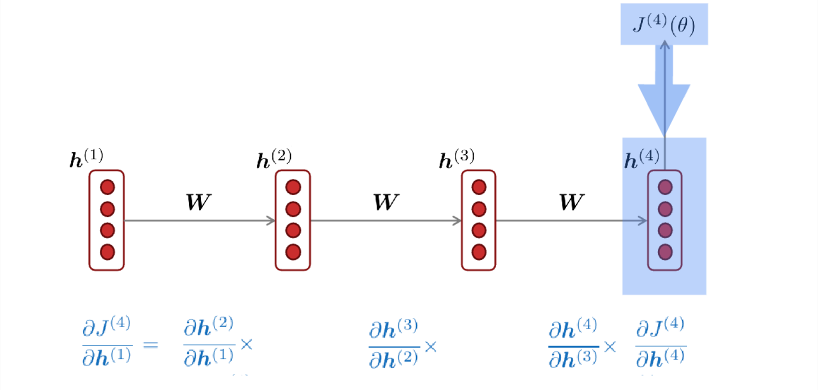



$J^{(4)}(\theta)$
$h^{(1)}$
$h^{(2)}$
$h^{(3)}$
$h^{(4)}$
W
W
W
$\frac{\partial J^{(4)}}{\partial h^{(1)}} =$?



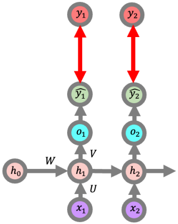
$$\frac{\partial J^{(4)}}{\partial \boldsymbol{h}^{(1)}} = \frac{\partial \boldsymbol{h}^{(2)}}{\partial \boldsymbol{h}^{(1)}} \times \frac{\partial J^{(4)}}{\partial \boldsymbol{h}^{(2)}}$$


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



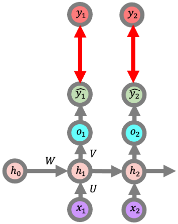
tldr: **If each step's gradient is too small**, multiplying across many steps makes it shrink exponentially.
The overall gradient $\rightarrow 0$, so the model cannot learn long-range dependencies.

More explanation: long-term dependency



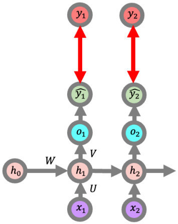
More explanation: long-term dependency and chain rule

$$\frac{\partial L_2}{\partial W}=\frac{\partial L_2}{\partial \widehat{y_2}}\frac{\partial \widehat{y_2}}{\partial o_2}\frac{\partial o_2}{\partial h_2}\frac{\partial h_2}{\partial W}+\frac{\partial L_2}{\partial \widehat{y_2}}\frac{\partial \widehat{y_2}}{\partial o_2}\frac{\partial o_2}{\partial h_2}\frac{\partial h_2}{\partial h_1}\frac{\partial h_1}{\partial W}$$



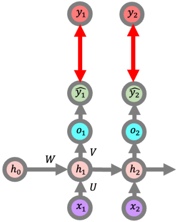
As we can see here, as time increases (as embedding nodes increase)

$$\frac{\partial L_2}{\partial W}=\frac{\partial L_2}{\partial \widehat{y_2}}\frac{\partial \widehat{y_2}}{\partial o_2}\frac{\partial o_2}{\partial h_2}\frac{\partial h_2}{\partial W}+\frac{\partial L_2}{\partial \widehat{y_2}}\frac{\partial \widehat{y_2}}{\partial o_2}\frac{\partial o_2}{\partial h_2}\boxed{\frac{\partial h_2}{\partial h_1}}\frac{\partial h_1}{\partial W}$$



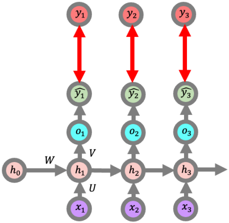
As we can see here, as time increases (as embedding nodes increase)

$$\frac{\partial L_2}{\partial W} = \frac{\partial L_2}{\partial \widehat{y_2}} \frac{\partial \widehat{y_2}}{\partial o_2} \frac{\partial o_2}{\partial h_2} \frac{\partial h_2}{\partial W} + \frac{\partial L_2}{\partial \widehat{y_2}} \frac{\partial \widehat{y_2}}{\partial o_2} \frac{\partial o_2}{\partial h_2} \boxed{\frac{\partial h_2}{\partial h_1}} \frac{\partial h_1}{\partial W}$$



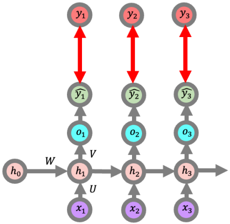
As we can see here, as time increases (as embedding nodes increase), the part that needs to be calculated by the chain rule

$$\frac{\partial L_3}{\partial W}=\frac{\partial L_3}{\partial \widehat{y_3}}\frac{\partial \widehat{y_3}}{\partial o_3}\frac{\partial o_3}{\partial h_3}\frac{\partial h_3}{\partial W}+\frac{\partial L_3}{\partial \widehat{y_3}}\frac{\partial \widehat{y_3}}{\partial o_3}\frac{\partial o_3}{\partial h_3}\frac{\partial h_3}{\partial h_2}\frac{\partial h_2}{\partial W}+\frac{\partial L_3}{\partial \widehat{y_3}}\frac{\partial \widehat{y_3}}{\partial o_3}\frac{\partial o_3}{\partial h_3}\frac{\partial h_3}{\partial h_2}\frac{\partial h_2}{\partial h_1}\frac{\partial h_1}{\partial W}$$



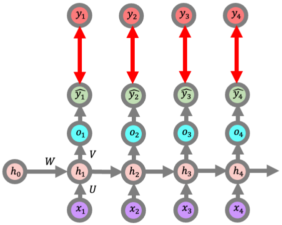
As we can see here, as time increases (as embedding nodes increase), the part that needs to be calculated by the chain rule

$$\frac{\partial L_3}{\partial W} = \frac{\partial L_3}{\partial \widehat{y_3}} \frac{\partial \widehat{y_3}}{\partial o_3} \frac{\partial o_3}{\partial h_3} \frac{\partial h_3}{\partial W} + \frac{\partial L_3}{\partial \widehat{y_3}} \frac{\partial \widehat{y_3}}{\partial o_3} \frac{\partial o_3}{\partial h_3} \frac{\partial h_3}{\partial h_2} \frac{\partial h_2}{\partial W} + \frac{\partial L_3}{\partial \widehat{y_3}} \frac{\partial \widehat{y_3}}{\partial o_3} \frac{\partial o_3}{\partial h_3} \frac{\partial h_3}{\partial h_2} \frac{\partial h_2}{\partial h_1} \frac{\partial h_1}{\partial W}$$



As we can see here, as time increases (as embedding nodes increase), the part that needs to be calculated by the chain rule keeps

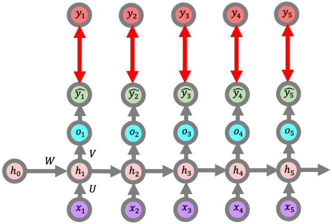
$$\frac{\partial L_4}{\partial W}=\frac{\partial L_4}{\partial \widehat{y_4}}\frac{\partial \widehat{y_4}}{\partial o_4}\frac{\partial o_4}{\partial h_4}\frac{\partial h_4}{\partial W}+\frac{\partial L_4}{\partial \widehat{y_4}}\frac{\partial \widehat{y_4}}{\partial o_4}\frac{\partial o_4}{\partial h_4}\frac{\partial h_4}{\partial h_3}\frac{\partial h_3}{\partial W}+\frac{\partial L_4}{\partial \widehat{y_4}}\frac{\partial \widehat{y_4}}{\partial o_4}\frac{\partial o_4}{\partial h_4}\frac{\partial h_4}{\partial h_3}\frac{\partial h_3}{\partial h_2}\frac{\partial h_2}{\partial W}$$
$$+\frac{\partial L_4}{\partial \widehat{y_4}}\frac{\partial \widehat{y_4}}{\partial o_4}\frac{\partial o_4}{\partial h_4}\frac{\partial h_4}{\partial h_3}\frac{\partial h_3}{\partial h_2}\frac{\partial h_2}{\partial h_1}\frac{\partial h_1}{\partial W}$$



As we can see here, as time increases (as embedding nodes increase), the part that needs to be calculated by the chain rule keeps increasing

$$\frac{\partial L_5}{\partial W}=\frac{\partial L_5}{\partial \widehat{y_5}}\frac{\partial \widehat{y_5}}{\partial o_5}\frac{\partial o_5}{\partial h_5}\frac{\partial h_5}{\partial W}+\frac{\partial L_5}{\partial \widehat{y_5}}\frac{\partial \widehat{y_5}}{\partial o_5}\frac{\partial o_5}{\partial h_5}\frac{\partial h_5}{\partial h_4}\frac{\partial h_4}{\partial W}+\frac{\partial L_5}{\partial \widehat{y_5}}\frac{\partial \widehat{y_5}}{\partial o_5}\frac{\partial o_5}{\partial h_5}\frac{\partial h_5}{\partial h_4}\frac{\partial h_4}{\partial h_3}\frac{\partial h_3}{\partial W}$$

$$+\frac{\partial L_5}{\partial \widehat{y_5}}\frac{\partial \widehat{y_5}}{\partial o_5}\frac{\partial o_5}{\partial h_5}\frac{\partial h_5}{\partial h_4}\frac{\partial h_4}{\partial h_3}\frac{\partial h_3}{\partial h_2}\frac{\partial h_2}{\partial W}+\frac{\partial L_5}{\partial \widehat{y_5}}\frac{\partial \widehat{y_5}}{\partial o_5}\frac{\partial o_5}{\partial h_5}\frac{\partial h_5}{\partial h_4}\frac{\partial h_4}{\partial h_3}\frac{\partial h_3}{\partial h_2}\frac{\partial h_2}{\partial h_1}\frac{\partial h_1}{\partial W}$$

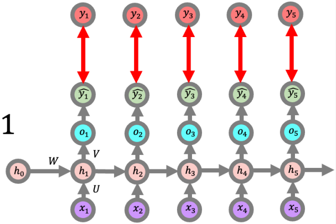


If these parts are smaller than 1

$$\frac{\partial L_5}{\partial W}=\frac{\partial L_5}{\partial \widehat{y_5}}\frac{\partial \widehat{y_5}}{\partial o_5}\frac{\partial o_5}{\partial h_5}\frac{\partial h_5}{\partial W}+\frac{\partial L_5}{\partial \widehat{y_5}}\frac{\partial \widehat{y_5}}{\partial o_5}\frac{\partial o_5}{\partial h_5}\frac{\partial h_5}{\partial h_4}\frac{\partial h_4}{\partial W}+\frac{\partial L_5}{\partial \widehat{y_5}}\frac{\partial \widehat{y_5}}{\partial o_5}\frac{\partial o_5}{\partial h_5}\frac{\partial h_5}{\partial h_4}\frac{\partial h_4}{\partial h_3}\frac{\partial h_3}{\partial W}$$

$$+\frac{\partial L_5}{\partial \widehat{y_5}}\frac{\partial \widehat{y_5}}{\partial o_5}\frac{\partial o_5}{\partial h_5}\frac{\partial h_5}{\partial h_4}\frac{\partial h_4}{\partial h_3}\frac{\partial h_3}{\partial h_2}\frac{\partial h_2}{\partial W}+\frac{\partial L_5}{\partial \widehat{y_5}}\frac{\partial \widehat{y_5}}{\partial o_5}\frac{\partial o_5}{\partial h_5}\frac{\partial h_5}{\partial h_4}\frac{\partial h_4}{\partial h_3}\frac{\partial h_3}{\partial h_2}\frac{\partial h_2}{\partial h_1}\frac{\partial h_1}{\partial W}$$

< 1

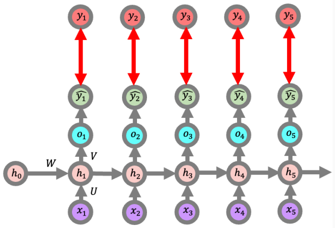


Then, as we keep multiplying through the chain rule, the gradient value for distant parts becomes smaller

$$\frac{\partial L_5}{\partial W}=\frac{\partial L_5}{\partial \widehat{y_5}}\frac{\partial \widehat{y_5}}{\partial o_5}\frac{\partial o_5}{\partial h_5}\frac{\partial h_5}{\partial W}+\frac{\partial L_5}{\partial \widehat{y_5}}\frac{\partial \widehat{y_5}}{\partial o_5}\frac{\partial o_5}{\partial h_5}\frac{\partial h_5}{\partial h_4}\frac{\partial h_4}{\partial W}+\frac{\partial L_5}{\partial \widehat{y_5}}\frac{\partial \widehat{y_5}}{\partial o_5}\frac{\partial o_5}{\partial h_5}\frac{\partial h_5}{\partial h_4}\frac{\partial h_4}{\partial h_3}\frac{\partial h_3}{\partial W}$$

$$+\frac{\partial L_5}{\partial \widehat{y_5}}\frac{\partial \widehat{y_5}}{\partial o_5}\frac{\partial o_5}{\partial h_5}\frac{\partial h_5}{\partial h_4}\frac{\partial h_4}{\partial h_3}\frac{\partial h_3}{\partial h_2}\frac{\partial h_2}{\partial W}+\boxed{\frac{\partial L_5}{\partial \widehat{y_5}}\frac{\partial \widehat{y_5}}{\partial o_5}\frac{\partial o_5}{\partial h_5}\frac{\partial h_5}{\partial h_4}\frac{\partial h_4}{\partial h_3}\frac{\partial h_3}{\partial h_2}\frac{\partial h_2}{\partial h_1}\frac{\partial h_1}{\partial W}} \downarrow$$

e.g., 0.1x0.3x0.2x0.1 = 0.0006

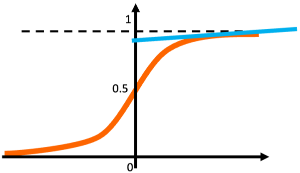


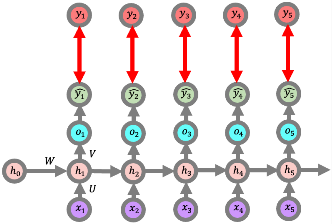
A smaller gradient means that its effect on learning is negligible,

$$\frac{\partial L_5}{\partial W}=\frac{\partial L_5}{\partial \widehat{y_5}}\frac{\partial \widehat{y_5}}{\partial o_5}\frac{\partial o_5}{\partial h_5}\frac{\partial h_5}{\partial W}+\frac{\partial L_5}{\partial \widehat{y_5}}\frac{\partial \widehat{y_5}}{\partial o_5}\frac{\partial o_5}{\partial h_5}\frac{\partial h_5}{\partial h_4}\frac{\partial h_4}{\partial W}+\frac{\partial L_5}{\partial \widehat{y_5}}\frac{\partial \widehat{y_5}}{\partial o_5}\frac{\partial o_5}{\partial h_5}\frac{\partial h_5}{\partial h_4}\frac{\partial h_4}{\partial h_3}\frac{\partial h_3}{\partial W}$$

$$+\frac{\partial L_5}{\partial \widehat{y_5}}\frac{\partial \widehat{y_5}}{\partial o_5}\frac{\partial o_5}{\partial h_5}\frac{\partial h_5}{\partial h_4}\frac{\partial h_4}{\partial h_3}\frac{\partial h_3}{\partial h_2}\frac{\partial h_2}{\partial W}+\boxed{\frac{\partial L_5}{\partial \widehat{y_5}}\frac{\partial \widehat{y_5}}{\partial o_5}\frac{\partial o_5}{\partial h_5}\frac{\partial h_5}{\partial h_4}\frac{\partial h_4}{\partial h_3}\frac{\partial h_3}{\partial h_2}\frac{\partial h_2}{\partial h_1}\frac{\partial h_1}{\partial W}}\downarrow$$

e.g., 0.1x0.3x0.2x0.1 = 0.0006



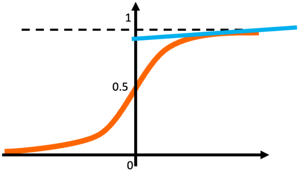


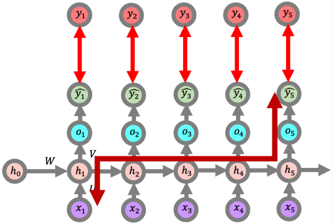
As a result, the farther back in time the input is, the smaller its effect on learning becomes

$$\frac{\partial L_5}{\partial W}=\frac{\partial L_5}{\partial \widehat{y_5}}\frac{\partial \widehat{y_5}}{\partial o_5}\frac{\partial o_5}{\partial h_5}\frac{\partial h_5}{\partial W}+\frac{\partial L_5}{\partial \widehat{y_5}}\frac{\partial \widehat{y_5}}{\partial o_5}\frac{\partial o_5}{\partial h_5}\frac{\partial h_5}{\partial h_4}\frac{\partial h_4}{\partial W}+\frac{\partial L_5}{\partial \widehat{y_5}}\frac{\partial \widehat{y_5}}{\partial o_5}\frac{\partial o_5}{\partial h_5}\frac{\partial h_5}{\partial h_4}\frac{\partial h_4}{\partial h_3}\frac{\partial h_3}{\partial W}$$

$$+\frac{\partial L_5}{\partial \widehat{y_5}}\frac{\partial \widehat{y_5}}{\partial o_5}\frac{\partial o_5}{\partial h_5}\frac{\partial h_5}{\partial h_4}\frac{\partial h_4}{\partial h_3}\frac{\partial h_3}{\partial h_2}\frac{\partial h_2}{\partial W}+\boxed{\frac{\partial L_5}{\partial \widehat{y_5}}\frac{\partial \widehat{y_5}}{\partial o_5}\frac{\partial o_5}{\partial h_5}\frac{\partial h_5}{\partial h_4}\frac{\partial h_4}{\partial h_3}\frac{\partial h_3}{\partial h_2}\frac{\partial h_2}{\partial h_1}\frac{\partial h_1}{\partial W}} \downarrow$$

e.g., 0.1x0.3x0.2x0.1 = 0.0006





Example:

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* ________

Example:

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

Example:

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 LM needs to model the dependency between "tickets" on the 7th step and the target word "tickets" at the end.
- But if the gradient is small, the model can't learn this dependency

Example:

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 LM needs to model the dependency between "tickets" on the 7th step and the target word "tickets" at the end.
- But if the gradient is small, the model can't learn this dependency
  - So, the model is unable to predict similar long-distance dependencies at test time

Problem with RNN 2: Exploding gradient

- When gradients become very large:

Problem with RNN 2: Exploding gradient

- When gradients become very large:
    - A single update step can overshoot the minimum

Problem with RNN 2: Exploding gradient

- When gradients become very large:
  - A single update step can overshoot the minimum
  - and destabilize or even blow up the model..!

Problems: Summary

- RNNs are equivalent to a deep network of depth T when unrolled over time (T = sequence length/time steps)

Problems: Summary

- RNNs are equivalent to a deep network of depth T when unrolled over time (T = sequence length/time steps)
- **Parameter sharing:** the same weight matrices are multiplied at each time step.

Problems: Summary

- RNNs are equivalent to a deep network of depth T when unrolled over time (T = sequence length/time steps)
- **Parameter sharing:** the same weight matrices are multiplied at each time step.
- If each multiplication factor:

Problems: Summary

- RNNs are equivalent to a deep network of depth T when unrolled over time (T = sequence length/time steps)
- **Parameter sharing:** the same weight matrices are multiplied at each time step.
- If each multiplication factor:
  - is < 1, gradients shrink exponentially (vanishing).

Problems: Summary

- RNNs are equivalent to a deep network of depth T when unrolled over time (T = sequence length/time steps)
- **Parameter sharing:** the same weight matrices are multiplied at each time step.
- If each multiplication factor:
  - is < 1, gradients shrink exponentially (vanishing).
  - is > 1, gradients grow exponentially (exploding).

Problems: Summary

- RNNs are equivalent to a deep network of depth T when unrolled over time (T = sequence length/time steps)
- **Parameter sharing:** the same weight matrices are multiplied at each time step.
- If each multiplication factor:
  - is < 1, gradients shrink exponentially (vanishing).
  - is > 1, gradients grow exponentially (exploding).
- Standard feedforward nets have limited depth, so this extreme behavior is less pronounced.

Vanishing problem: Solution

Solutions explored:

- Separate memory cell (e.g., LSTM) with gating mechanisms to add/erase information.

Vanishing problem: Solution

Solutions explored:

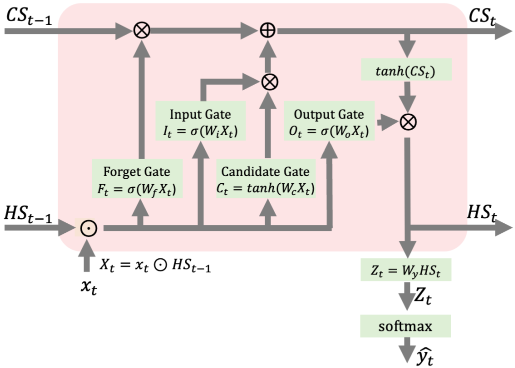
- Separate memory cell (e.g., LSTM) with gating mechanisms to add/erase information.
- Direct pass-through connections (attention, residual links) for better gradient flow.

LSTMs

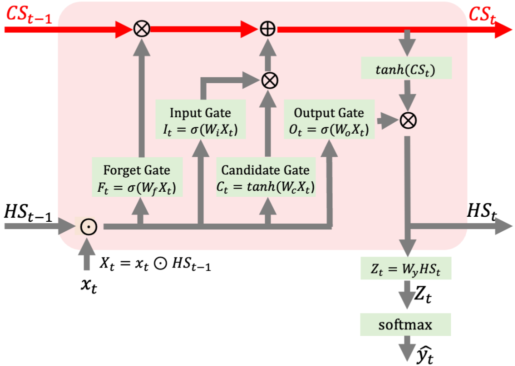
Overview

Separate memory cell with gating mechanisms to add/erase information.

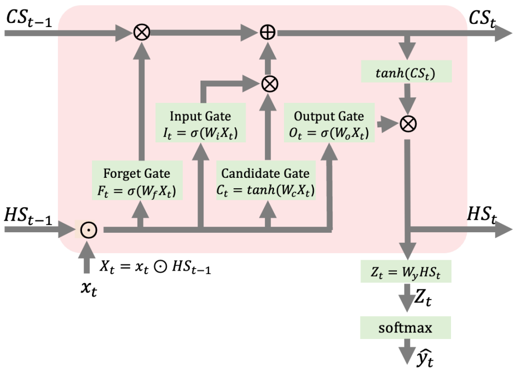
Then, let's understand LSTM's separate memory cell.



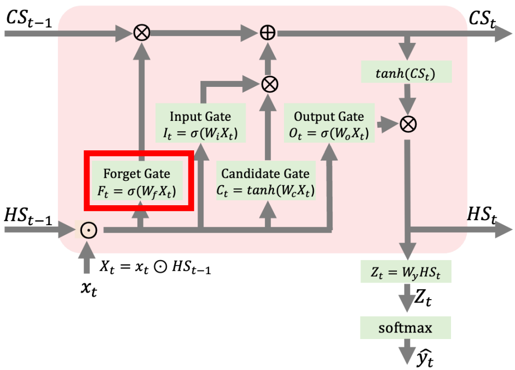
The secret lies in the information called the **cell state (CS)**.



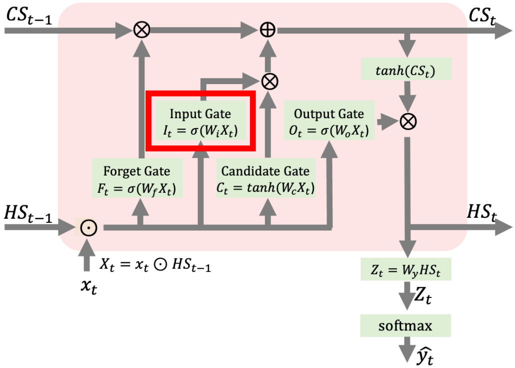
And LSTM has four gates that differ from an RNN.



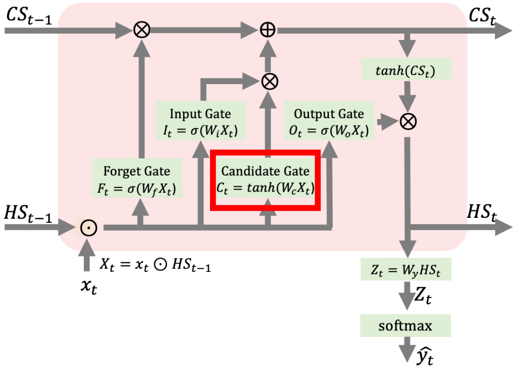
And LSTM has four gates that differ from an RNN.



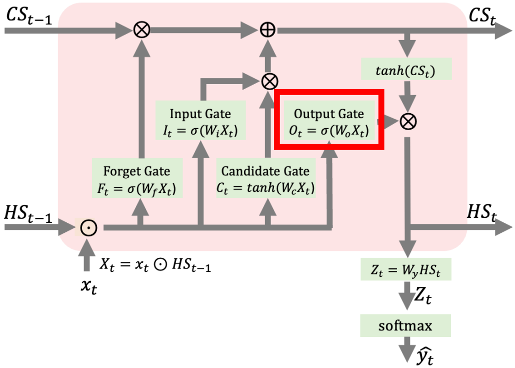
And LSTM has four gates that differ from an RNN.



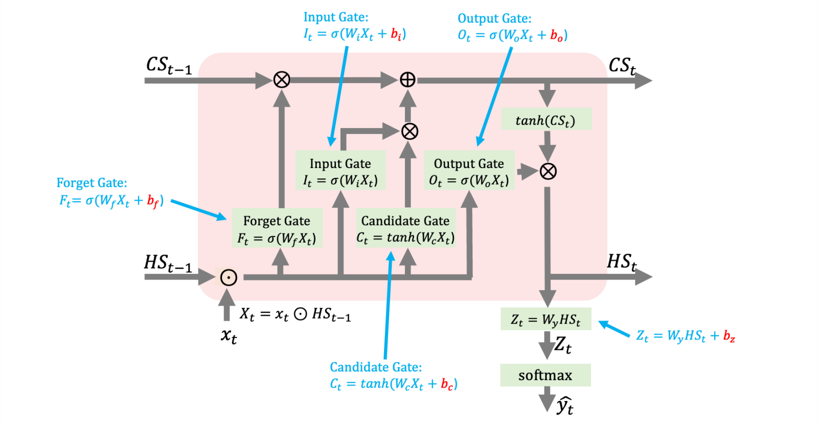
And LSTM has four gates that differ from an RNN.



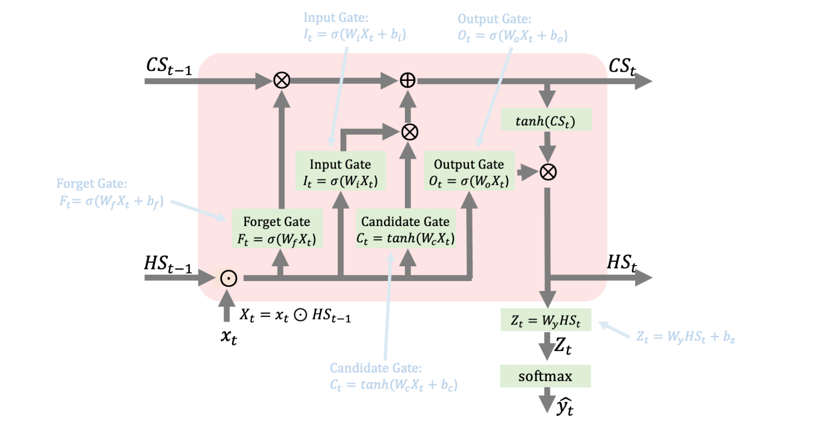
And LSTM has four gates that differ from an RNN.



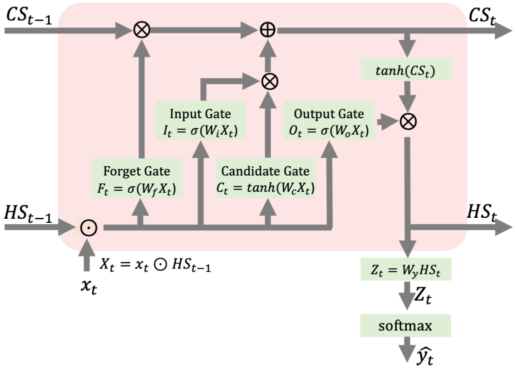
Originally, each gate and layer should include a bias term.



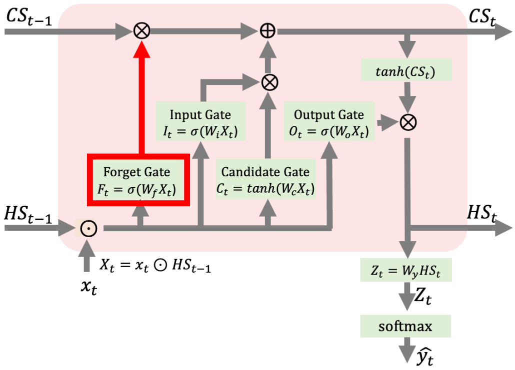
But for convenience, we'll omit them for now.



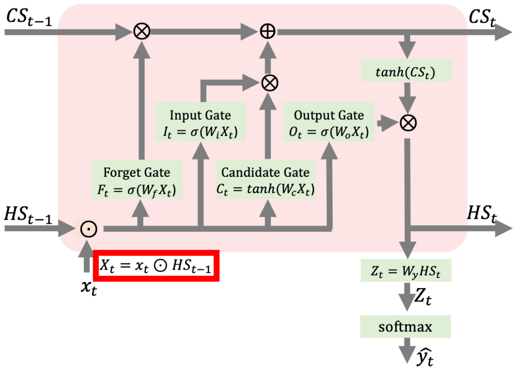
Now, let's see how each gate processes information.



First, as the name suggests, the Forget Gate decides which information to erase (forget).

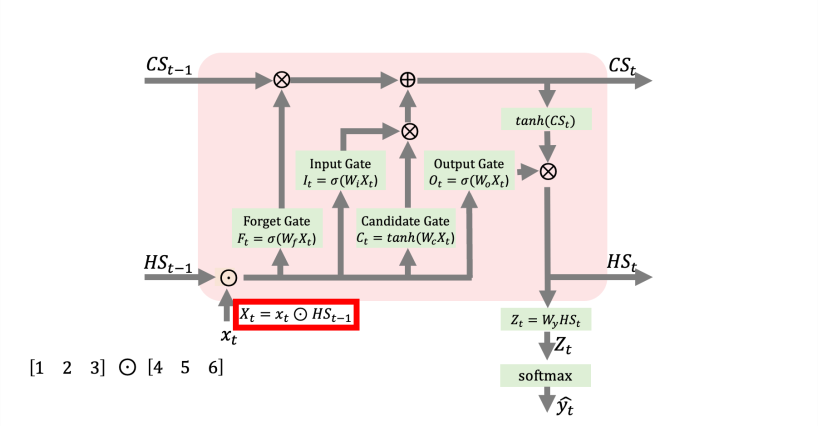


The input to the the Forget Gate is the concatenation of the previous hidden state (HS_{t-1}) and the current input (x_t).

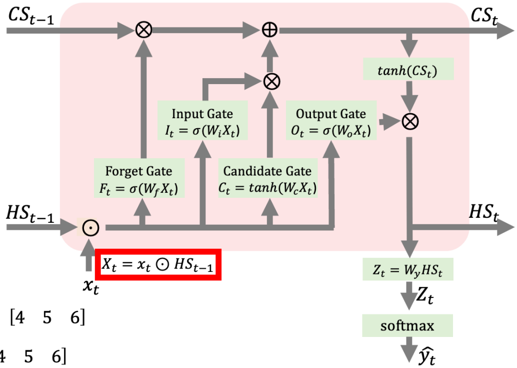


Concatenate?

Concatenate? Joining two vectors/matrices end-to-end.



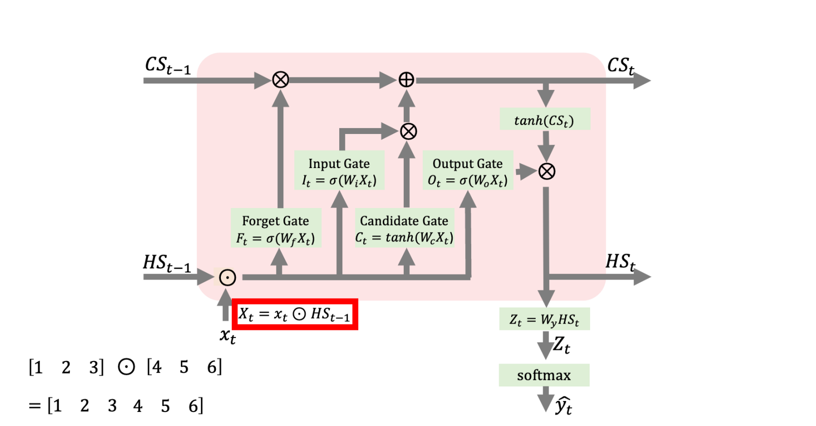
Concatenate? Joining two vectors/matrices end-to-end.



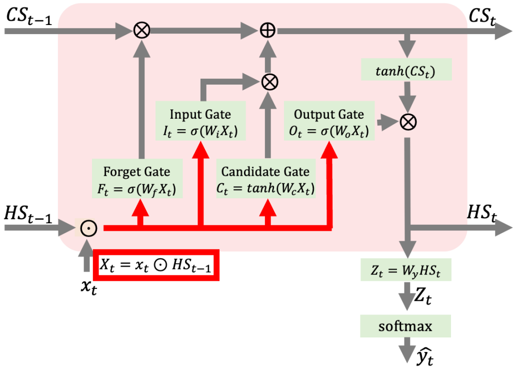
$$[1 \quad 2 \quad 3] \odot [4 \quad 5 \quad 6]$$

$$= [1 \quad 2 \quad 3 \quad 4 \quad 5 \quad 6]$$

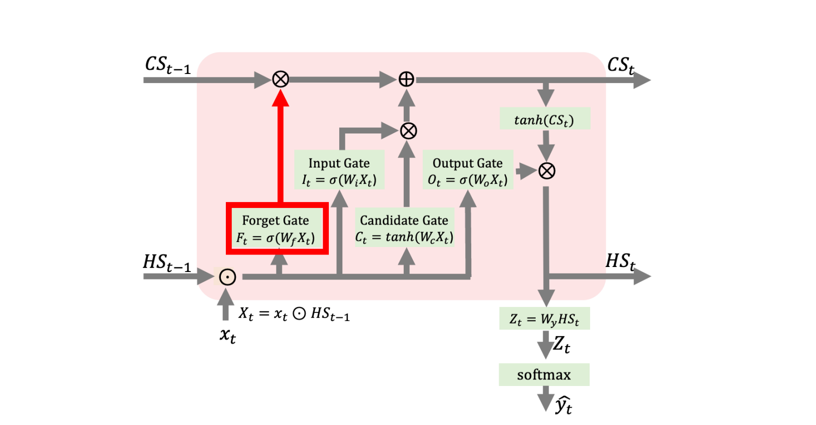
By doing this, the concatenated x_t **becomes a kind of short-term memory** that **bundles the previous hidden state and the current input together**.



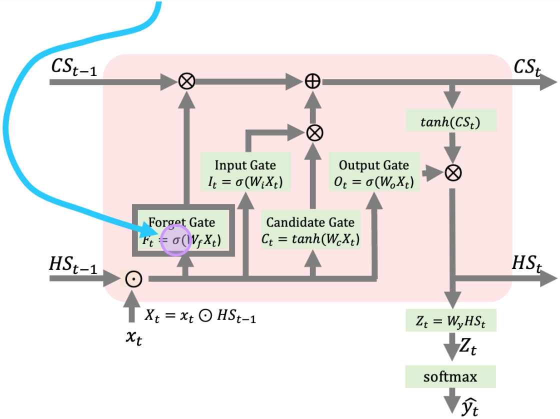
Remember: this x_t serves as the **input** to all gates in the LSTM.



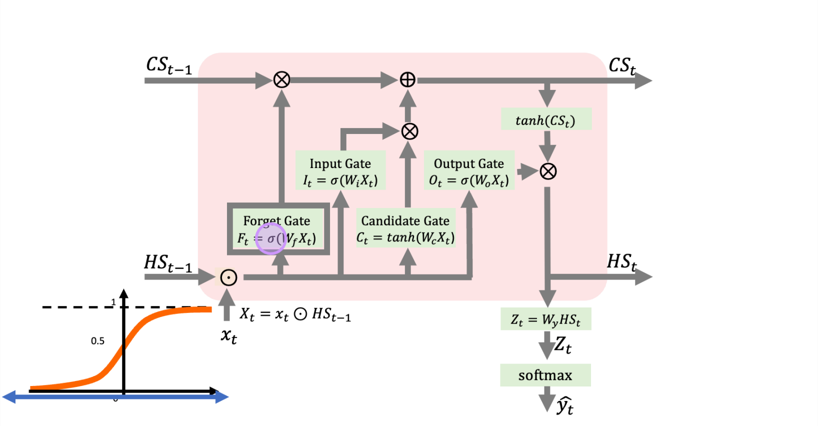
The first thing to note in the Forget Gate is:



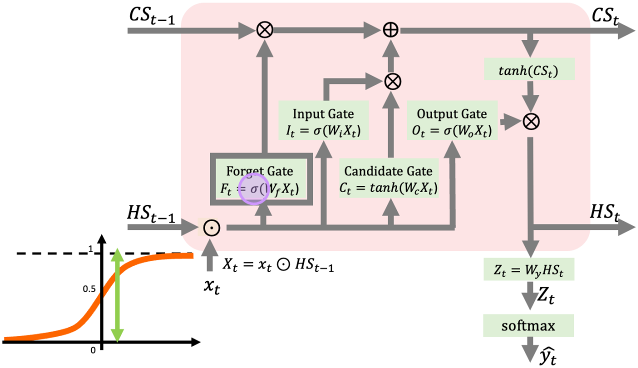
There is a sigmoid function inside the Forget Gate.



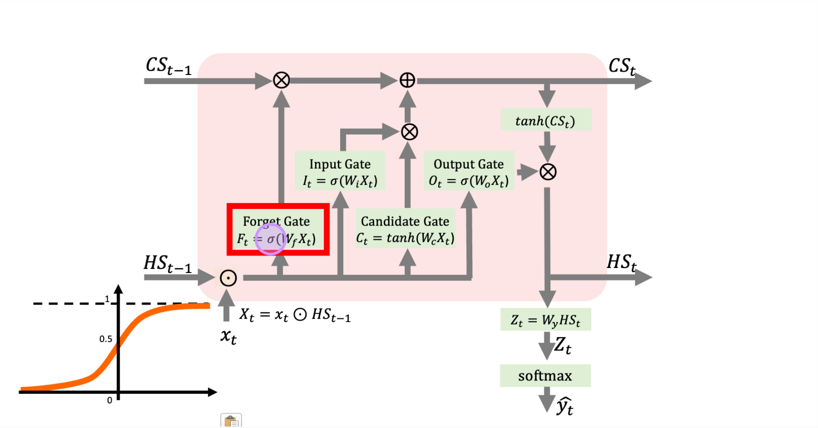
As we learned about the sigmoid, regardless of the input,



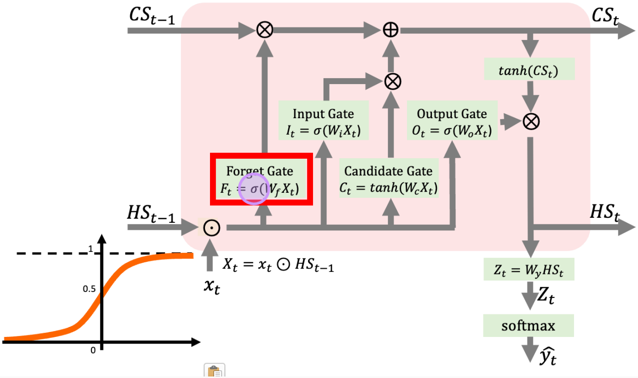
it returns a value between 0 and 1.



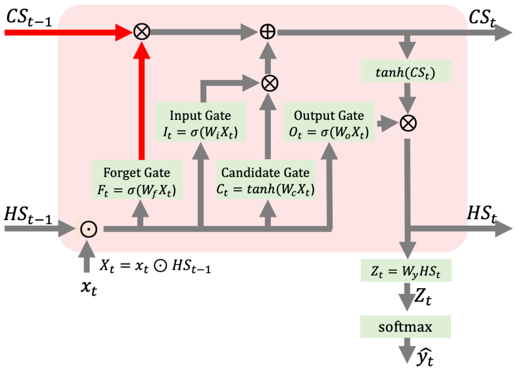
So, what the Forget Gate does is: it takes the (just-prior + current) input, multiplies by weights,



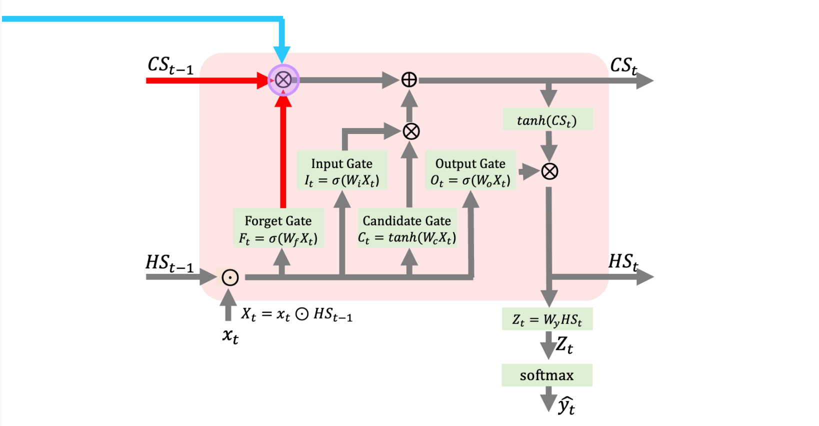
and maps it to values between 0 and 1.



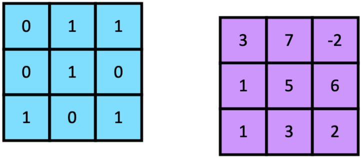
Then, these 0–1 values meet the cell state values



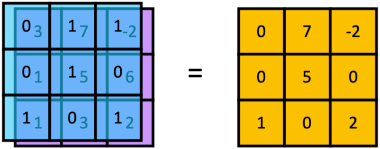
and undergo element-wise multiplication.



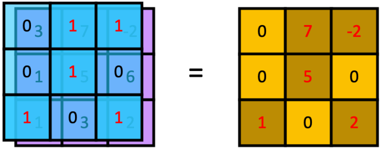
Notes: Element-wise multiplication means **multiplying two matrices by their corresponding elements**.



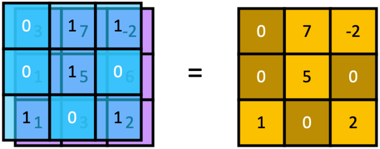
Notes: Element-wise multiplication means **multiplying two matrices by their corresponding elements**.



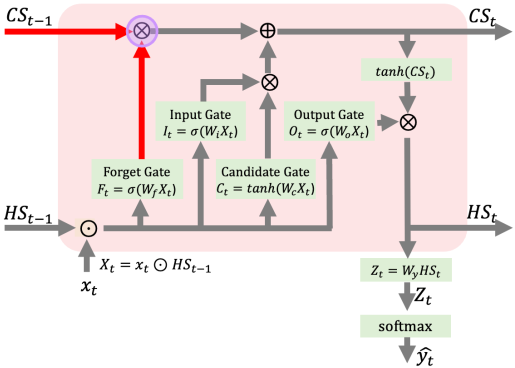
We do this so that entries near 1 are kept



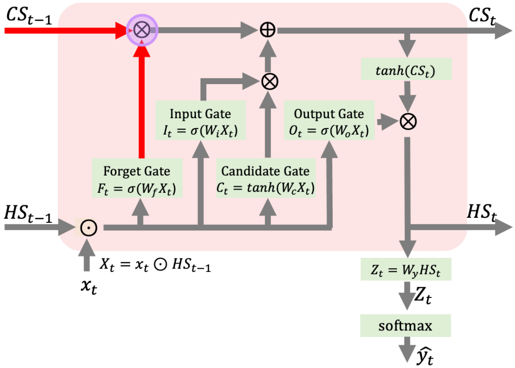
and **entries near 0 are erased (forgotten)**.



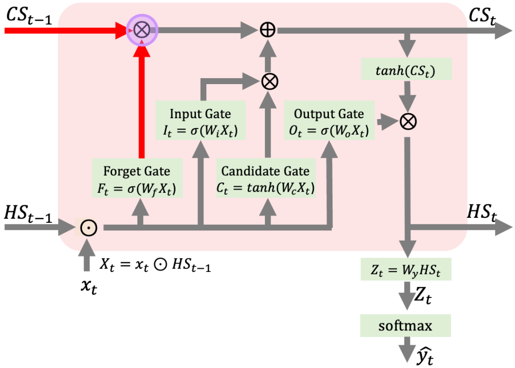
For example, suppose the Forget Gate's output consisted only of 0s and 1s.



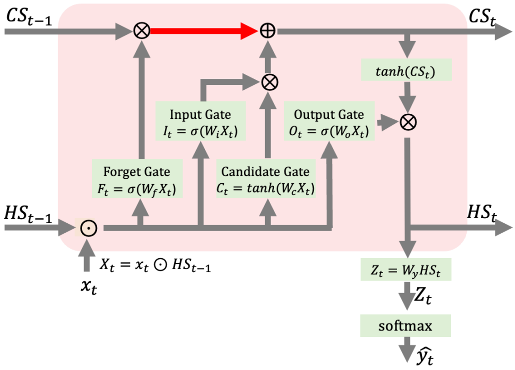
Where the Forget Gate outputs 0, the element-wise product



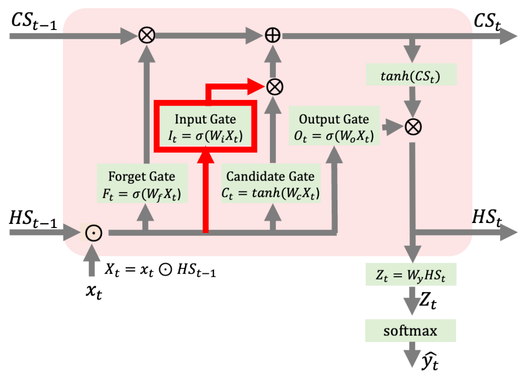
turns those cell-state entries to 0 (or effectively very small).



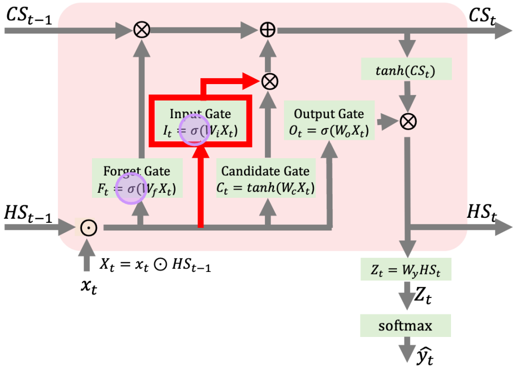
In short, as the cell state (CS) passes through the Forget Gate, it forgets what should be forgotten.



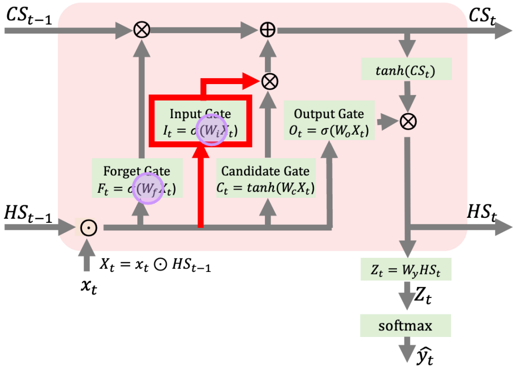
Next, the Input Gate. Its computation is the same pattern as the Forget Gate



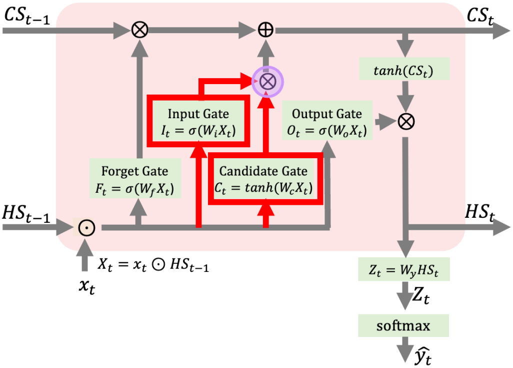
because both use a sigmoid function.



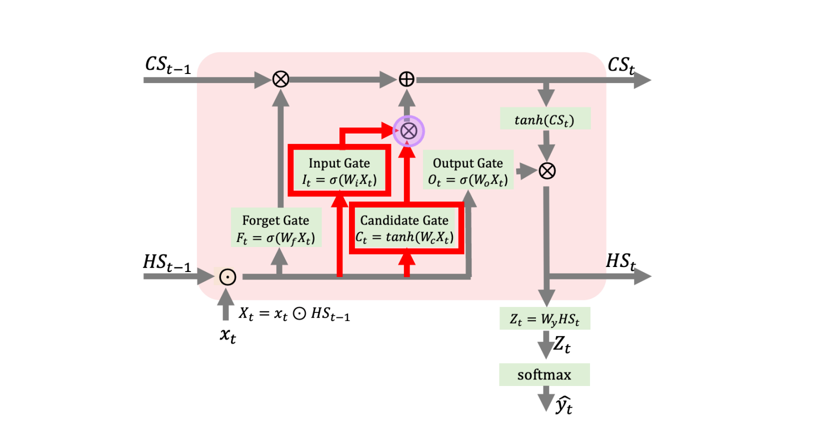
(But) the weights are different.



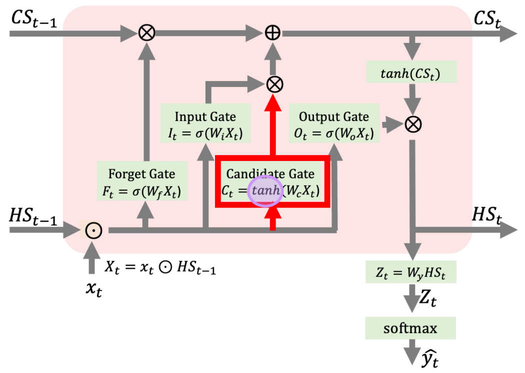
This Input Gate works together with the Candidate Gate



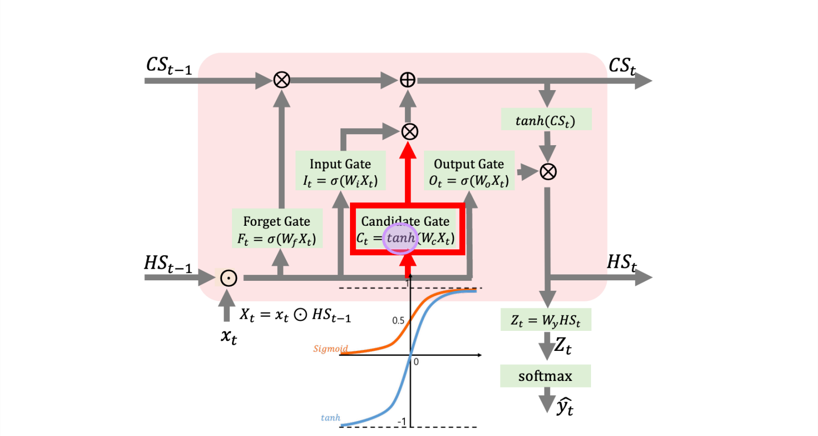
to update the cell state with what should be “remembered.”



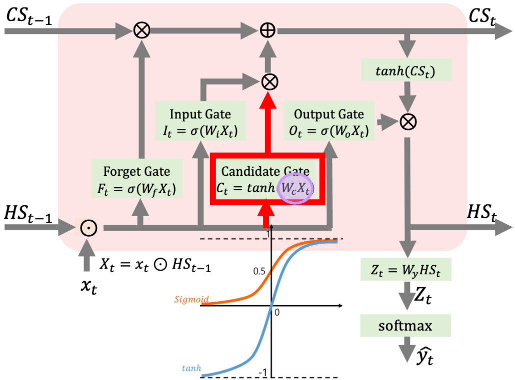
The Candidate Gate uses tanh rather than a sigmoid inside.



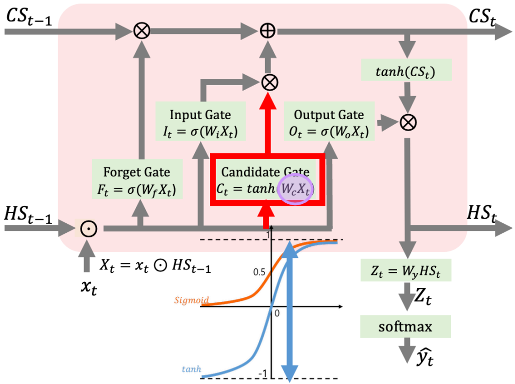
The $\tanh$ function maps inputs to values between -1 and 1.



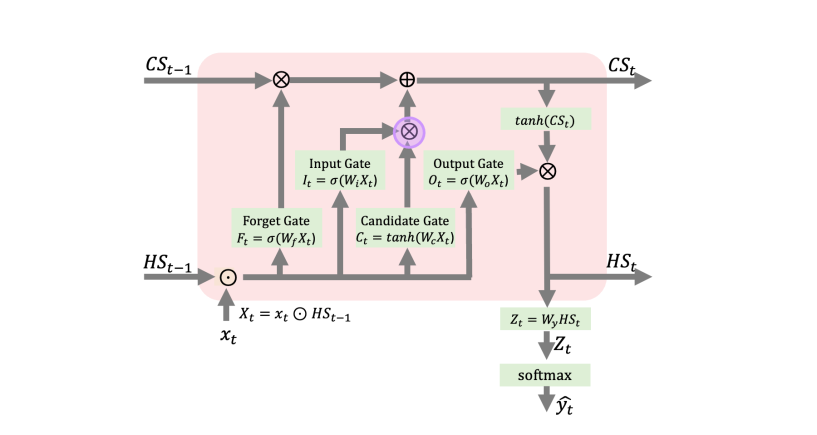
So, what the Candidate Gate does is: multiply the input by weights,



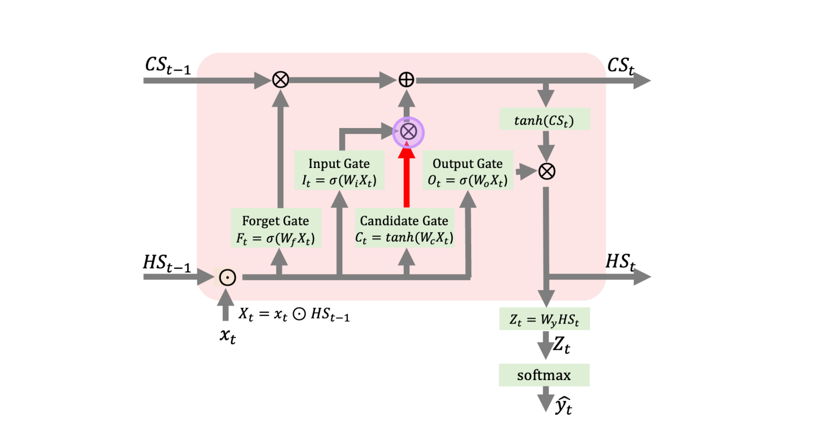
and then preserve the sign while **normalizing the range**.



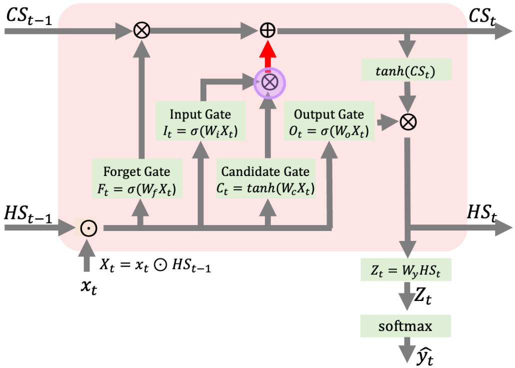
Then, via element-wise multiplication with the 0–1 values from the Input Gate,



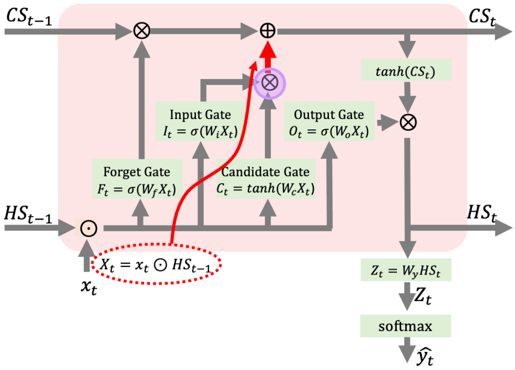
some Candidate outputs are pushed close to 0 while others are kept as they are.



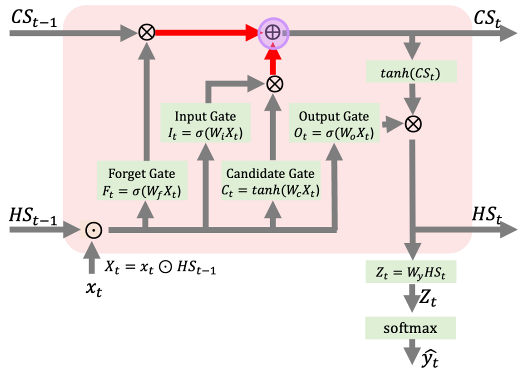
Those kept values



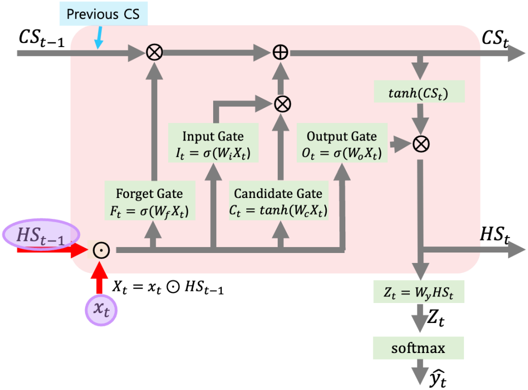
become the parts of the current input (short-term) to be remembered.



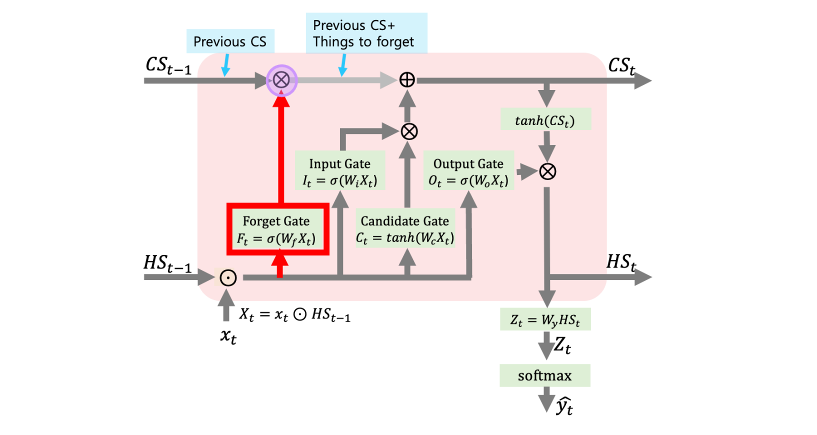
Then the remaining values are added into the cell state to update it.



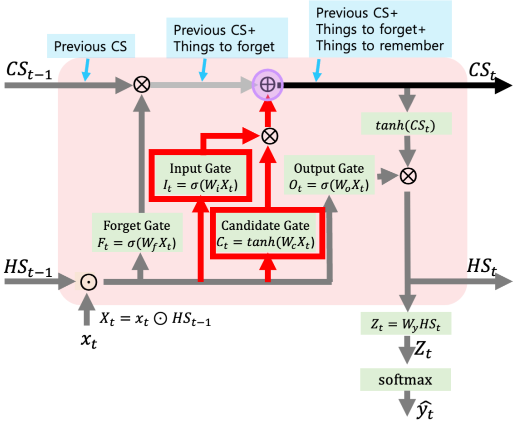
In short, given the previous hidden state and the current input,



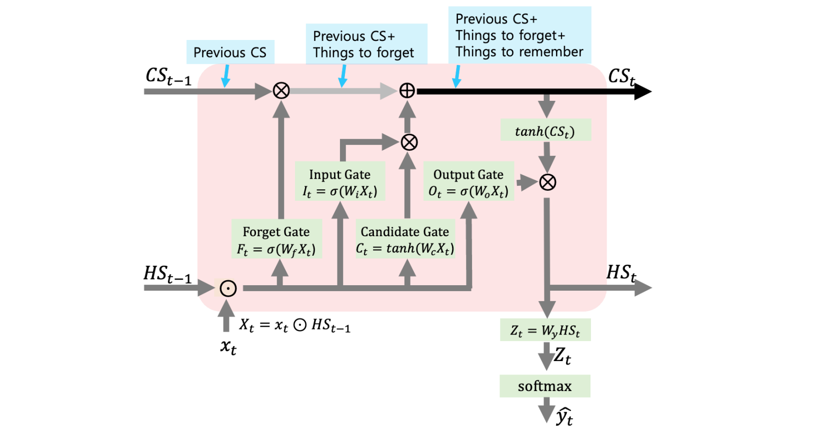
we forget what should be forgotten from the previous cell state,



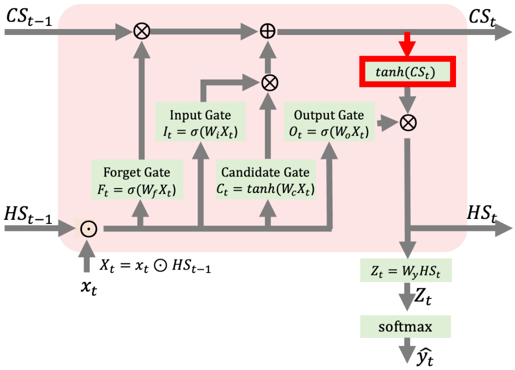
and remember what should be remembered,



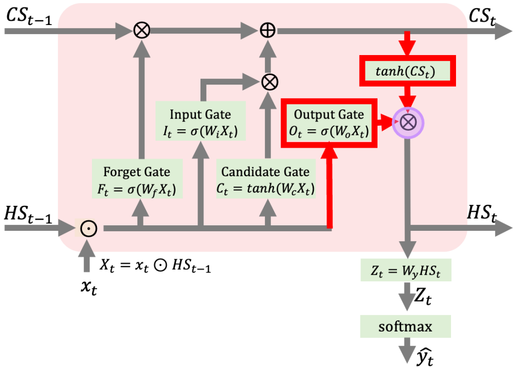
thereby updating LSTM's long-term memory (the cell state).



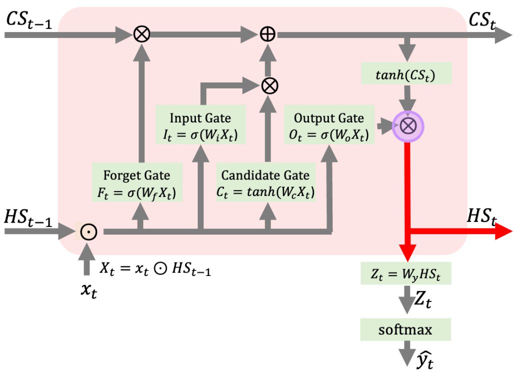
Next, we normalize this long-term state via $\tanh$ (to $[-1,1]$),



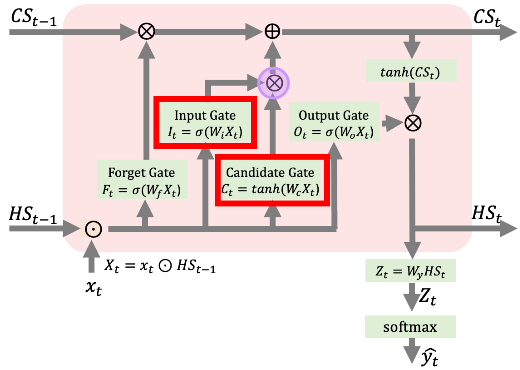
and take an element-wise product with the Output Gate's values.



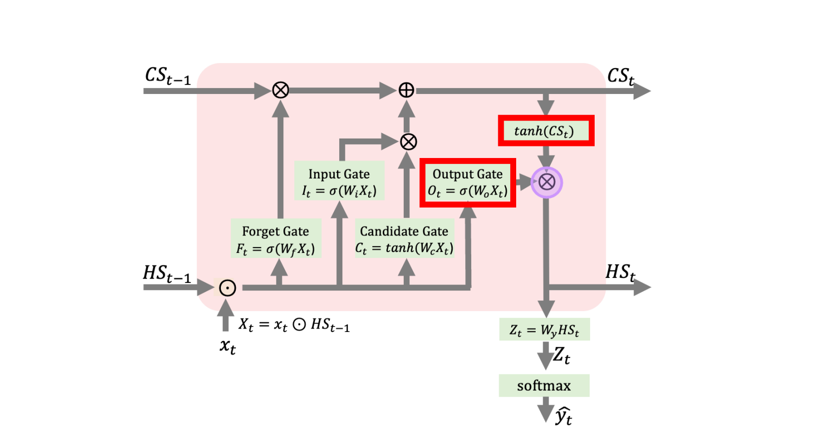
This produces the new hidden state HS_t.



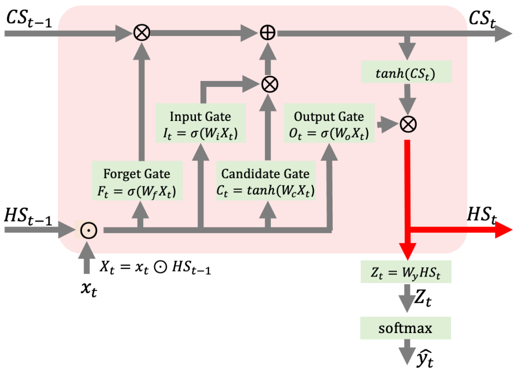
Just as the collaboration of the Input Gate and Candidate Gate keeps the "to-be-remembered" part of the current (short-term) input,



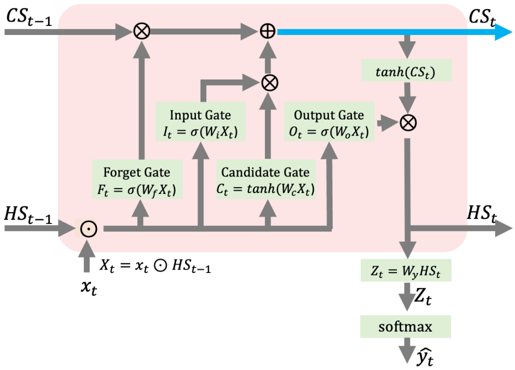
the collaboration of the Output Gate and $\tanh(CS_t)$ creates a new hidden state (HS_t) from the updated cell state (CS_t) that reflects the characteristics of the current input (X_t) more strongly.



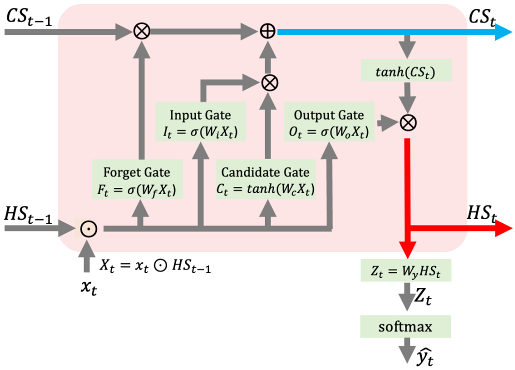
Thus this hidden state (HS_t) tends to show more short-term characteristics than CS_t.



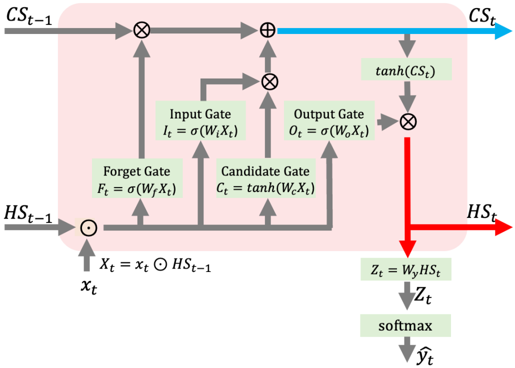
So if CS_t carries more long-term information,



HS_t, given the same inputs, carries information closer to short-term,



and by leveraging these two information flows, LSTM can **handle long-term dependency problems more effectively** than a vanilla RNN.



2. Real-world success in NLP tasks

- Introduced by Hochreiter & Schmidhuber (1997)

2. Real-world success in NLP tasks

- Introduced by Hochreiter & Schmidhuber (1997)
- In 2013-2025, LSTMs started achieving state-of-the-art results in NLP tasks, including:

2. Real-world success in NLP tasks

- Introduced by Hochreiter & Schmidhuber (1997)
- In 2013-2025, LSTMs started achieving state-of-the-art results in NLP tasks, including:
  - handwriting recognition, speech recognition, machine translation, parsing, and image captioning, as well as language modeling

2. Real-world success in NLP tasks

- Introduced by Hochreiter & Schmidhuber (1997)
- In 2013-2025, LSTMs started achieving state-of-the-art results in NLP tasks, including:
  - handwriting recognition, speech recognition, machine translation, parsing, and image captioning, as well as language modeling
- Recently (2019-2025), Transformers have become dominant for all tasks

2. Real-world success in NLP tasks

- Introduced by Hochreiter & Schmidhuber (1997)
- In 2013-2025, LSTMs started achieving state-of-the-art results in NLP tasks, including:
  - handwriting recognition, speech recognition, machine translation, parsing, and image captioning, as well as language modeling
- Recently (2019-2025), Transformers have become dominant for all tasks
  - For example, in WMT (a Machine Translation conference/competition):

2. Real-world success in NLP tasks

- Introduced by Hochreiter & Schmidhuber (1997)
- In 2013-2025, LSTMs started achieving state-of-the-art results in NLP tasks, including:
  - handwriting recognition, speech recognition, machine translation, parsing, and image captioning, as well as language modeling
- Recently (2019-2025), Transformers have become dominant for all tasks
  - For example, in WMT (a Machine Translation conference/competition):
    - In WMT 2014, there were 0 neural machine translation systems

2. Real-world success in NLP tasks

- Introduced by Hochreiter & Schmidhuber (1997)
- In 2013-2025, LSTMs started achieving state-of-the-art results in NLP tasks, including:
  - handwriting recognition, speech recognition, machine translation, parsing, and image captioning, as well as language modeling
- Recently (2019-2025), Transformers have become dominant for all tasks
  - For example, in WMT (a Machine Translation conference/competition):
    - In WMT 2014, there were 0 neural machine translation systems
    - In WMT 2016, the summary report contains RNN 44 times (and these systems won)

2. Real-world success in NLP tasks

- Introduced by Hochreiter & Schmidhuber (1997)
- In 2013-2025, LSTMs started achieving state-of-the-art results in NLP tasks, including:
  - handwriting recognition, speech recognition, machine translation, parsing, and image captioning, as well as language modeling
- Recently (2019-2025), Transformers have become dominant for all tasks
  - For example, in WMT (a Machine Translation conference/competition):
    - In WMT 2014, there were 0 neural machine translation systems
    - In WMT 2016, the summary report contains RNN 44 times (and these systems won)
    - In WMT 2019: RNN 7 times, Transformer 105 times

Bidirectional/multi-layer RNNs/LSTMs

- A standard RNN only uses **past context**.

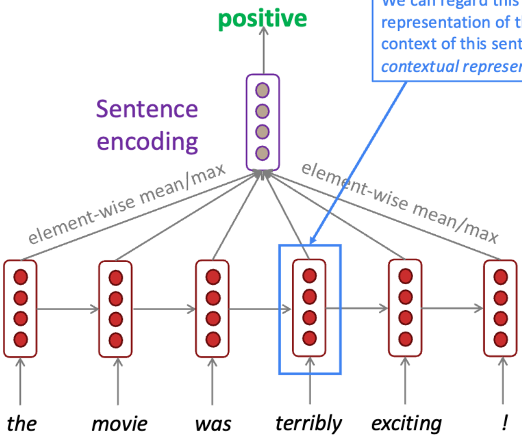
1. Motivation

- A standard RNN only uses **past context**.
- For many NLP tasks (e.g., tagging, parsing, translation), knowing **both previous and future contexts** improves predictions.

1. Motivation

- A standard RNN only uses **past context**.
- For many NLP tasks (e.g., tagging, parsing, translation), knowing **both previous and future contexts** improves predictions.
- Bidirectional RNNs address this by **processing the sequence in both directions**.

Task: Sentiment Classification

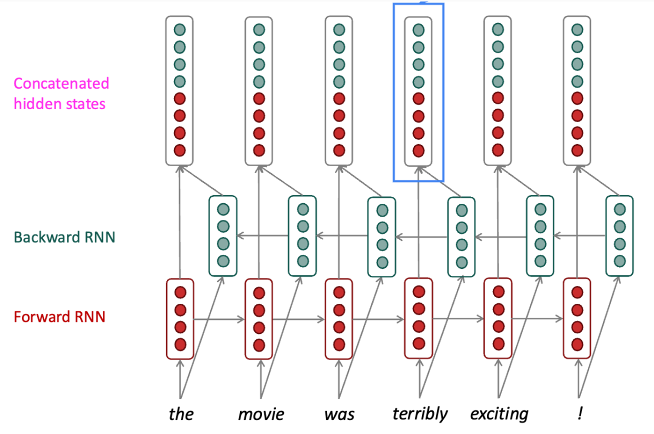


We can regard this hidden state as a representation of the word *"terribly"* in the context of this sentence. We call this a *contextual representation.*

These contextual representations only contain information about the *left* context (e.g. *"the movie was"*).

What about *right* context?

In this example, *"exciting"* is in the right context and this modifies the meaning of *"terribly"* (from negative to positive)



Forward + Backward: The contextual representation of "terribly" has both left and right context.

On timestep *t*:

This is a general notation to mean "compute one forward step of the RNN" – it could be a simple RNN or LSTM computation.

Forward RNN $\overrightarrow{\boldsymbol{h}}^{(t)} = \boxed{\mathrm{RNN}_{\mathrm{FW}}}(\overrightarrow{\boldsymbol{h}}^{(t-1)}, \boldsymbol{x}^{(t)})$

Backward RNN $\overleftarrow{\boldsymbol{h}}^{(t)} = \mathrm{RNN}_{\mathrm{BW}}(\overleftarrow{\boldsymbol{h}}^{(t+1)}, \boldsymbol{x}^{(t)})$

Generally, these two RNNs have separate weights

Concatenated hidden states $\boxed{\boldsymbol{h}^{(t)}} = [\overrightarrow{\boldsymbol{h}}^{(t)}; \overleftarrow{\boldsymbol{h}}^{(t)}]$

We regard this as "the hidden state" of a bidirectional RNN. This is what we pass on to the next parts of the network.

2. More information

- Hidden states combine forward and backward context.

2. More information

- Hidden states combine forward and backward context.
- Require access to the entire sequence (not suitable for language modeling).

2. More information

- Hidden states combine forward and backward context.
- Require access to the entire sequence (not suitable for language modeling).
- Very effective for encoding tasks (e.g., tagging, parsing, translation).

2. More information

- Hidden states combine forward and backward context.
- Require access to the entire sequence (not suitable for language modeling).
- Very effective for encoding tasks (e.g., tagging, parsing, translation).
- Example: BERT (Bidirectional Encoder Representations from Transformers) leverages bidirectionality for powerful contextual embeddings.

2. More information

- Hidden states combine forward and backward context.
- Require access to the entire sequence (not suitable for language modeling).
- Very effective for encoding tasks (e.g., tagging, parsing, translation).
- Example: BERT (Bidirectional Encoder Representations from Transformers) leverages bidirectionality for powerful contextual embeddings.
- Can be extended by stacking layers (Multi-layer RNNs).

Wrap-up

Wrap-up

- RNNs

Wrap-up

- RNNs
- Problems with RNNs: Vanishing & Exploding

Wrap-up

- RNNs
- Problems with RNNs: Vanishing & Exploding
- LSTMs: Short-term/Long-term

Wrap-up

- RNNs
- Problems with RNNs: Vanishing & Exploding
- LSTMs: Short-term/Long-term
    - Four gates

Wrap-up

- RNNs
- Problems with RNNs: Vanishing & Exploding
- LSTMs: Short-term/Long-term
    - Four gates
    - 1. Forget gates

Wrap-up

- RNNs
- Problems with RNNs: Vanishing & Exploding
- LSTMs: Short-term/Long-term
    - Four gates
    - 1. Forget gates
    - 2. Input Gate

Wrap-up

- RNNs
- Problems with RNNs: Vanishing & Exploding
- LSTMs: Short-term/Long-term
    - Four gates
    - 1. Forget gates
    - 2. Input Gate
    - 3. Candidate Gate

Wrap-up

- RNNs
- Problems with RNNs: Vanishing & Exploding
- LSTMs: Short-term/Long-term
    - Four gates
    - 1. Forget gates
    - 2. Input Gate
    - 3. Candidate Gate
    - 4. Output Gates

Wrap-up

- RNNs
- Problems with RNNs: Vanishing & Exploding
- LSTMs: Short-term/Long-term
    - Four gates
    - 1. Forget gates
    - 2. Input Gate
    - 3. Candidate Gate
    - 4. Output Gates
- Bidirectional RNNs for more context

Where we are at

6	9/30	Translation, Seq2Seq, Attention	
	10/2	Lab 6 – RNNs	Lab exercise 6
7	10/7	Self-attention & Transformer	
	10/9	Group meeting	Background research topic submission
8	10/14	**Fall break (No class)**	
	10/16	Quiz (Online)	

Reminder

1. Background research brief

Released on Tuesday 09/16/2025

Each groups should submit the following to prepare your background-research presentation and to seed your final presentation/paper. Please aim to have a working draft ready for your group check-in on October 9th. After the group meeting, the final version of the draft should be submitted by October 10th (Friday). This is not a graded assignment.

Things to include

1. **Topic / Area**
   - One sentence stating the focus
   - 3–5 keywords
2. **Research question / Problem**
   - 1–2 sentences clearly stating the core question or hypothesis
3. **Mini annotated bibliography (3–5 papers)** — for **each** paper include:
   - Full citation (consistent style)
   - 1-sentence contribution (key finding/idea)
   - Method/Data (e.g., corpus, model, experiment)
   - Relevance (why it matters for your group project)