Recurrent Neural Networks

Sudong Lee

sudonglee@ulsan.ac.kr

https://dais.ulsan.ac.kr/

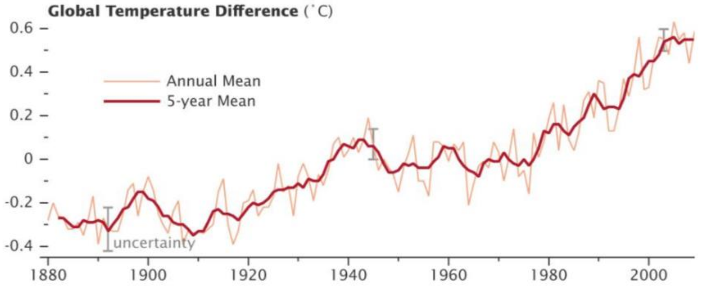
Contents

- Introduction to Sequential Data
- Vanilla Recurrent Neural Network
- LSTM (Long Short-Term Memory)

Introduction to Sequential Data

What is Sequential Data?

"Sequential data refer to data in which the order of instances matters."

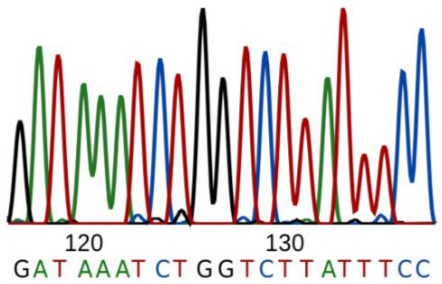


Time Series

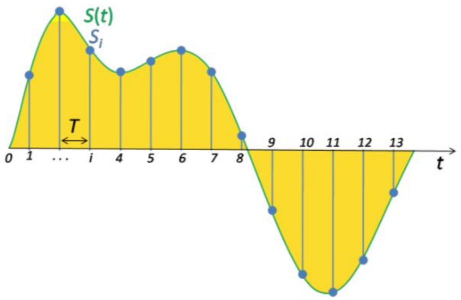
"I am a boy."

"Am I a boy?"

Natural language



DNA sequencing



Audio sampling

Image Source | https://heung-bae-lee.github.io/2020/01/12/deep_learning_08/

What is Sequential Data?

"Sequential data have the temporal dependency, so the order of instances matters."

- Notations
  - Non-sequential:

$$\boldsymbol{X}^T = \left[x_{i,j}\right] \text{ for } i = 1, \dots, n; j = 1, \dots, p. \quad (n \times p)$$

  - Sequential:

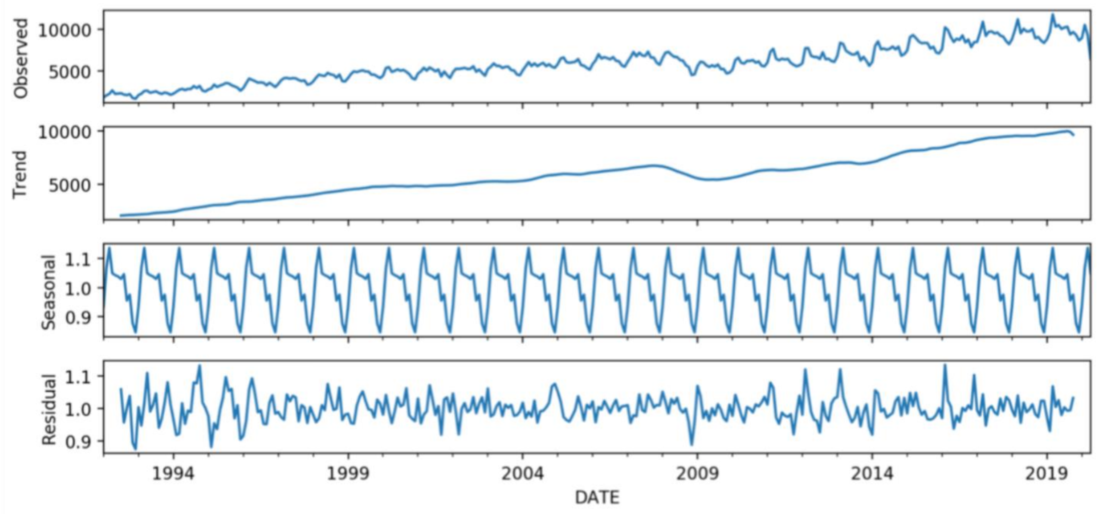
$$\boldsymbol{X}^T = [x_i(t)] \text{ for } i = 1, \dots, p;\ t = 1, \dots, T \quad (T \times p)$$

$$\boldsymbol{x}(t) = \left[x_1(t), \dots, x_p(t)\right]^T$$

$$\boldsymbol{X} = [\boldsymbol{x}(1), \dots, \boldsymbol{x}(T)]$$

Types of Sequential Data: Time Series

"Time series data is a sequence of data points collected or recorded at regular intervals."



Decomposition of a used car sales

<u>Characteristics of Time Series Data</u>

- **Temporal ordering**

  data points are arranged chronologically.
- **Regular intervals**

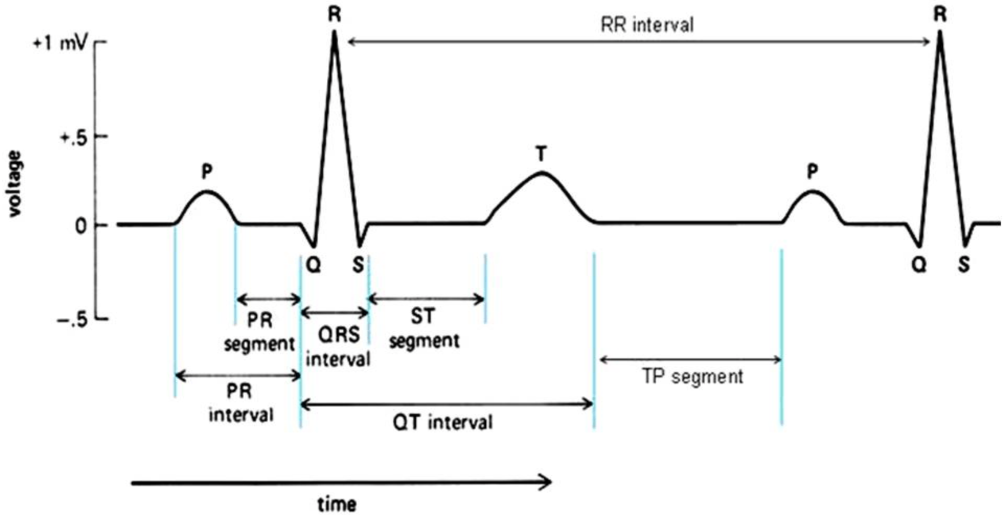
  collected at consistent time intervals (e.g., hourly, daily, ⋯)
- **Dependency**

  each data point is related to or influenced by previous points.

Image Source | https://timeseriesreasoning.com/contents/time-series-decomposition/?ref=timescale.com

Types of Sequential Data: Signal

"Signal data imply a continuous or discrete series of measurements over time or space."



ECG (Electrocardiogram) signal data

<u>Characteristics of Signal Data</u>

- **Time-varying**

  represents how a quantity changes over time.

- **Continuous or discrete**

  continuous or discrete in nature.

- **Amplitude and frequency**

  characterized by their amplitude (strength) and frequency (rate of change).

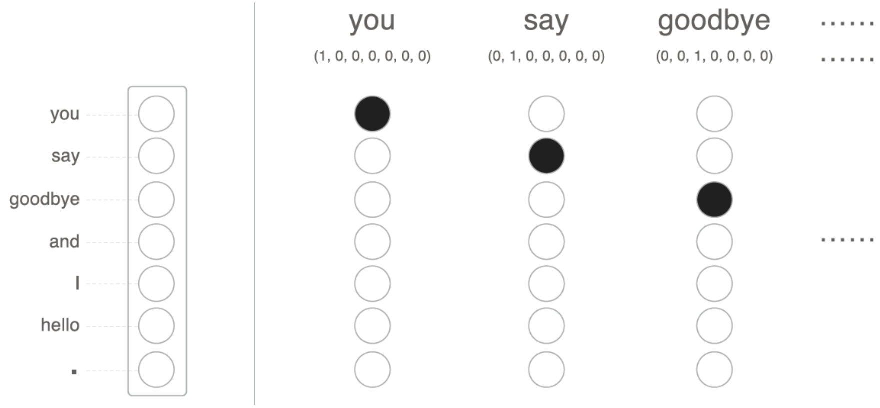
Image Source | https://kr.mathworks.com/help/dsp/ug/real-time-ecg-qrs-detection.html

Types of Sequential Data: Natural Language

"Natural language data expressed in human language are encoded as vectors."

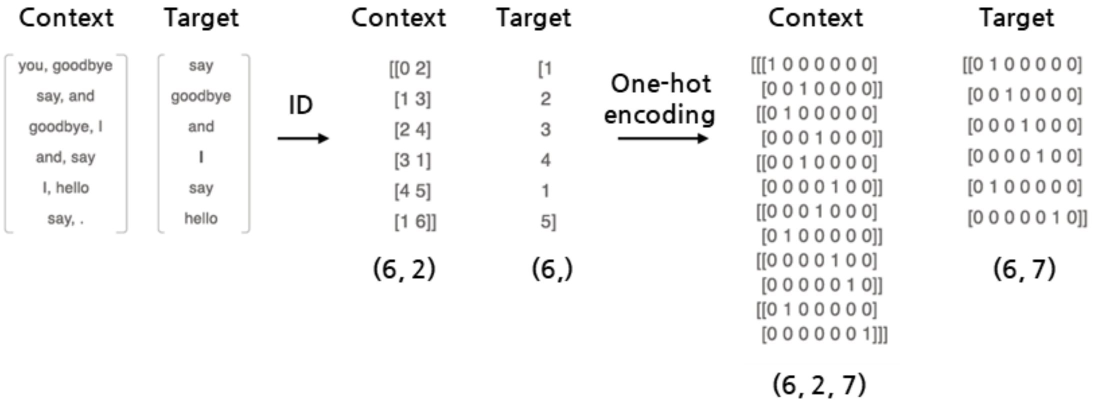
Corpus: "you say goodbye and I say hello."

Word (text)	ID (label)	One-hot encoding
you goodbye	0 2	(1, 0, 0, 0, 0, 0, 0) (0, 0, 1, 0, 0, 0, 0)



One-hot encoding of words using a bag-of-words

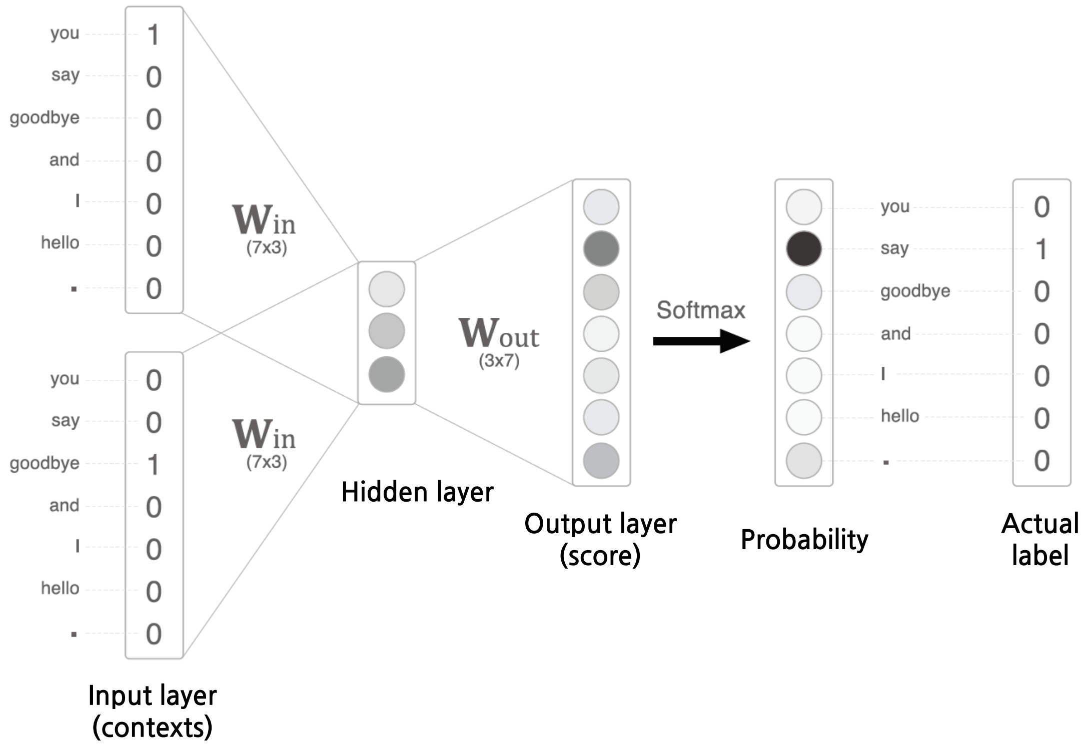
Inference: "you [?] goodbye and I say hello."



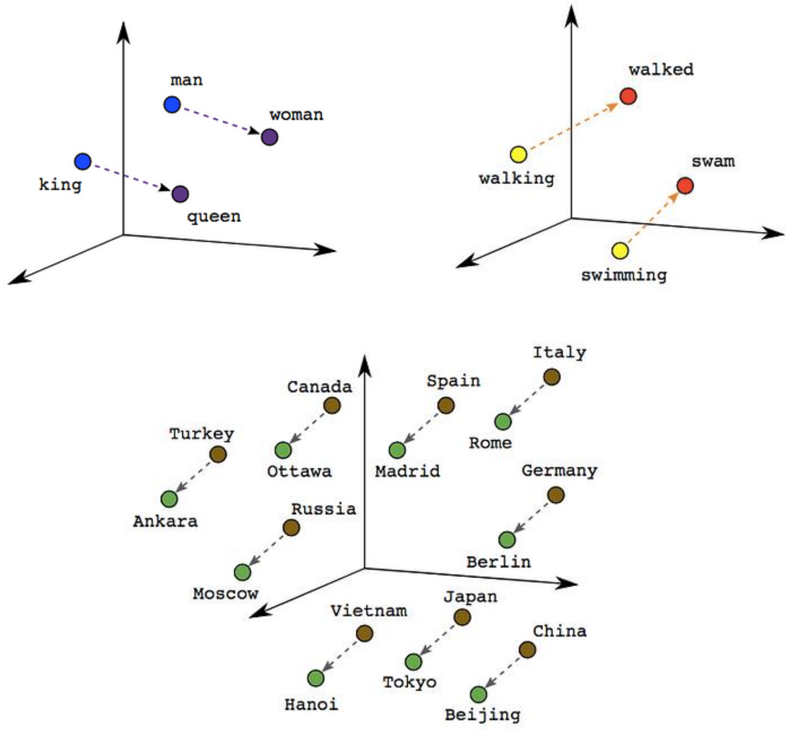
One-hot encoding of words using CBOW (continuous bag of words)

Types of Sequential Data: Natural Language

"Natural language data expressed in human language are encoded as vectors."



word2vec using CBOW

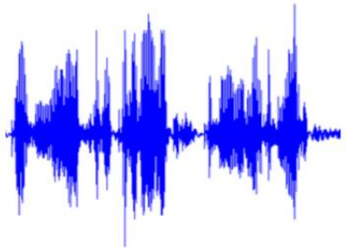


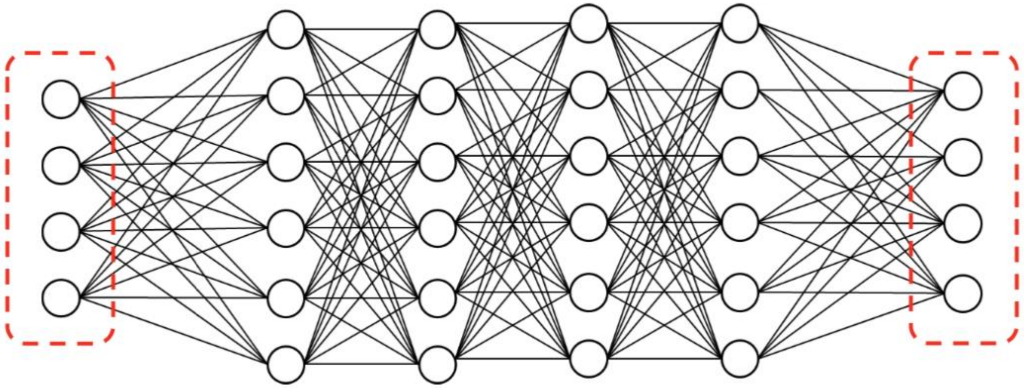
Word embeddings by word2vec

Vanilla Recurrent Neural Network

Representation of Sequential Data

"How can we effectively extract the representation of sequential data using deep learning?"





"I am a boy" $\neq$ "am I a boy"

Representation of Sequential Data

"How can we effectively extract the representation of sequential data using deep learning?"

Order	Input
1	What
2	Time
3	Is
4	it?

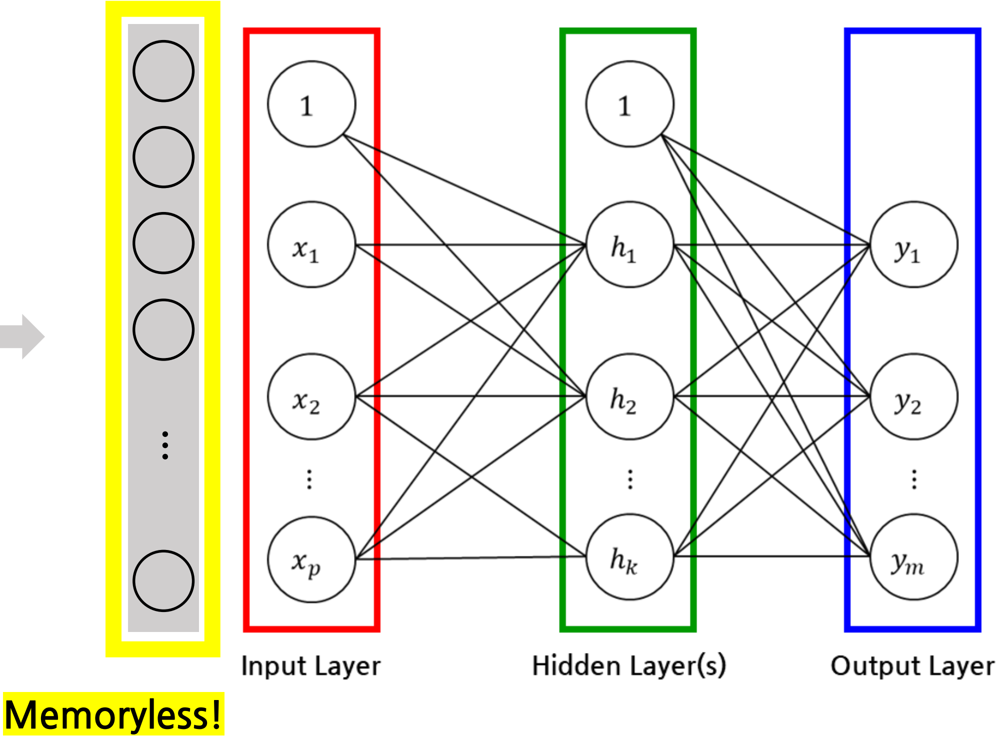
→ Sequential Data Processor → "It's 20:26."

"I am a boy" ≠ "am I a boy"

Limitation of Vanilla Neural Network for Sequential Data

"Independent nodes in the input layer are memoryless of the input sequence."

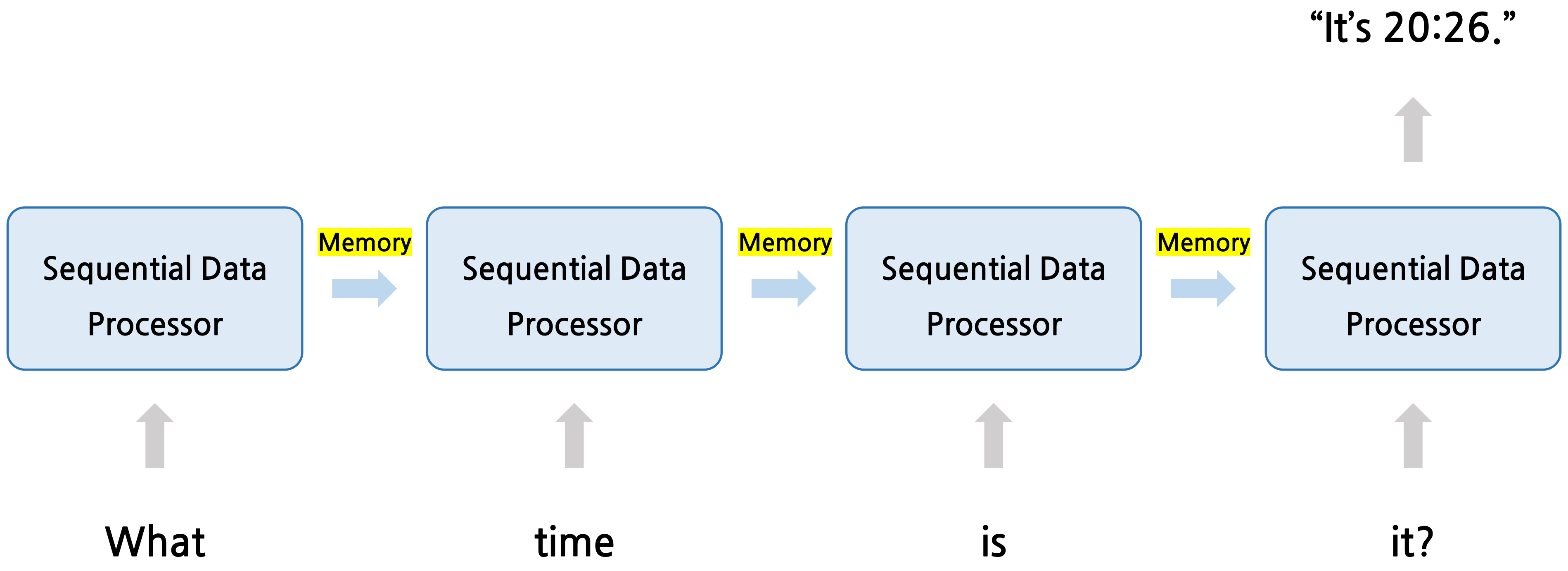
Order	Input
1	What
2	Time
3	Is
4	it?



"???? "

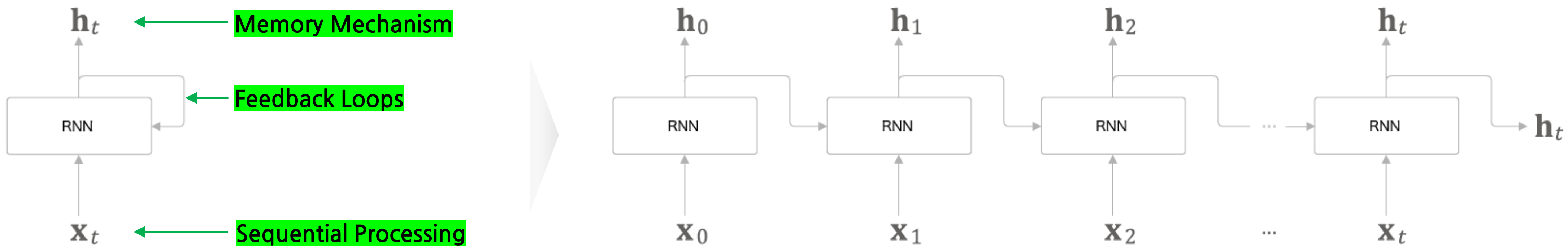
Memory System

"A sequential data processor is a memory system that can memorize the input order."



Recurrent Neural Network (RNN)

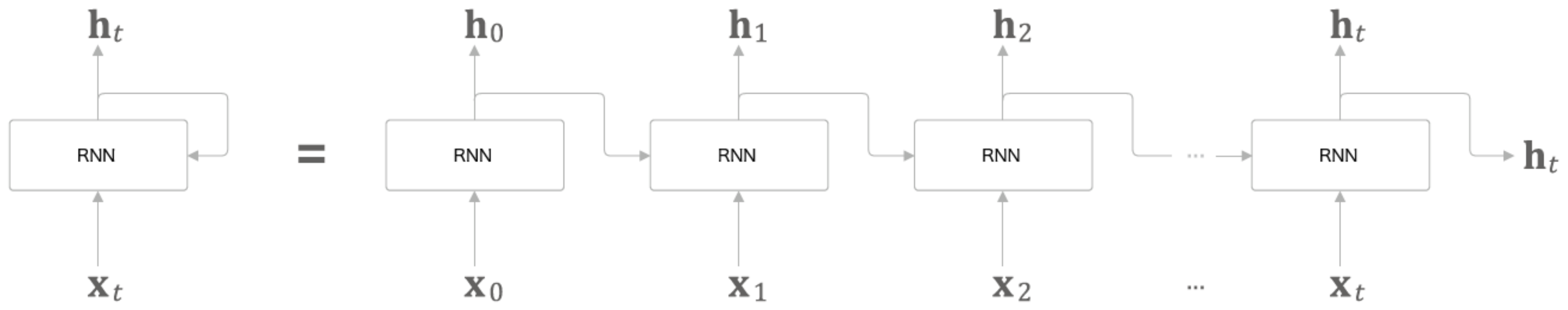
"A recurrent neural network recurrently applies a sequence processor using *hidden states.*"



- $\boldsymbol{x}_t$: the input vector at time t.
- $\boldsymbol{h}_t$: the hidden state vector at time t.

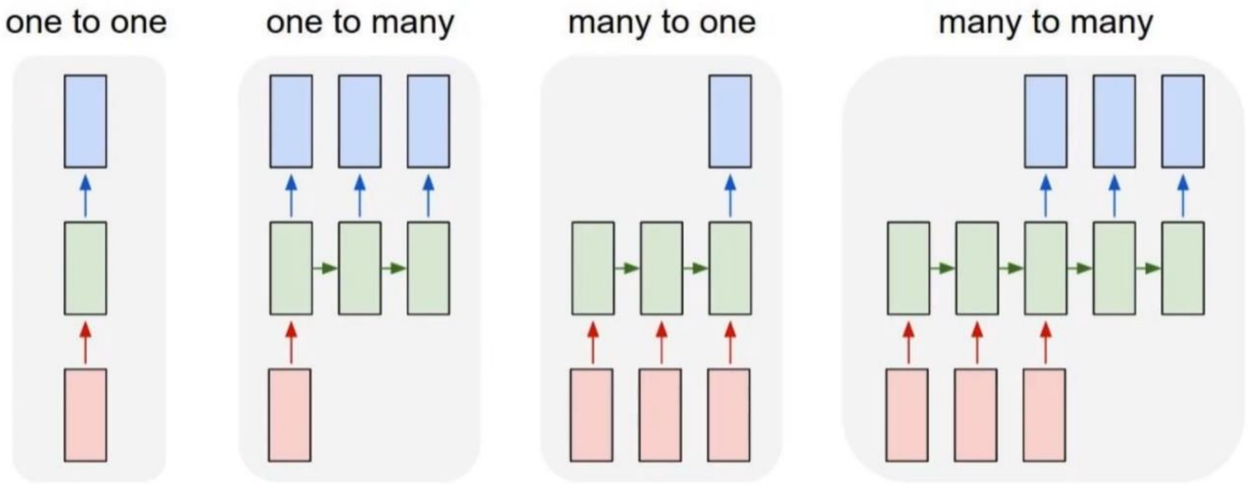
Recurrent Neural Network (RNN)

"A recurrent neural network recurrently applies a sequence processor using *hidden states.*"



- $\boldsymbol{h}_t = \tanh(\boldsymbol{h}_{t-1}^T \boldsymbol{W}_{hh} + \boldsymbol{x}_t \boldsymbol{W}_{xh} + \boldsymbol{b}_h)$
- $\boldsymbol{y}_t = \tanh(\boldsymbol{h}_t \boldsymbol{W}_{hy} + \boldsymbol{b}_y)$

Various Usage of RNN



Examples

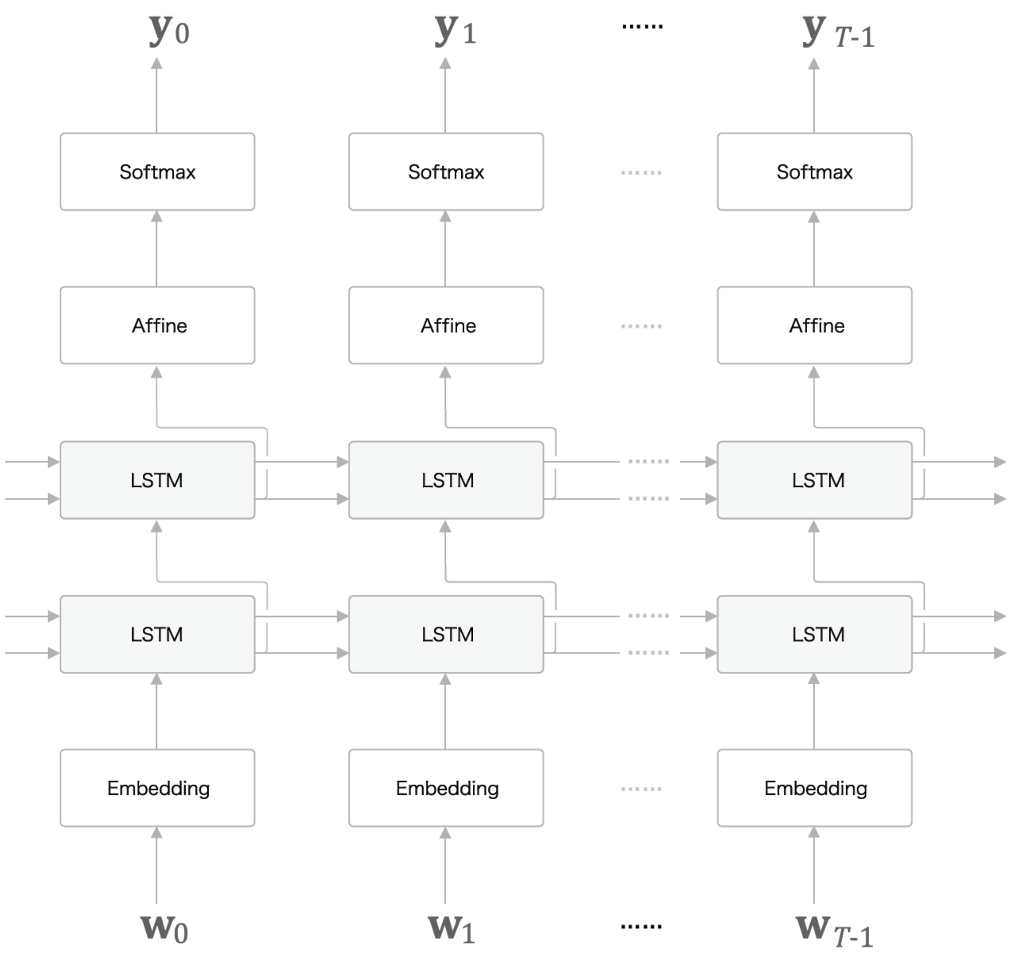
① One-to-one: MLP

② One-to-many: description of an input image

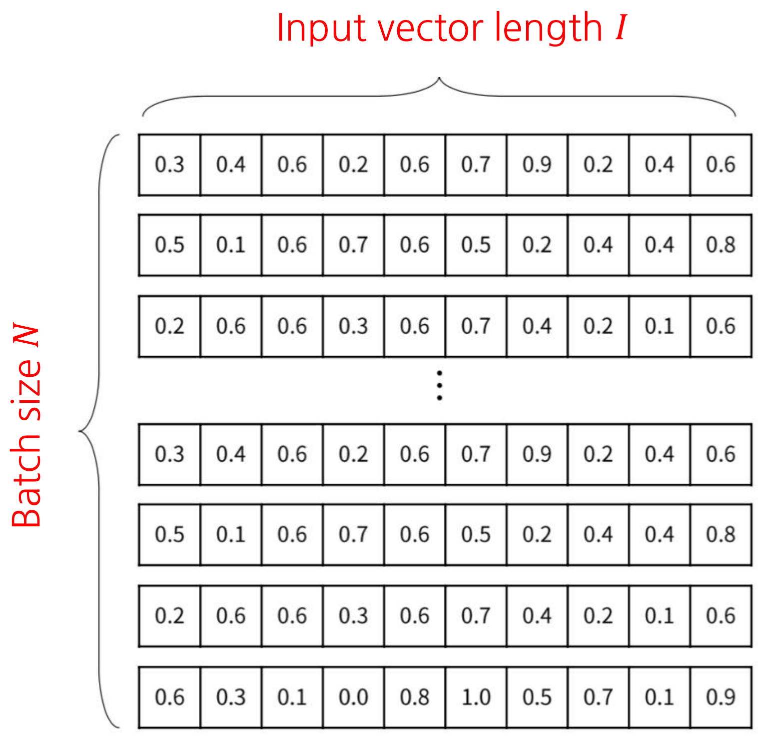
③ Many-to-one: time series classification

④ Many-to-many: machine translation

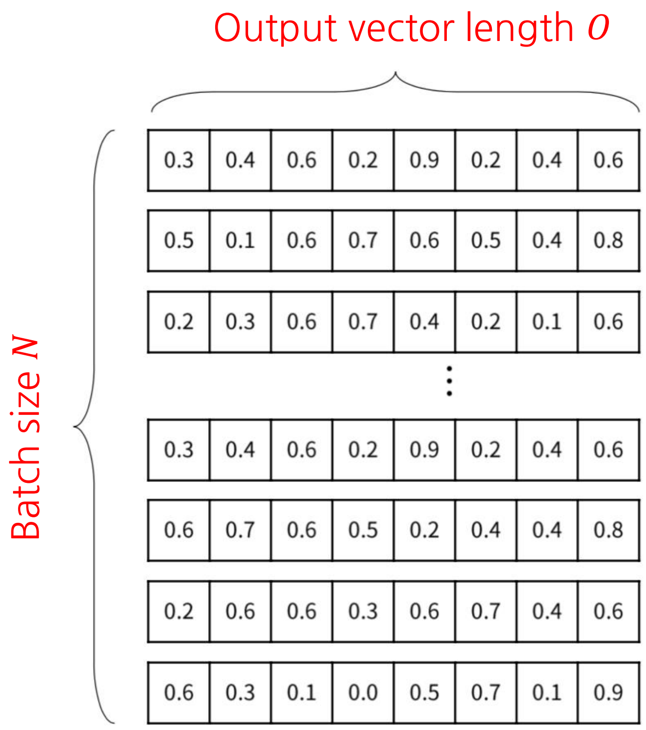
Example: A Multi-layered LSTM for the Many-to-Many Problem



Understanding RNNs from a Tensor Perspective

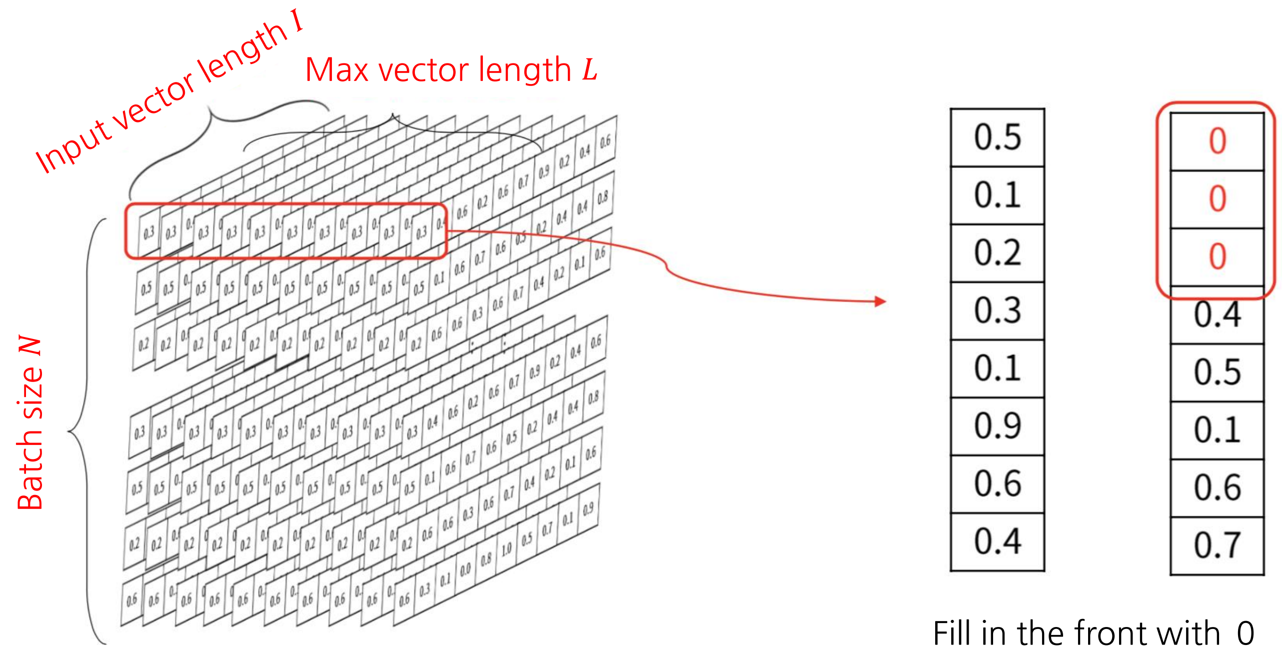


Input tensor ($N \times I$)



Output tensor ($N \times O$)

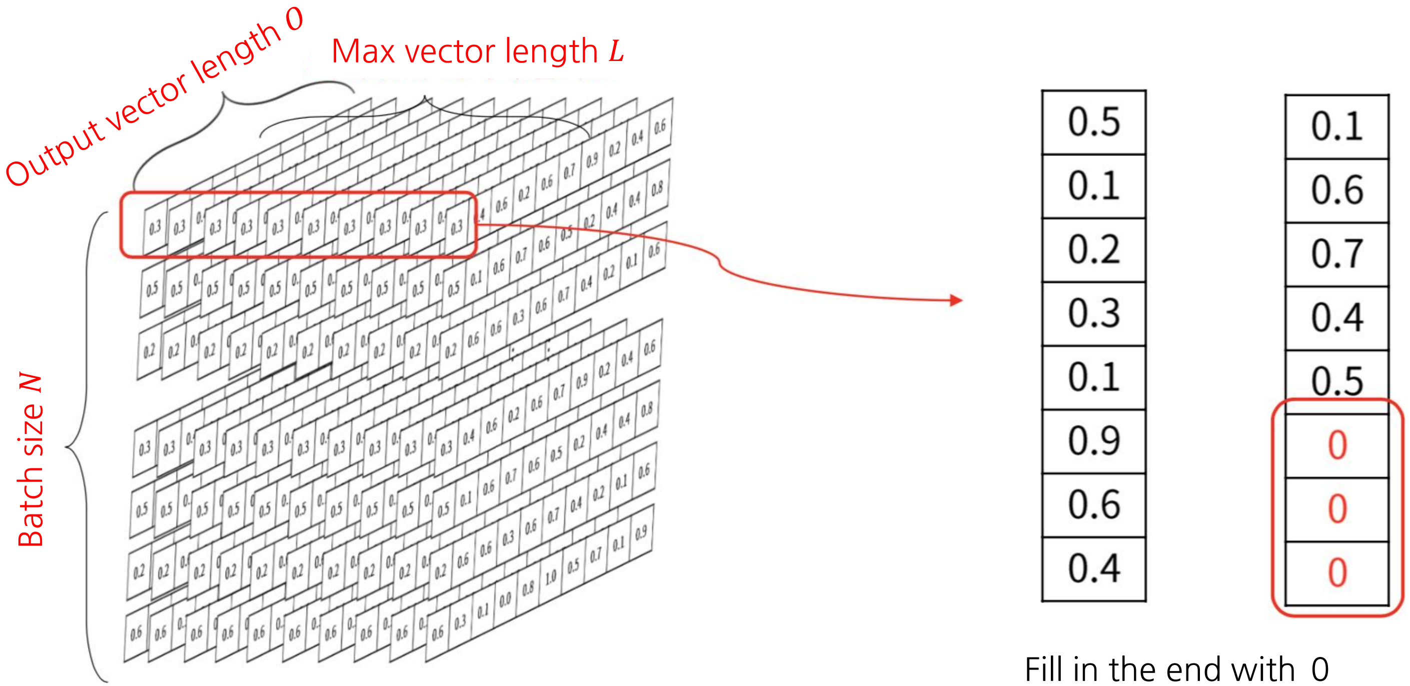
Understanding RNNs from a Tensor Perspective



Input tensor $(N \times L \times I)$

Fill in the front with 0
when the length is shorter than L.

Understanding RNNs from a Tensor Perspective



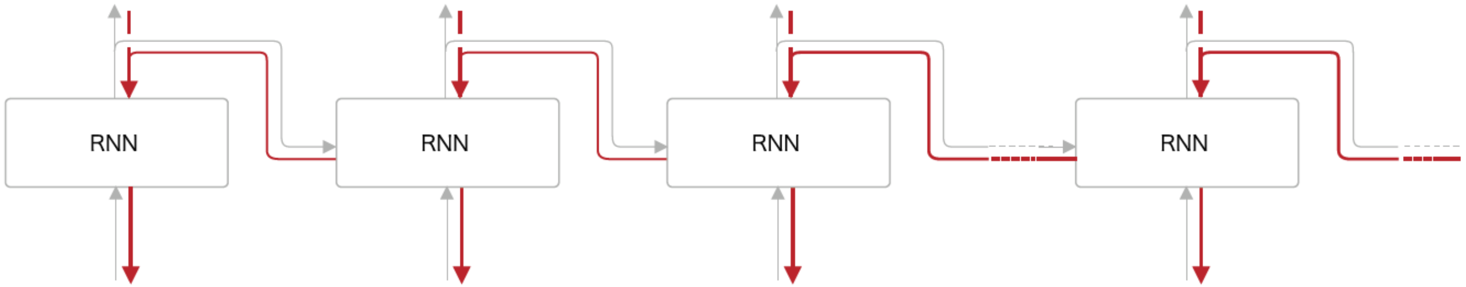
Output tensor $(N \times L \times O)$

Fill in the end with 0

when the length is shorter than L.

BPTT: Backpropagation Through Time

"BPTT is an extension of the standard backpropagation algorithm adapted for the recurrent structure of RNN."



Challenges and Limitations of BPTT

- **Vanishing and exploding gradients:**

  As gradients are propagated back through many time steps,

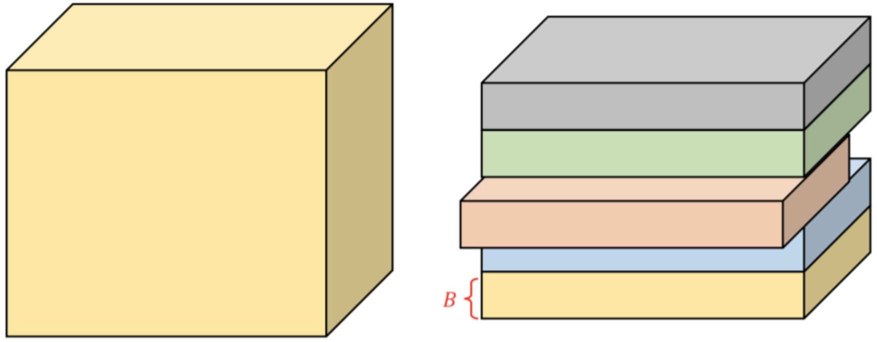
  they can become very small (vanishing) or very large (exploding).

- **Computational complexity:**

  Computationally expensive, especially for long sequences, as it requires storing activations for all time steps.

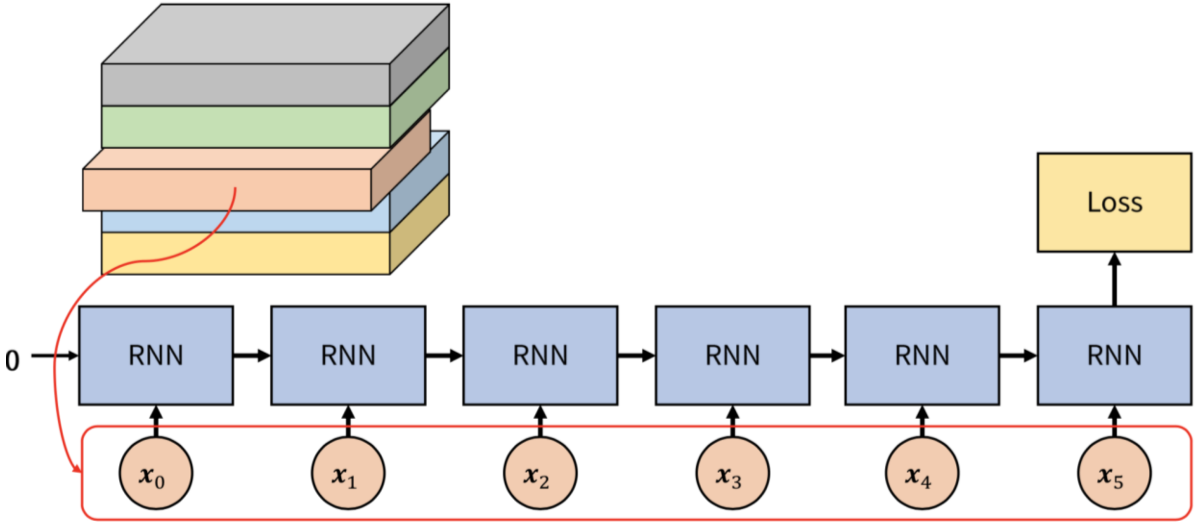
In practice, BPTT is often truncated to a certain number of time steps to manage computational costs, which can limit the network's ability to learn long-term dependencies.

BPTT: Backpropagation Through Time

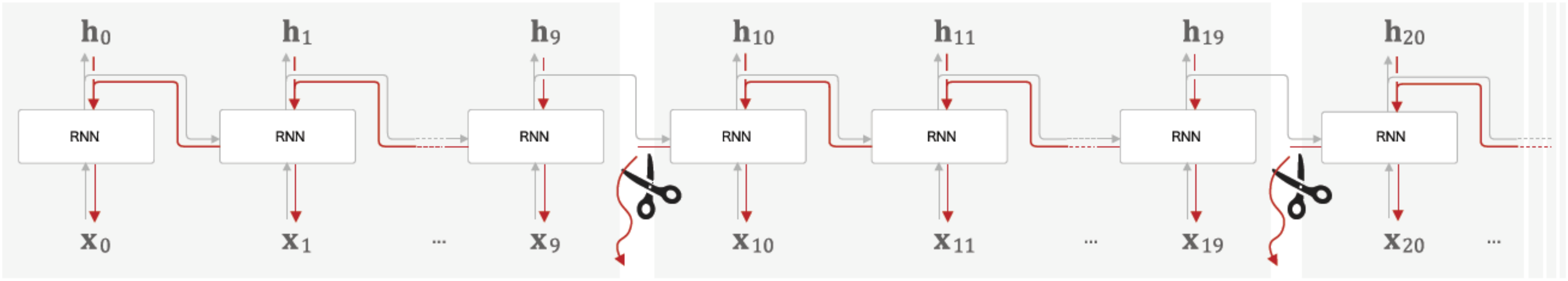


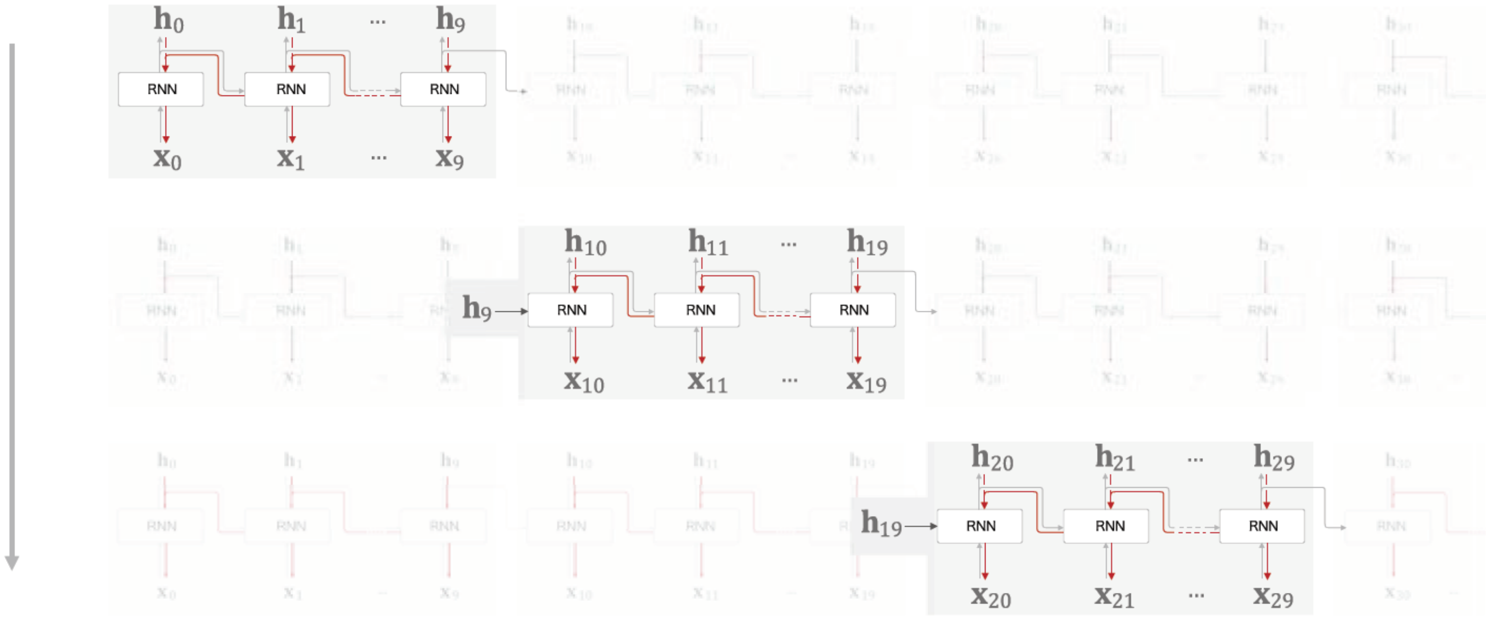
Training data $(N \times L \times I)$

Batch $(B \times L \times I)$



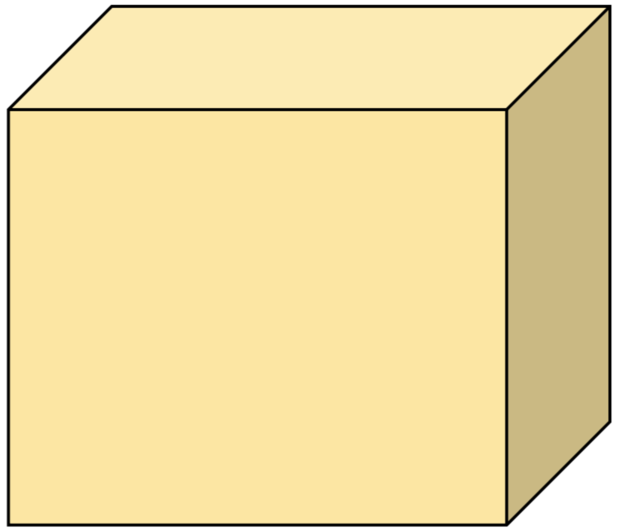
Truncated BPTT



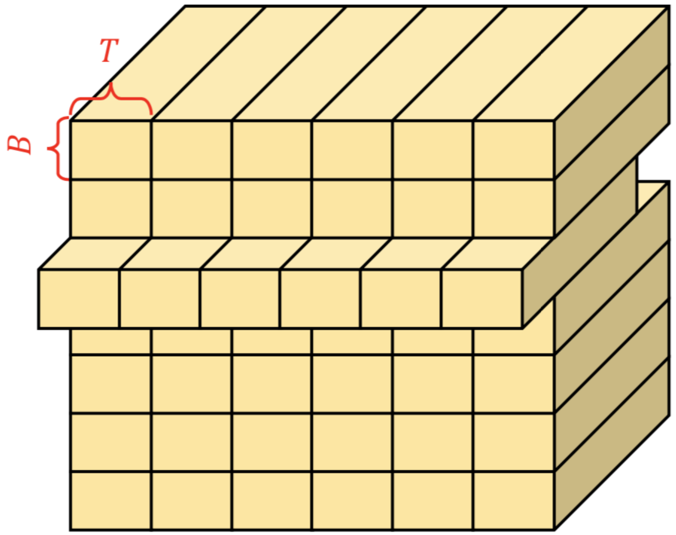


Truncated BPTT

"Truncates the length of sequential data to a constant T length to reduce the size of the computation at once."



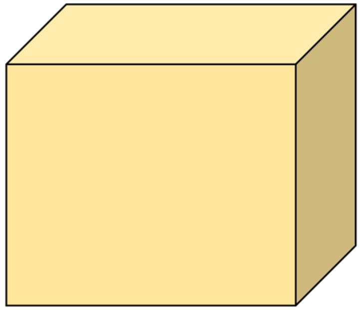
Training data $(N \times L \times I)$



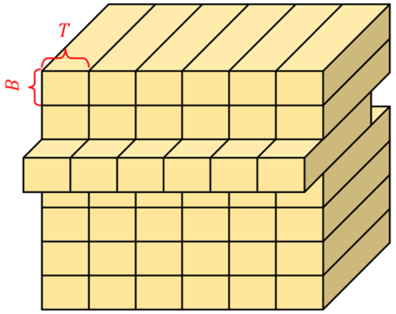
Truncated batch $(B \times T \times I)$

Truncated BPTT

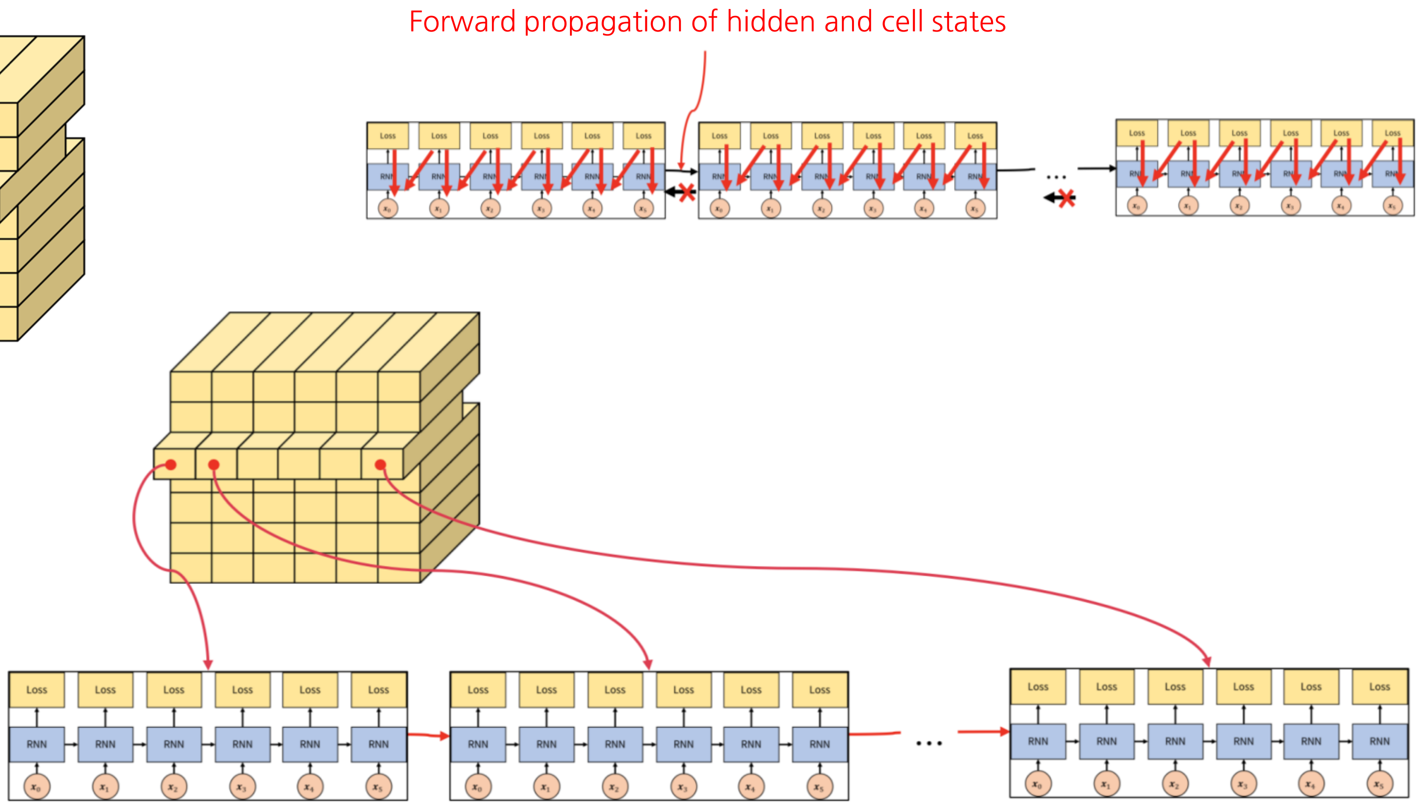
“Truncates the length of sequential data to a constant T length to reduce the size of the computation at once.”



Training data $(N \times L \times I)$



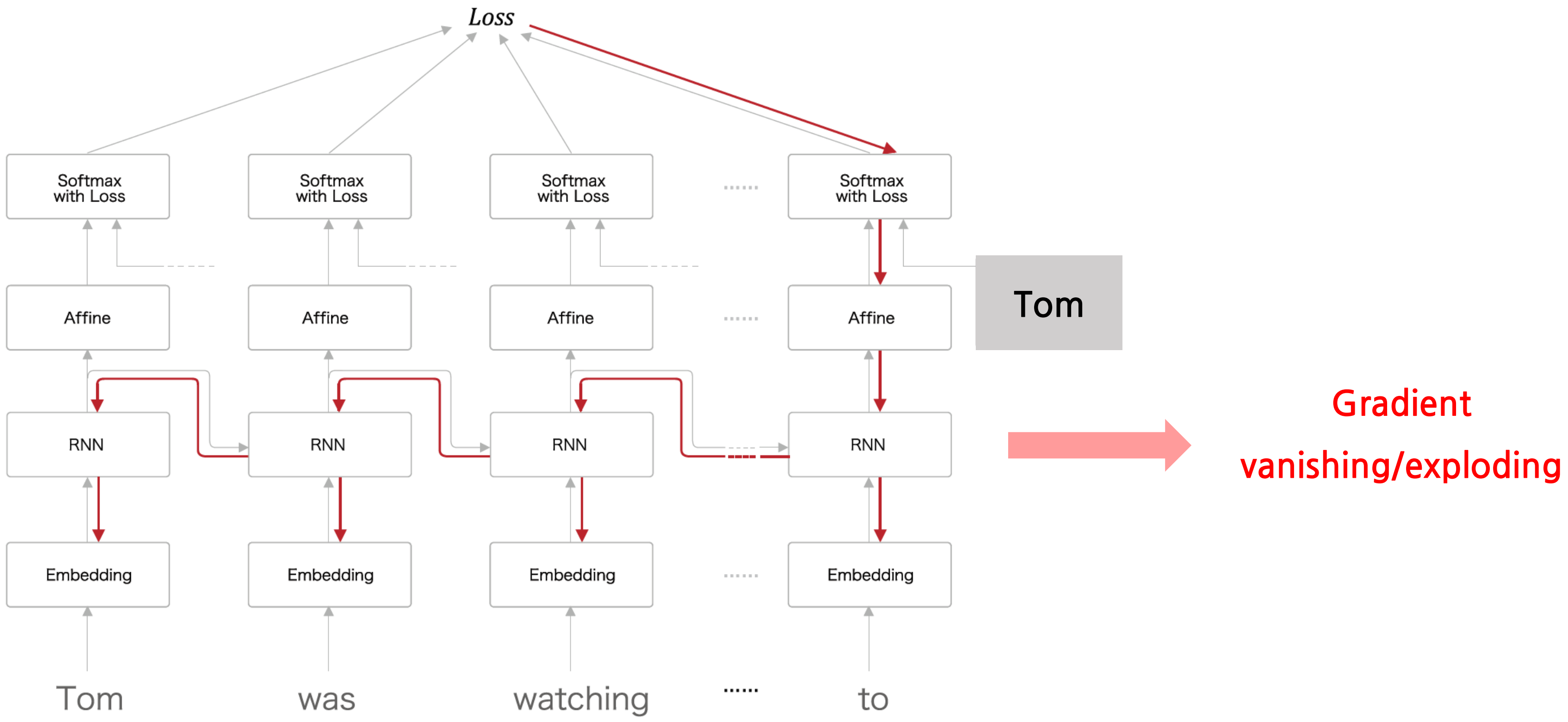
Truncated batch $(B \times T \times I)$



Limitation of Vanilla RNNs

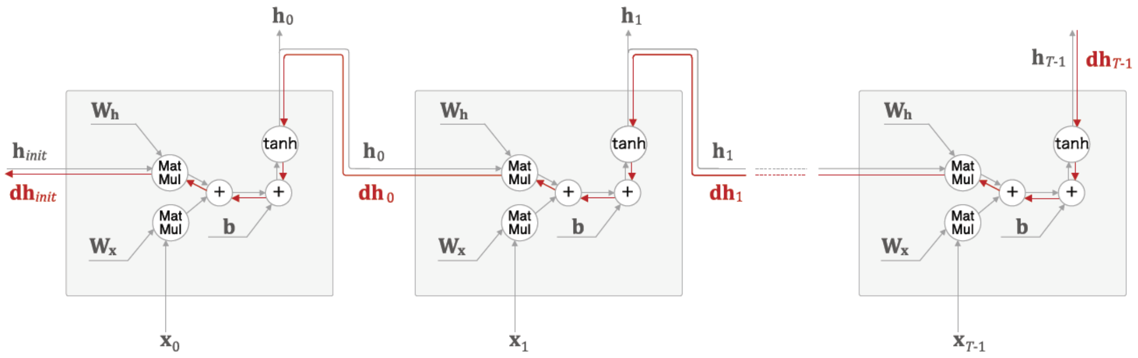
"Vanilla RNNs are limited in their ability to learn long-term memory due to computational cost and gradient issues. "

"Tom was watching TV in his room. Mary came into the room. Mary said hi to [?]."

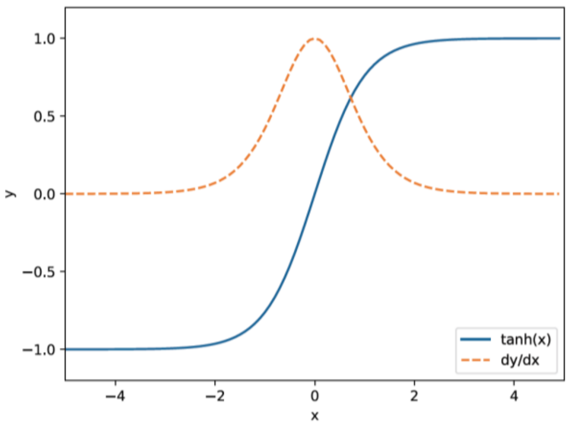


Causes of Gradient Vanishing and Exploding

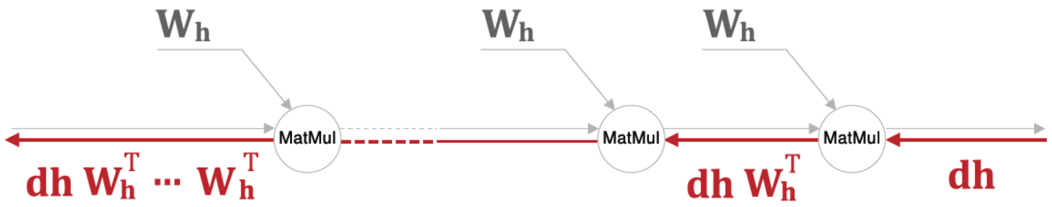
"The structure of RNN inherently causes gradient problems."



Source ① *tanh*



Source ② ***MatMul***

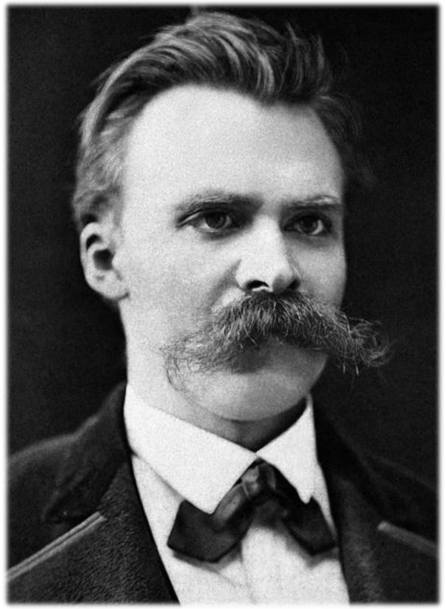


How can we memorize the long-term dependency?

LSTM (Long Short-Term Memory)

Introduction to LSTM

“How can we memorize the long-term dependency?”



Friedrich Nietzsche

“Blessed are the forgetful

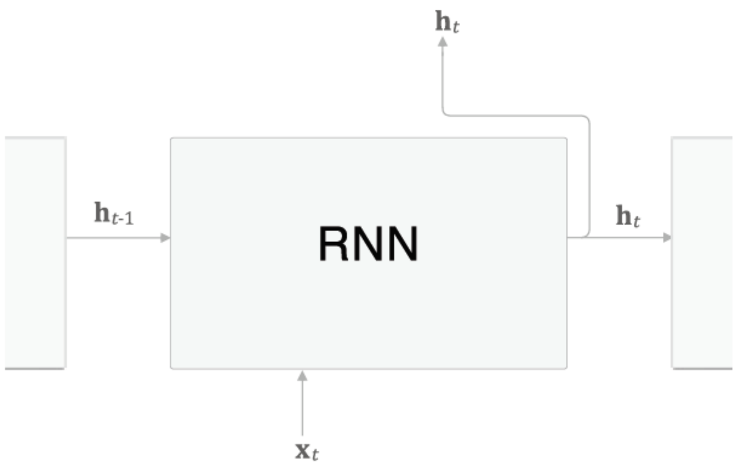
for they get the better even of their blunders.”

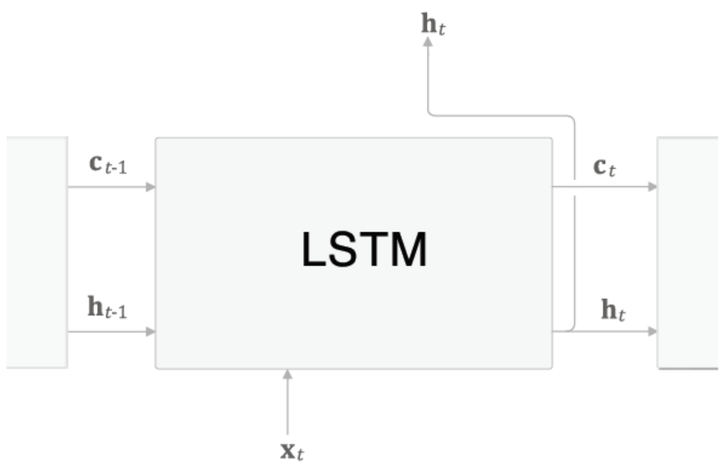
망각하는 자는 복이 있나니,

자신의 실수조차 잊기 때문이라.

Introduction to LSTM

"LSTM (Long Short-Term Memory) handles long-term dependencies in sequential data using the memory cell and gates."

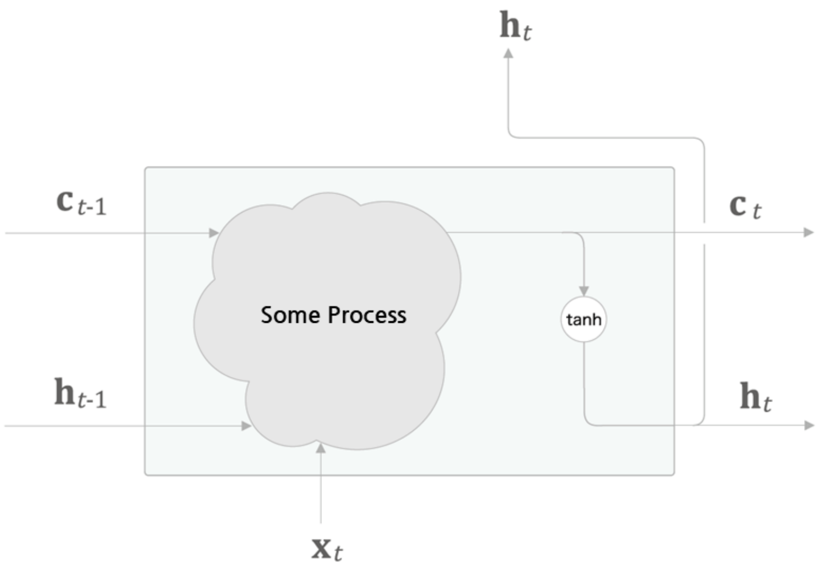




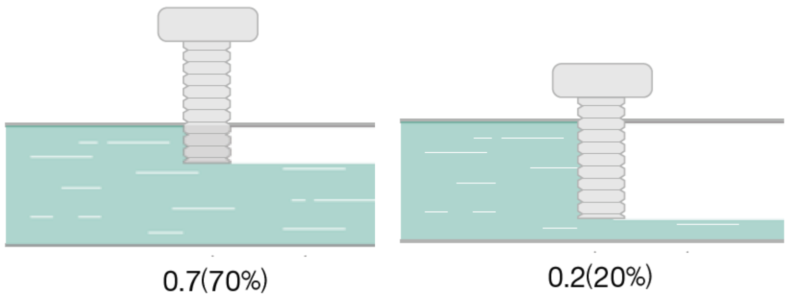
- $\boldsymbol{c}_t$: the (memory) *cell* state at time t.

Introduction to LSTM

"LSTM (Long Short-Term Memory) handles long-term dependencies in sequential data using the memory cell and gates."



An LSTM unit (layer)

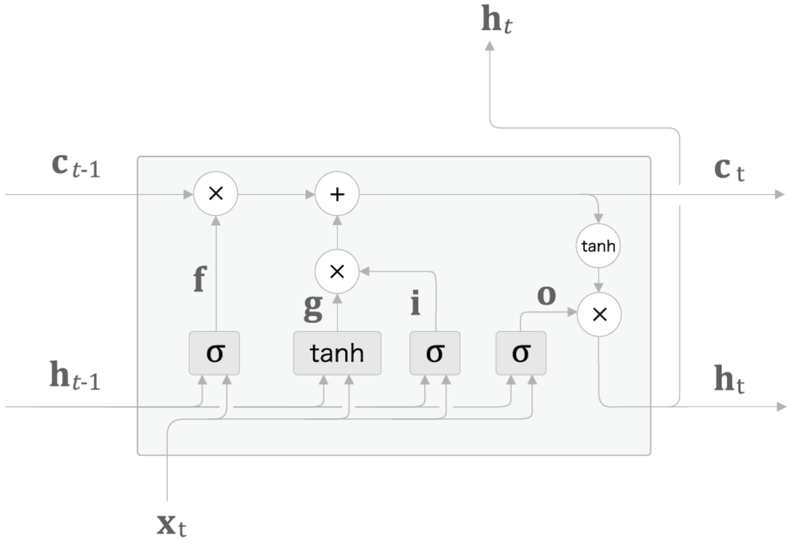


"A gate controls the amount of flow."

Gate

LSTM at a Glance

"An LSTM unit consists of a memory cell, gates, and hidden state."



States

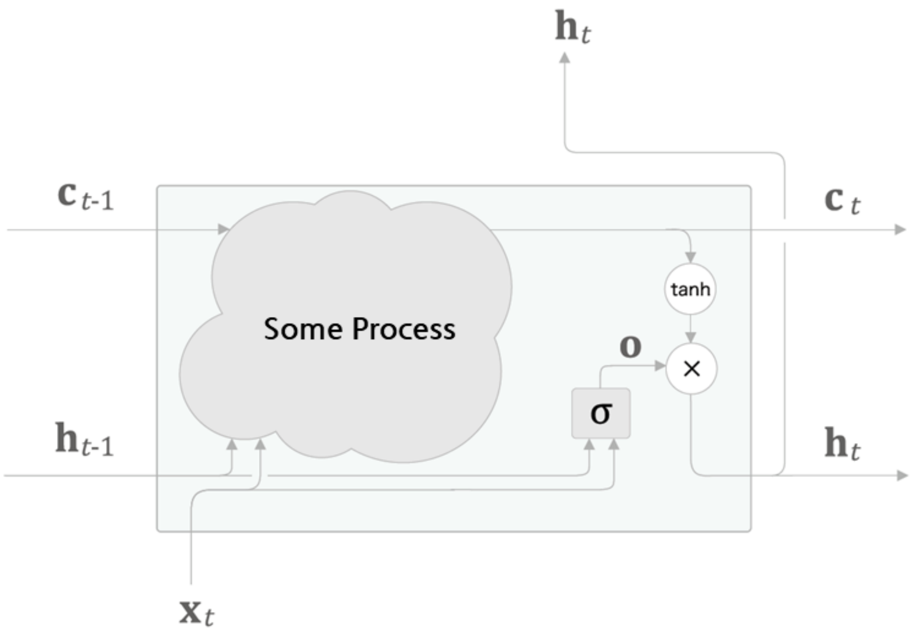
- (Memory) cell state ($\boldsymbol{c}$): storing information over long periods.
- Hidden state ($\boldsymbol{h}$): acts as short-term memory, similar to vanilla RNNs.

Gates

- Output gate ($\boldsymbol{o}$): filters the updated cell state to produce the next hidden state.
- Forget gate ($\boldsymbol{f}$): determines what information to discard from the previous state.
- Input gate ($\boldsymbol{i}$): decides which new information ($\boldsymbol{g}$) should be added to the cell state.

Output Gate

"An output gate filters the updated cell state to produce the next hidden state."



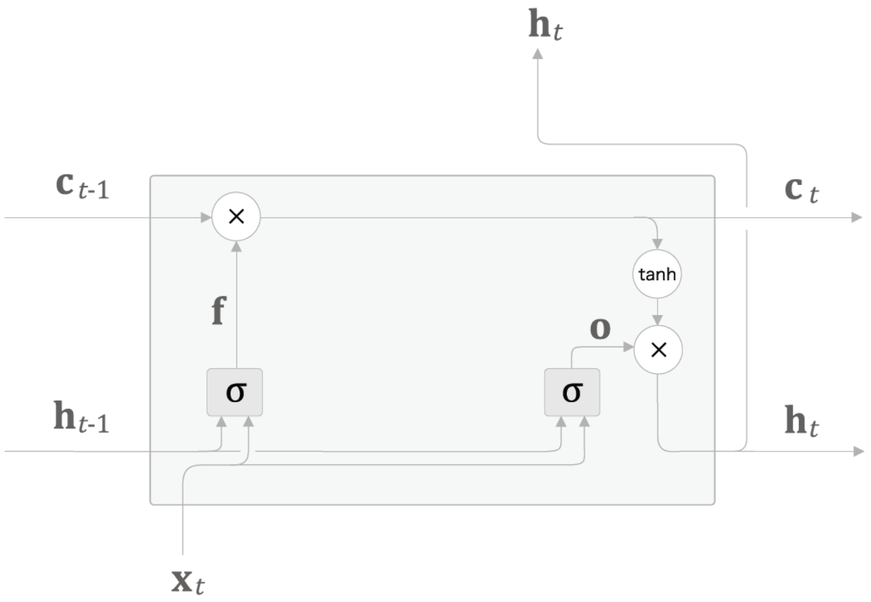
- $\boldsymbol{o} = \sigma(\boldsymbol{x}_t \boldsymbol{W}_x^{(o)} + \boldsymbol{h}_{t-1} \boldsymbol{W}_h^{(o)} + \boldsymbol{b}^{(o)})$
- $\boldsymbol{h}_t = \boldsymbol{o} \odot \tanh(\boldsymbol{c}_t)$

  where $\odot$ is elementwise product

  (a.k.a., Hadamard product).

Forget Gate

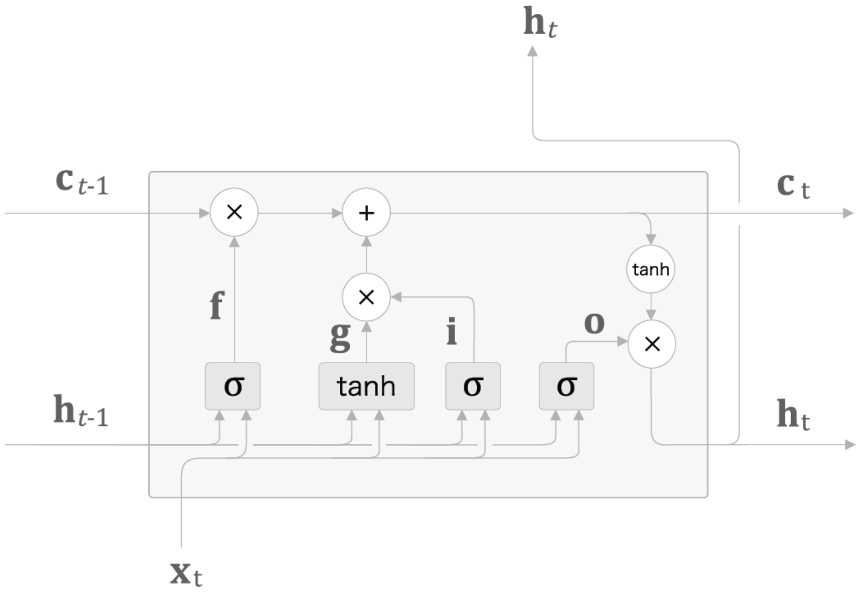
"A forget gate determines what information to discard from the previous state."



- $\boldsymbol{f} = \sigma(\boldsymbol{x}_t \boldsymbol{W}_x^{(f)} + \boldsymbol{h}_{t-1} \boldsymbol{W}_h^{(f)} + \boldsymbol{b}^{(f)})$
- $\boldsymbol{c}_t = \boldsymbol{f} \odot \boldsymbol{c}_{t-1}$

Input Gate

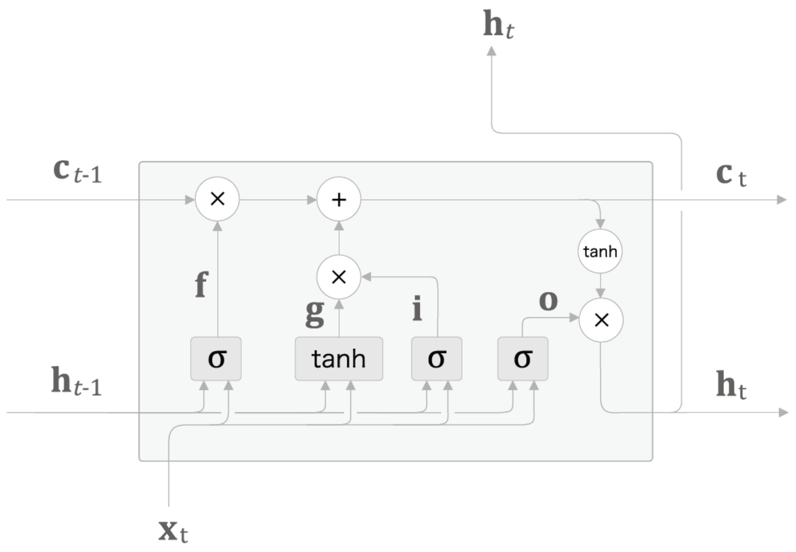
"An input gate decides which new information (***g***) should be added to the cell state."



- $\boldsymbol{g} = \tanh(\boldsymbol{x}_t \boldsymbol{W}_x^{(g)} + \boldsymbol{h}_{t-1} \boldsymbol{W}_h^{(g)} + \boldsymbol{b}^{(g)})$
- $\boldsymbol{i} = \sigma(\boldsymbol{x}_t \boldsymbol{W}_x^{(i)} + \boldsymbol{h}_{t-1} \boldsymbol{W}_h^{(i)} + \boldsymbol{b}^{(i)})$
- $\boldsymbol{c}_t = \boldsymbol{f} \odot \boldsymbol{c}_{t-1} + \boldsymbol{g} \odot \boldsymbol{i}$

Recap: LSTM at a Glance

"An LSTM unit consists of a memory cell, gates, and hidden state."



- $\boldsymbol{f} = \sigma(\boldsymbol{x}_t \boldsymbol{W}_x^{(f)} + \boldsymbol{h}_{t-1} \boldsymbol{W}_h^{(f)} + \boldsymbol{b}^{(f)})$
- $\boldsymbol{g} = \tanh(\boldsymbol{x}_t \boldsymbol{W}_x^{(g)} + \boldsymbol{h}_{t-1} \boldsymbol{W}_h^{(g)} + \boldsymbol{b}^{(g)})$
- $\boldsymbol{i} = \sigma(\boldsymbol{x}_t \boldsymbol{W}_x^{(i)} + \boldsymbol{h}_{t-1} \boldsymbol{W}_h^{(i)} + \boldsymbol{b}^{(i)})$
- $\boldsymbol{o} = \sigma(\boldsymbol{x}_t \boldsymbol{W}_x^{(o)} + \boldsymbol{h}_{t-1} \boldsymbol{W}_h^{(o)} + \boldsymbol{b}^{(o)})$
- $\boldsymbol{c}_t = \boldsymbol{f} \odot \boldsymbol{c}_{t-1} + \boldsymbol{g} \odot \boldsymbol{i}$
- $\boldsymbol{h}_t = \boldsymbol{o} \odot \tanh(\boldsymbol{c}_t)$

$tanh$ and $Sigmoid$ in LSTM

- **Why do we use $tanh$ in LSTM?**
  - Bounded output range
    - $tanh$ squishes values between -1 and 1, which helps prevent exploding gradients during training.
    - $tanh$ provides the normalization effect during the recursive processing of past values.
  - Zero-centered output

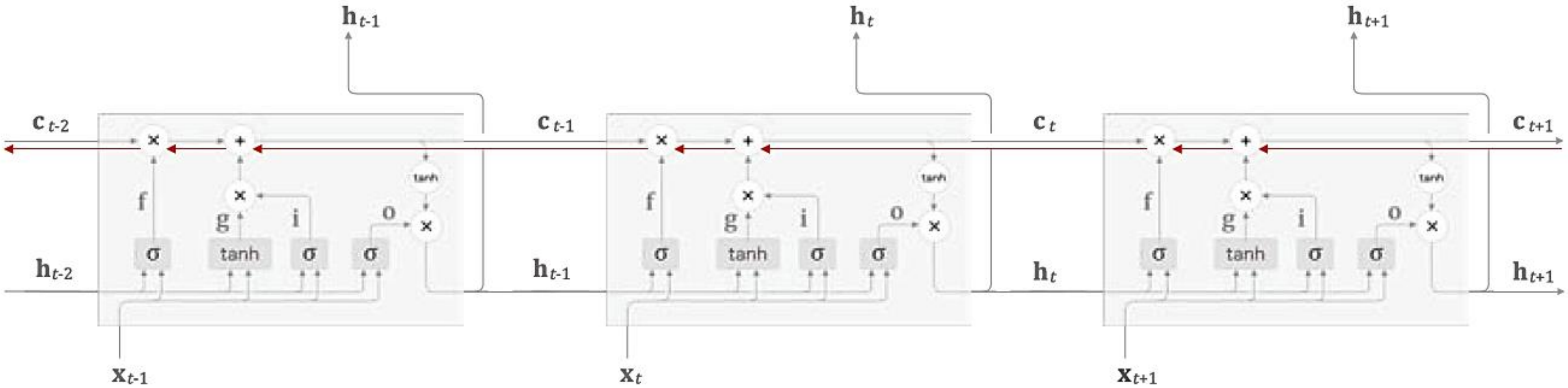
    Unlike *sigmoid*, which outputs between 0 and 1, *tanh* can output both positive and negative values that allow the hidden state values to increase or decrease, providing more flexibility in modeling sequential data.

- **Why do we use $sigmoid$ in LSTM?**
  - Gating mechanism

    Its output range for 0 to 1 makes it ideal for the gates, as it allows them to control the flow of information.

The Gradient Flow in LSTM

"LSTM addresses the gradient problems by the backpropagation through the memory cells."



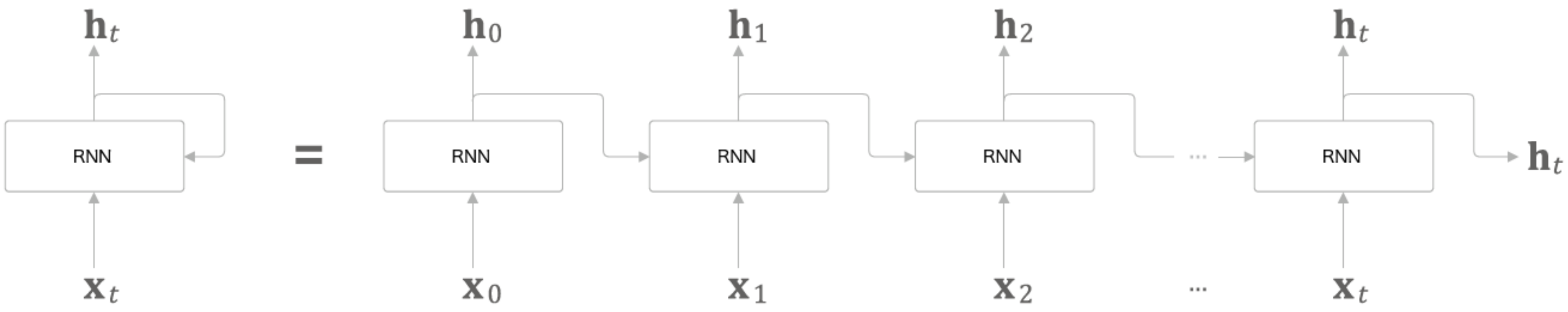
"Only + and × operations on the path.":

- + : no change in gradients.
- × : independent elementwise products instead of recursive matrix multiplications.

Takeaways

Recurrent Neural Network (RNN)

"A recurrent neural network recurrently applies a sequence processor using *hidden states.*"

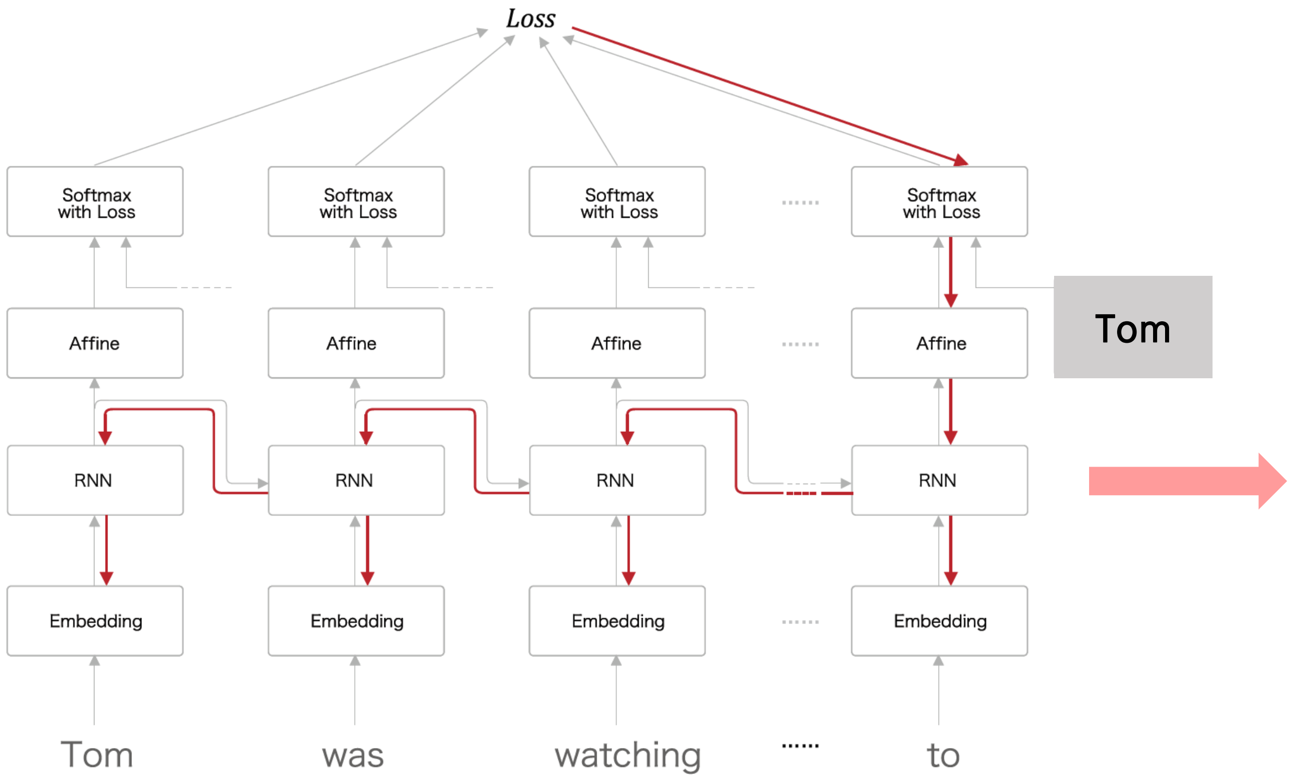


- $\boldsymbol{h}_t = \tanh(\boldsymbol{h}_{t-1}^T \boldsymbol{W}_{hh} + \boldsymbol{x}_t \boldsymbol{W}_{xh} + \boldsymbol{b}_h)$
- $\boldsymbol{y}_t = \tanh(\boldsymbol{h}_t \boldsymbol{W}_{hy} + \boldsymbol{b}_y)$

Limitation of Vanilla RNNs

"Vanilla RNNs are limited in their ability to learn long-term memory due to computational cost and gradient issues. "

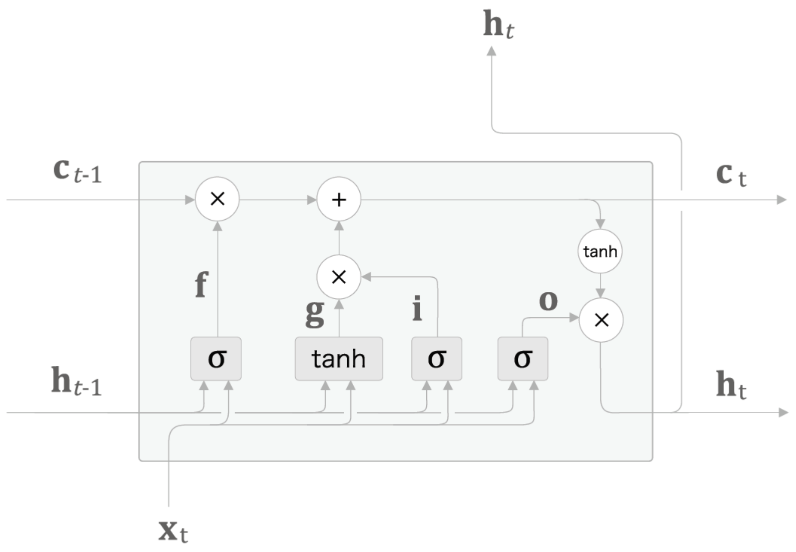
"Tom was watching TV in his room. Mary came into the room. Mary said hi to [?]."



Gradient vanishing/exploding

Recap: LSTM at a Glance

"An LSTM unit consists of a memory cell, gates, and hidden state."



- $\boldsymbol{f} = \sigma(\boldsymbol{x}_t \boldsymbol{W}_x^{(f)} + \boldsymbol{h}_{t-1} \boldsymbol{W}_h^{(f)} + \boldsymbol{b}^{(f)})$
- $\boldsymbol{g} = \tanh(\boldsymbol{x}_t \boldsymbol{W}_x^{(g)} + \boldsymbol{h}_{t-1} \boldsymbol{W}_h^{(g)} + \boldsymbol{b}^{(g)})$
- $\boldsymbol{i} = \sigma(\boldsymbol{x}_t \boldsymbol{W}_x^{(i)} + \boldsymbol{h}_{t-1} \boldsymbol{W}_h^{(i)} + \boldsymbol{b}^{(i)})$
- $\boldsymbol{o} = \sigma(\boldsymbol{x}_t \boldsymbol{W}_x^{(o)} + \boldsymbol{h}_{t-1} \boldsymbol{W}_h^{(o)} + \boldsymbol{b}^{(o)})$
- $\boldsymbol{c}_t = \boldsymbol{f} \odot \boldsymbol{c}_{t-1} + \boldsymbol{g} \odot \boldsymbol{i}$
- $\boldsymbol{h}_t = \boldsymbol{o} \odot \tanh(\boldsymbol{c}_t)$

Thank you! 😀