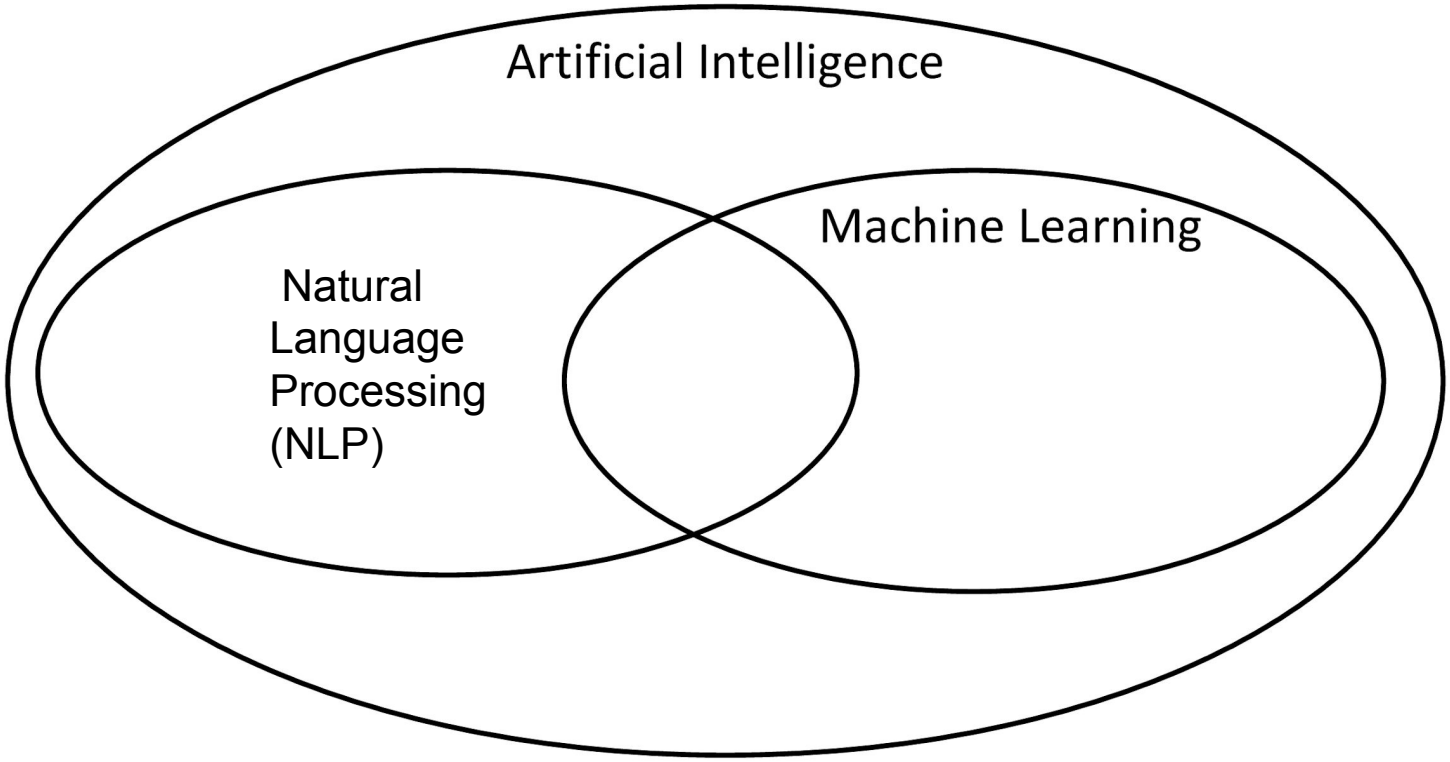
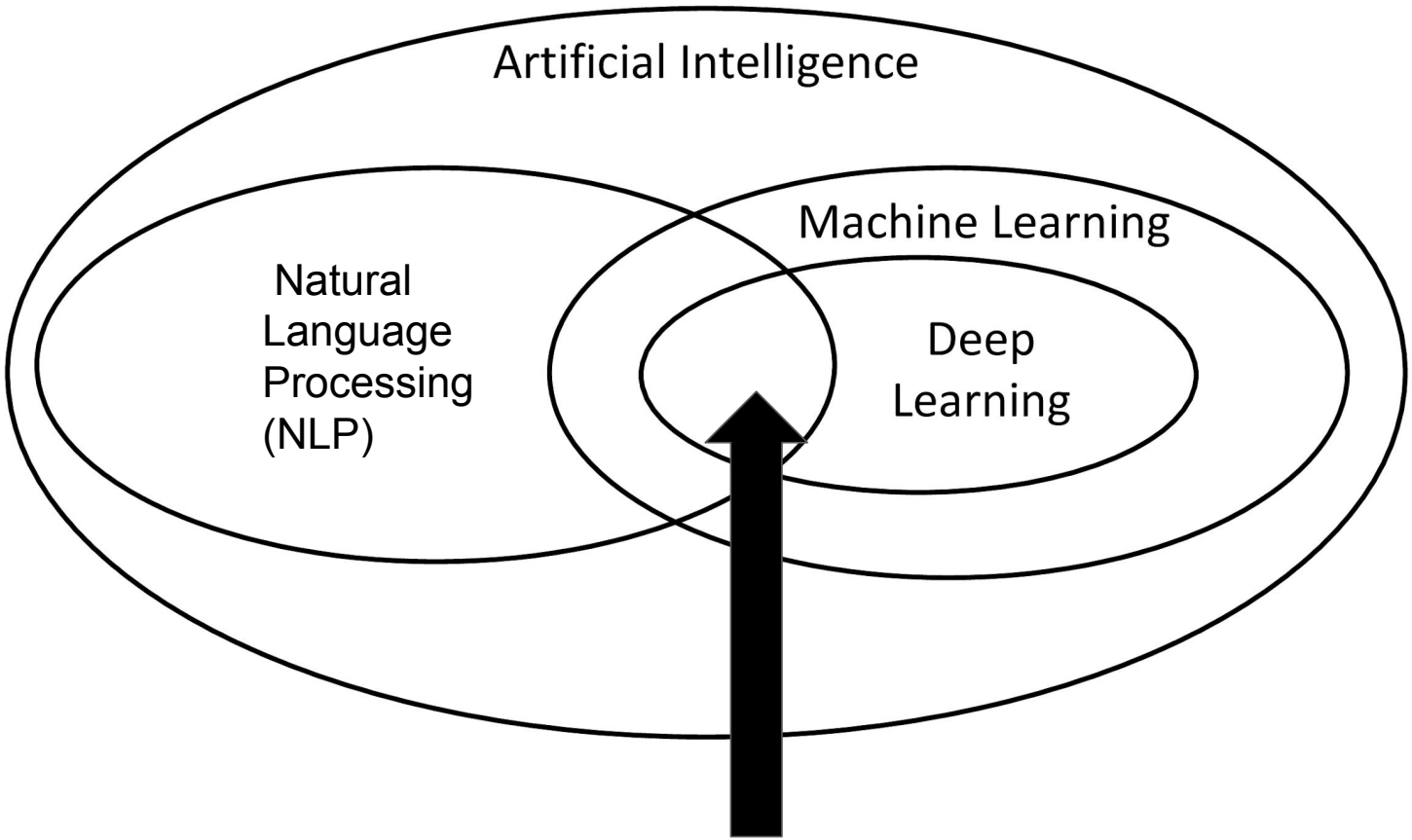
Deep Learning for Natural Language Processing (NLP)

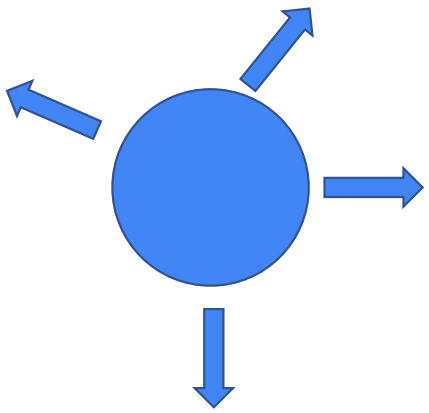

Artificial Intelligence
Machine Learning
Natural
Language
Processing
(NLP)



Deep Learning for Natural Language Processing (NLP)

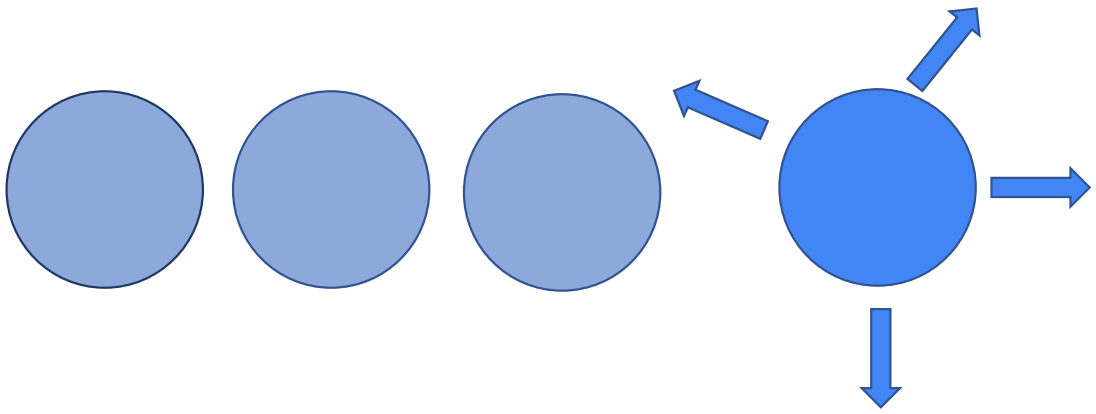
Sequence Modelling

Sequence modeling

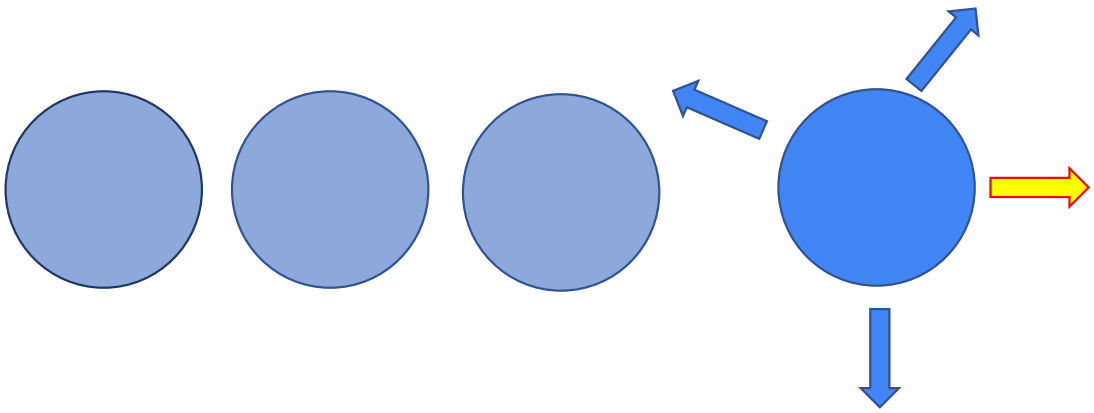


?

Sequence modeling



Sequence modeling



Sequences are Everywhere

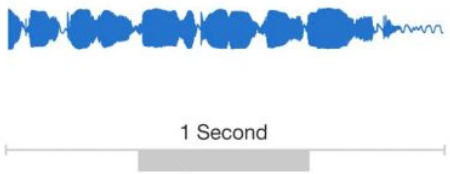
"Sequences really seem to be everywhere! We should learn how to model them. What is the best way to do that? Stay tuned!"

Words, letters

Sequences are Everywhere

"Sequences really seem to be everywhere! We should learn how to model them. What is the best way to do that? Stay tuned!"

Words, letters

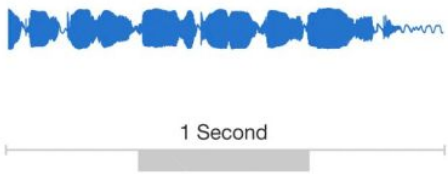


Speech

Sequences are Everywhere

"Sequences really seem to be everywhere! We should learn how to model them. What is the best way to do that? Stay tuned!"

Words, letters



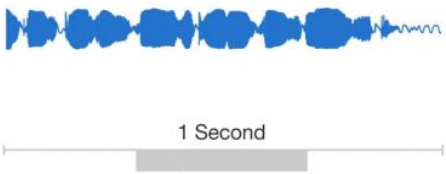
Speech

Videos

Sequences are Everywhere

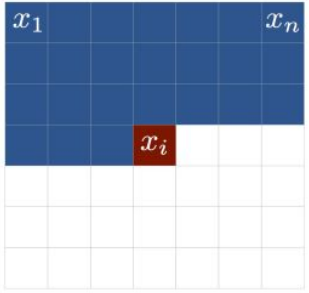
"Sequences really seem to be everywhere! We should learn how to model them. What is the best way to do that? Stay tuned!"

Words, letters



Speech

Videos

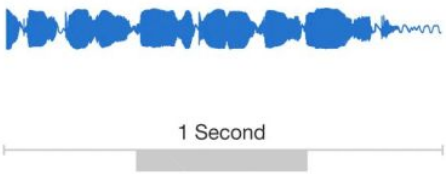


Images

Sequences are Everywhere

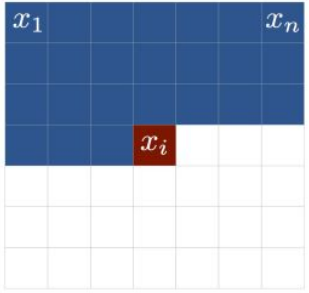
"Sequences really seem to be everywhere! We should learn how to model them. What is the best way to do that? Stay tuned!"

Words, letters



Speech

Videos



Images

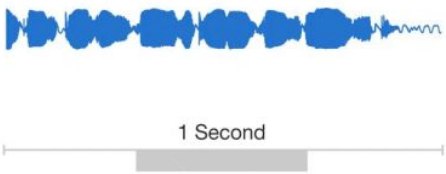
```python
import numpy as np
import matplotlib.pyplot as plt

def forward_backward_prop(w, T):
  hs = [0.5]
  for _ in range(T):
    hs.append(np.tanh(w*hs[-1]))

  dh = 1
  for t in range(T):
    dh = (1-hs[-1-t] ** 2) * w * dh

  return hs[-1], dh

T = 10  # sequence length
wlim = 4  #limit of interval over weights w

results = []
ws = np.linspace(-wlim, wlim, 1000)
for w in ws:
  results.append(forward_backward_prop(w, T))

plt.plot(ws, [r[0] for r in results], label='RNN state')
plt.plot(ws, [r[1] for r in results], label='Gradients')
```

Programs

Sequences are Everywhere

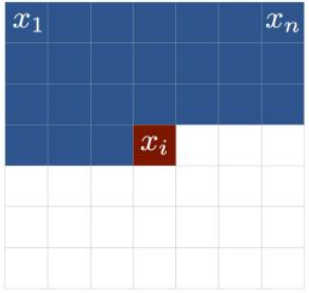
"Sequences really seem to be everywhere! We should learn how to model them. What is the best way to do that? Stay tuned!"

Words, letters



Speech

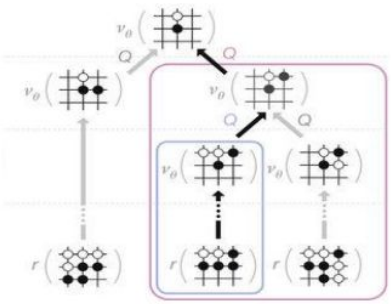
Videos



Images

```
import numpy as np
import matplotlib.pyplot as plt

def forward_backward_prop(w, T):
  hs = [0.5]
  for _ in range(T):
    hs.append(np.tanh(w*hs[-1]))

  dh = 1
  for t in range(T):
    dh = (1-hs[-1-t] ** 2) * w * dh

  return hs[-1], dh

T = 10  # sequence length
wlim = 4  #limit of interval over weights w

results = []
ws = np.linspace(-wlim, wlim, 1000)
for w in ws:
  results.append(forward_backward_prop(w, T))

plt.plot(ws, [r[0] for r in results], label='RNN state')
plt.plot(ws, [r[1] for r in results], label='Gradients')
```

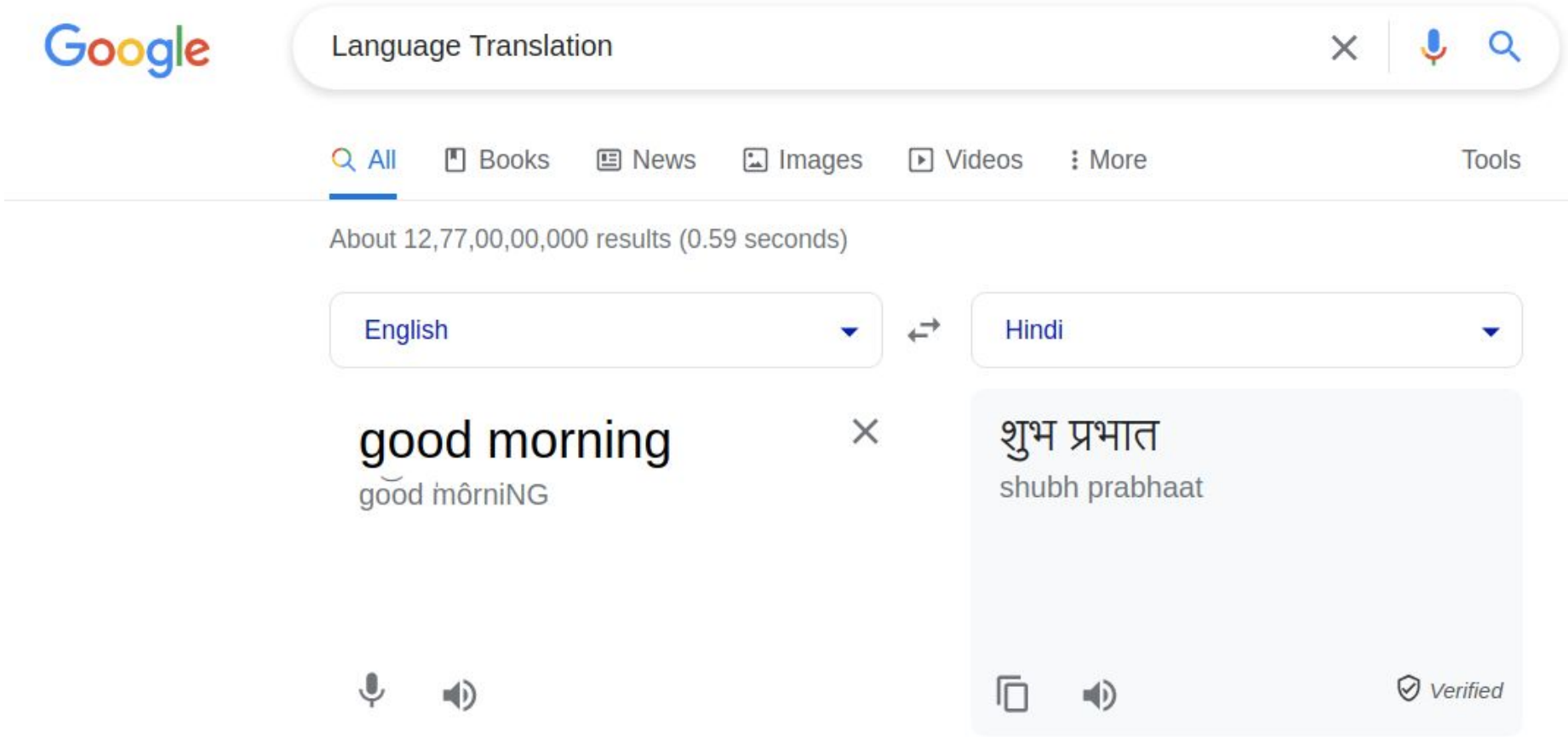
Programs



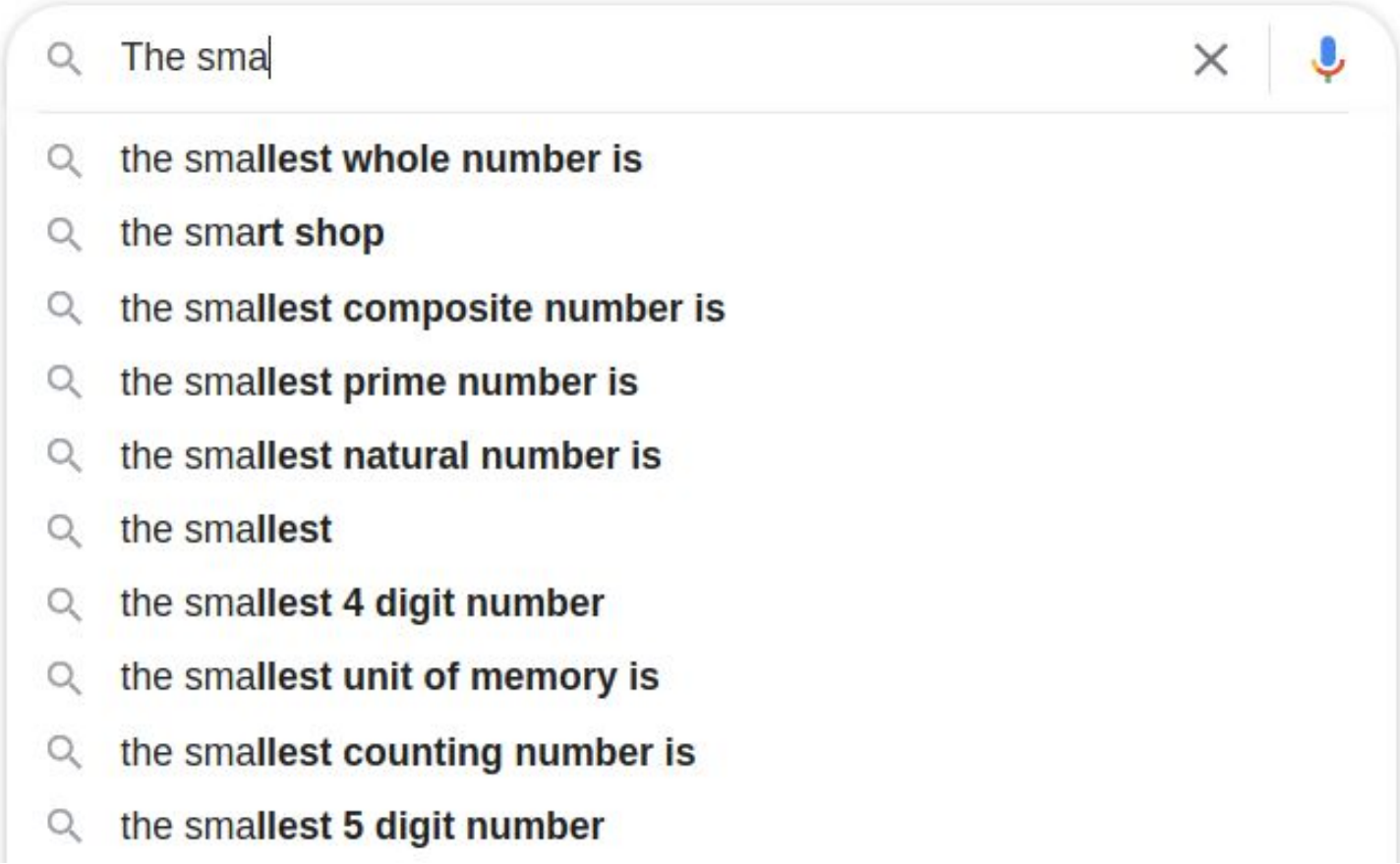
Decision making

NLP Tasks Overview

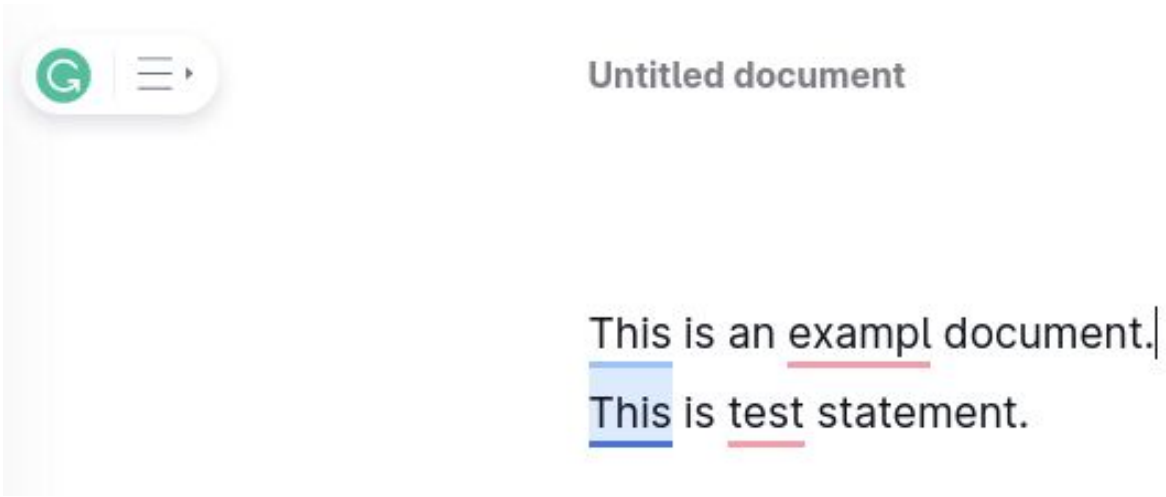
Language Translation



Query Recommendations



Spelling and Grammar Corrections



Sentiment Analysis

★★★★★ **Excellent taste**

Reviewed in India on 30 March 2020

Size: 504 g (Pack of 36) | **Verified Purchase**

I'm hooked on to Girnar teas. They are excellent and taste great. Easy to dissolve and hassle free. If you are the kind of person who wants a consistent taste for their chai, then switch to Girnar and carry the sachets wherever you go. A tad expensive though

★☆☆☆☆ **Poorest in quality**

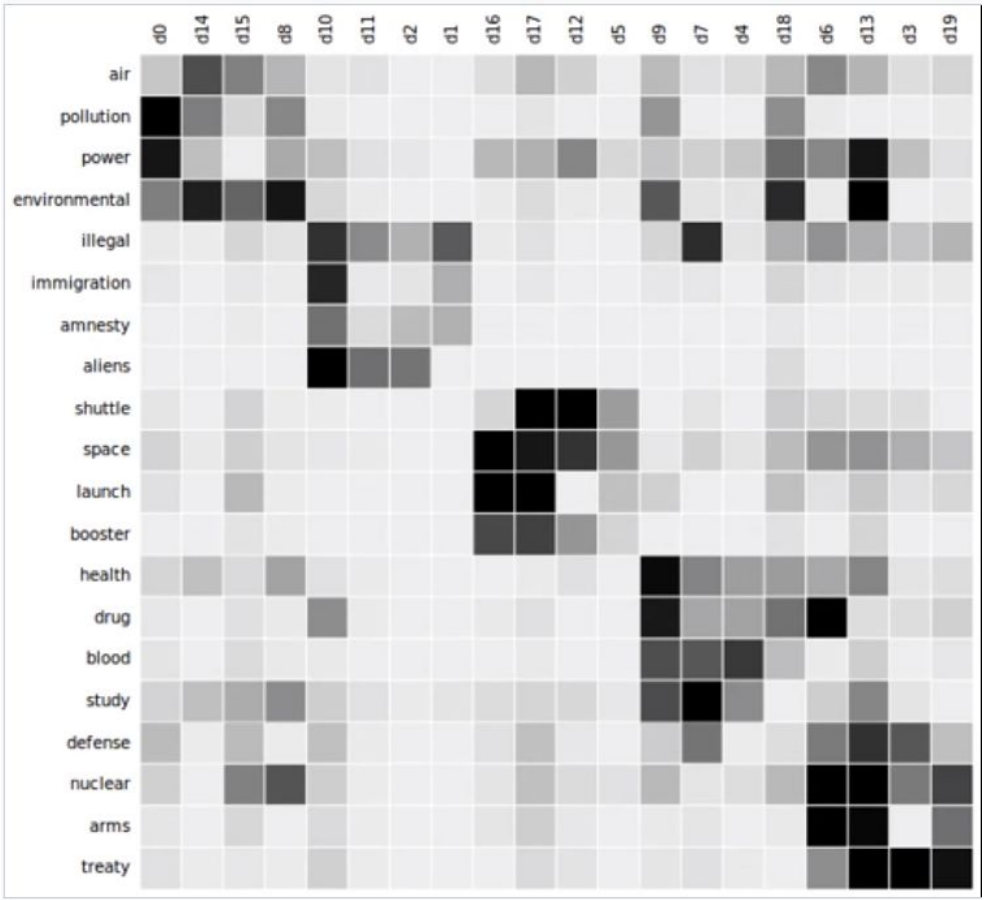
Reviewed in India on 26 December 2020

Size: 10 Sachets (Low Sugar) | **Verified Purchase**

I have been old customer of this product. But this is worst in taste. I dont know whether it is original or not? or company has deteriorated the quality badly.

One person found this helpful

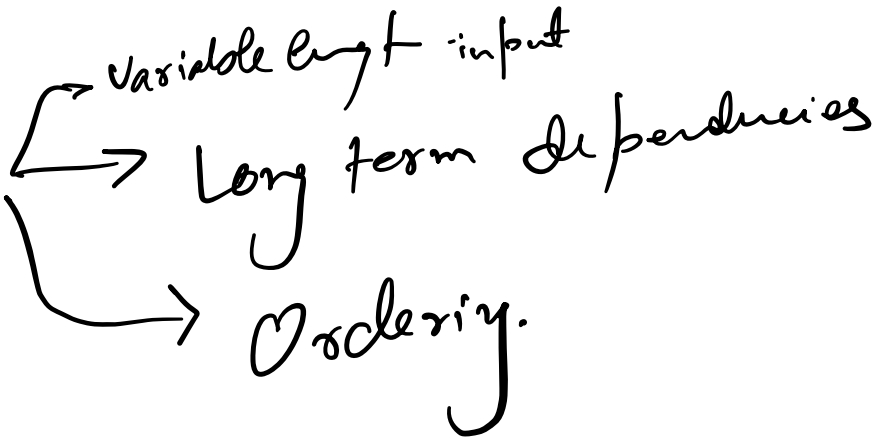
Topic modeling



Document-Word Matrix

NLP Tasks challenge

NLP Tasks challenge



- Variable length sequence
  - The food is great
  - Jaipur city is famous as a pink city

NLP Tasks challenge

- Variable length sequence

  The food is great

  Jaipur city is famous as a pink city

- Long-term dependency

  France is where he grew up, but now he is in India. He speaks fluent _____.

Punjabi

NLP Tasks challenge

- Variable length sequence

  The food is great

  Jaipur city is famous as a pink city

- Long-term dependency

  France is where he grew up, but now he is in India. He speaks fluent _____. **French**

NLP Tasks challenge

- Variable length sequence
  - The food is great
  - Jaipur city is famous as a pink city
- Long-term dependency
  - **France** is where he grew up, but now he is in India. He speaks fluent _____. **French**
- Differences in sequence order.
  - The food was good, not bad at all
  - The food was bad, not good at all.

NLP Tasks challenge

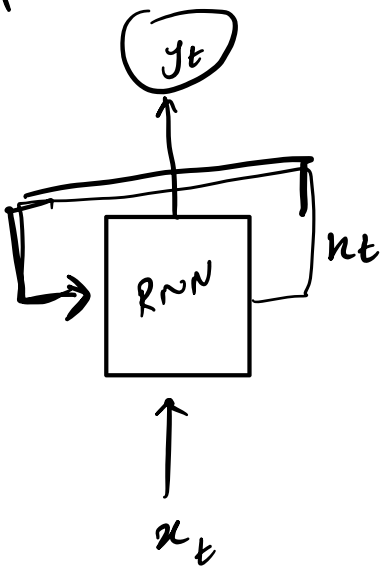
- Variable length sequence
  - The food is great
  - Jaipur city is famous as a pink city
- Long-term dependency
  - **France** is where he grew up, but now he is in India. He speaks fluent _____. **French**
- Difference in sequence order.
  - The food was good, not bad at all
  - The food was bad, not good at all.

We can model NLP Tasks as Sequence to Sequence Problem

Sequence to Sequence Learning with Neural Networks

RNN

RNN



$$h_t = \tanh\left(W_{xh} x_t + W_{hh} h_{t-1} + b_h\right)$$

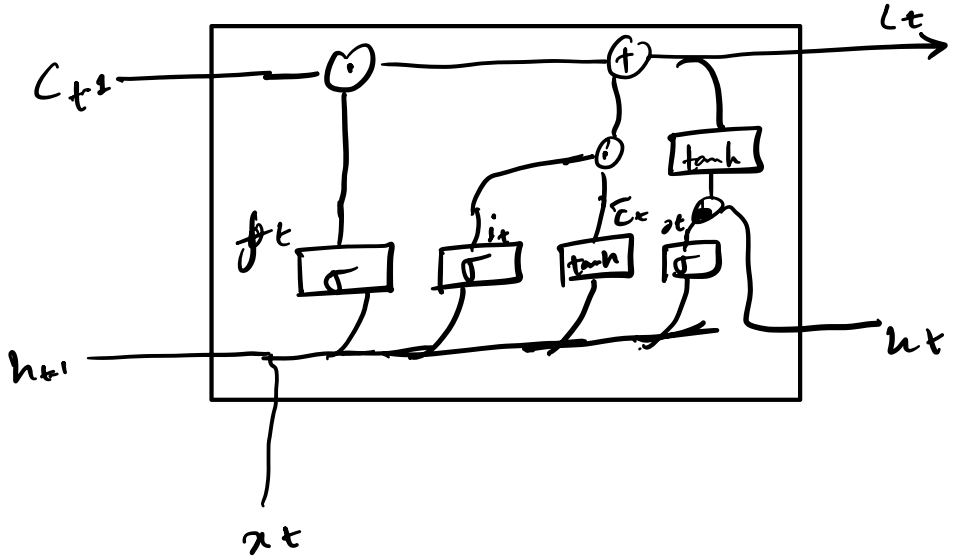
LSTM

$$\begin{pmatrix} i_t \\ o_t \\ f_t \\ \tilde{c}_t \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{pmatrix} \left(W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix} + b_h \right)$$

$$c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t$$

$$h_t = o_t \odot \tanh(c_t)$$

LSTM cell.



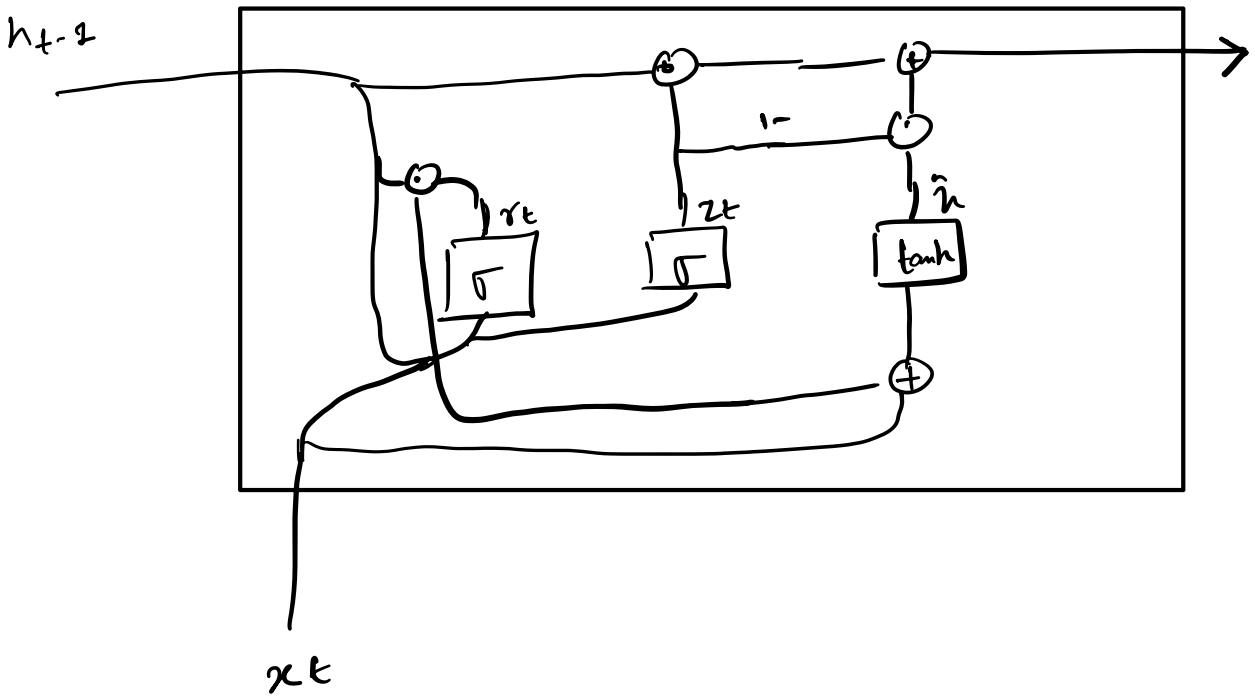
Gated Recurrent Unit

$$r_t = \sigma\left(W_{xr} x_t + W_{hr} h_{t-1}\right)$$

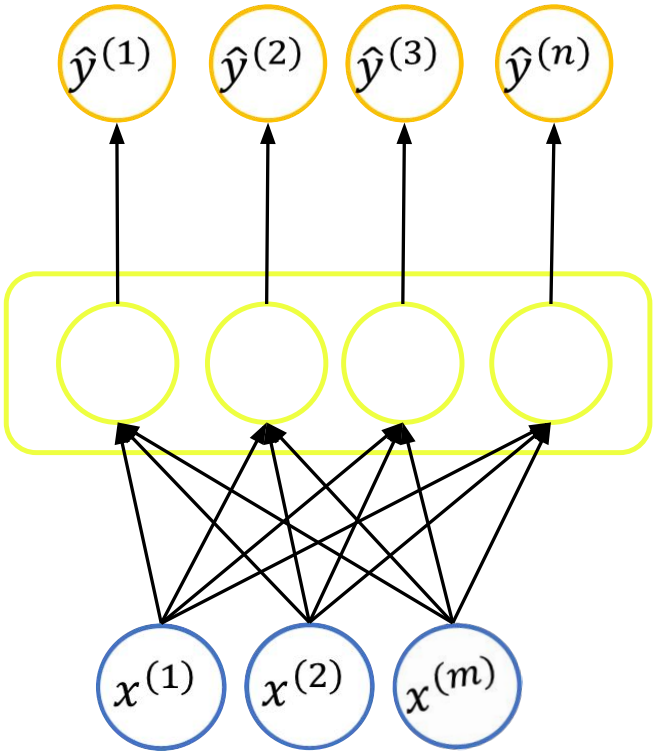
$$z_t = \sigma\left(W_{xz} x_t + W_{hz} h_{t-1}\right)$$

$$\hat{h}_t = \tanh\left(W_{xh} x_t + W_{hh}\left(r_t \odot h_{t-1}\right)\right)$$

$$h_t = z_t \odot h_{t-1} + (1 - z_t) \odot \hat{h}_t$$



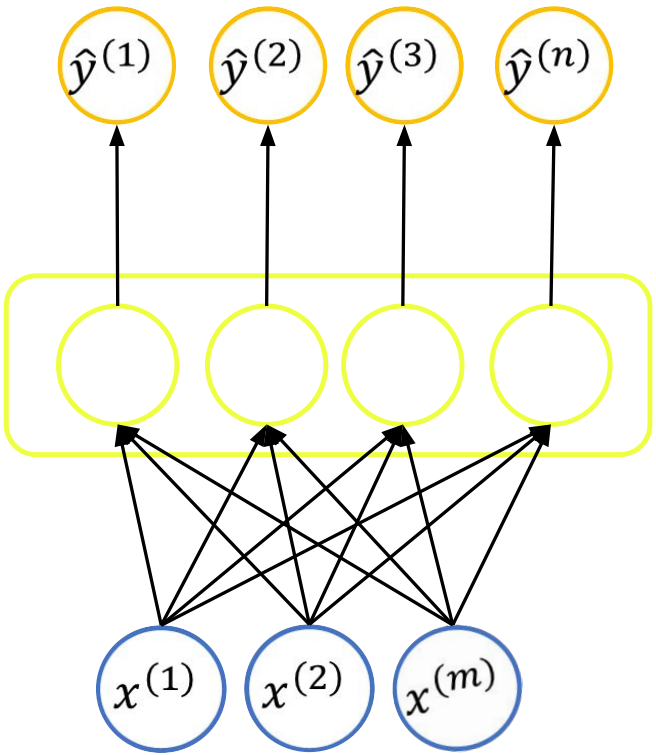
Feed-Forward Neural Network



$\hat{y} \in \mathbb{R}^n$

$x \in \mathbb{R}^m$

Feed-Forward Neural Network

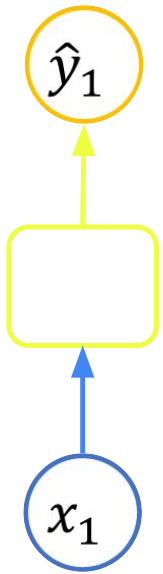


$\hat{y} \in \mathbb{R}^n$

Difficult to model speech recognition, question answering, and machine translation (Sequence to Sequence) etc.

$x \in \mathbb{R}^m$

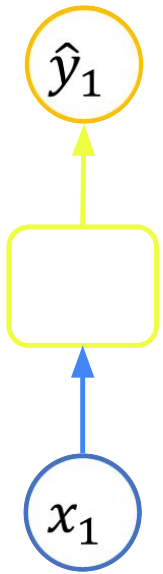
Sequence to Sequence Problem



One to One

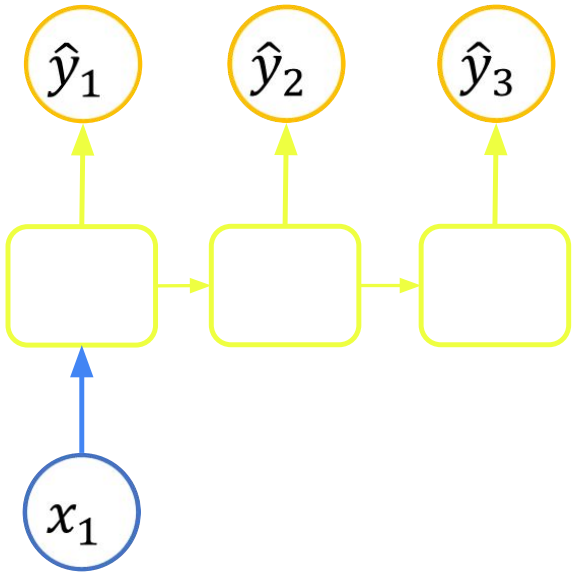
Image Classification

Sequence to Sequence Problem



One to One

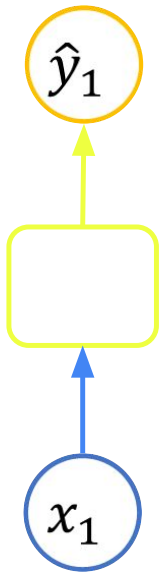
Image Classification



One to many

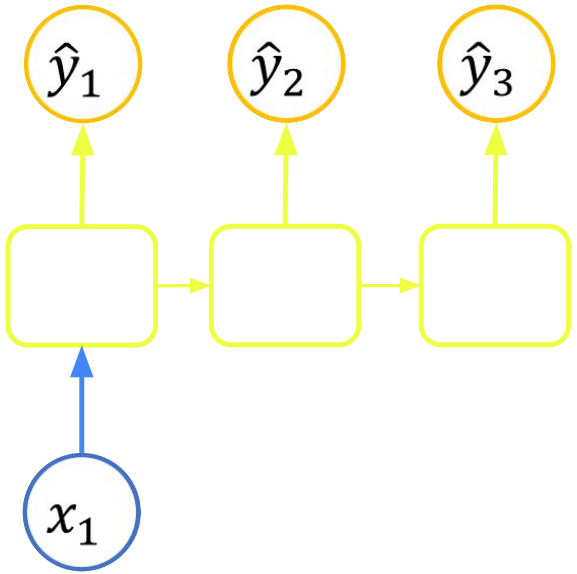
Image Captioning

Sequence to Sequence Problem



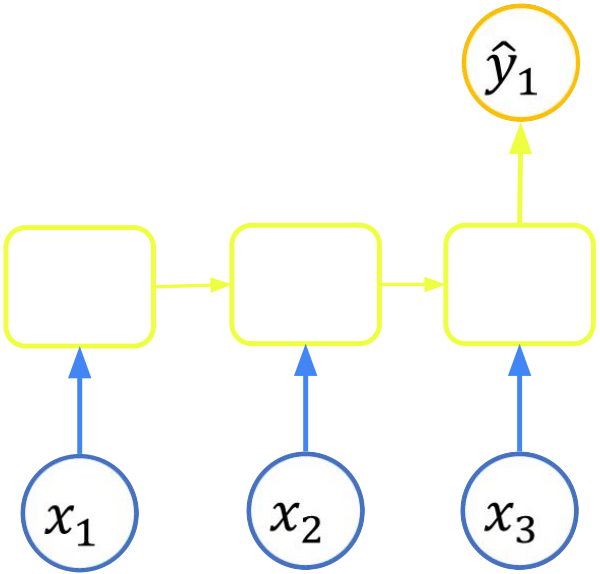
One to One

Image Classification



One to many

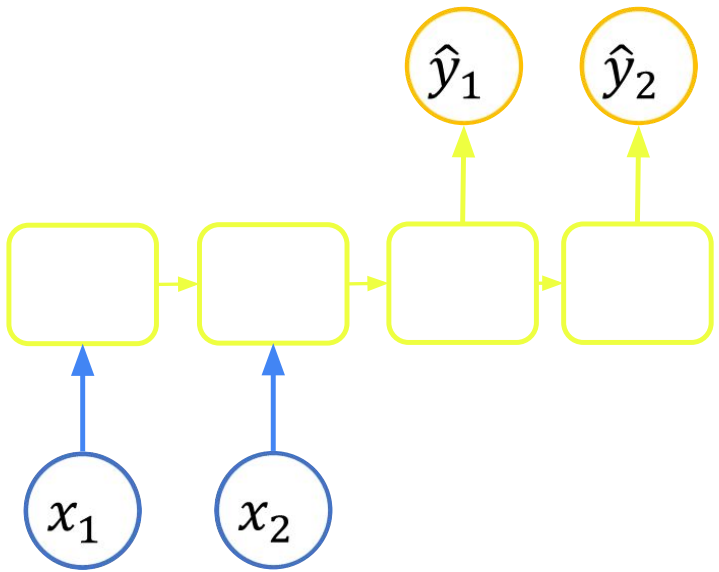
Image Captioning



many to one

Language Recognition

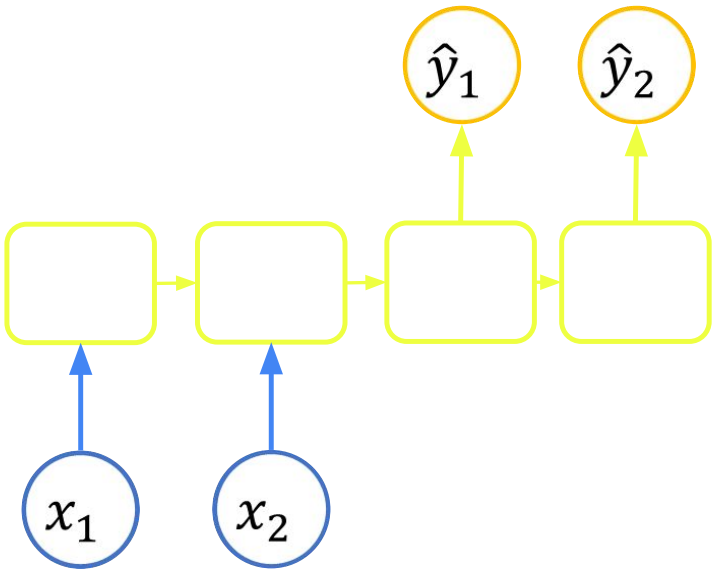
Sequence to Sequence Problem



Many to Many

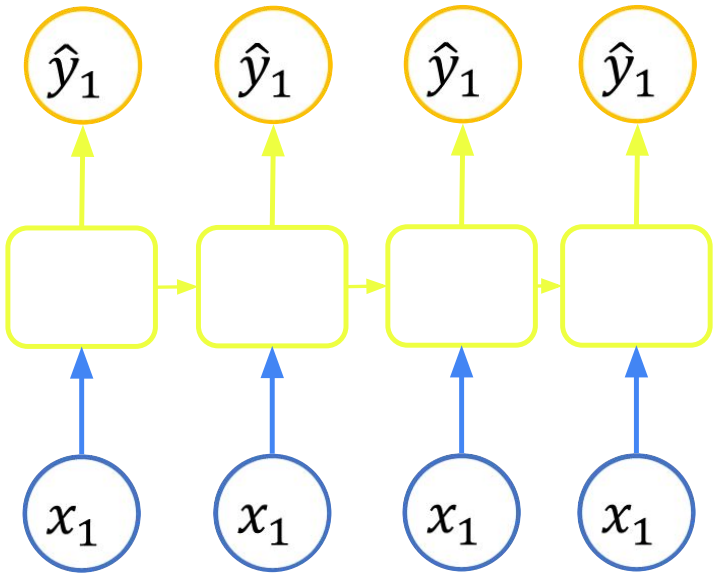
Machine Translation

Sequence to Sequence Problem



Many to Many

Machine Translation



Many to many

Video Activity Recognition

seq2seq Learning

- Seq2seq models are neural network architecture used to transform input sequences into output sequences of **variable length**.

seq2seq Learning

- Seq2seq models are neural network architecture used to transform input sequences into output sequences of **variable length**.
- Seq2seq does the sequence transformation using a simple RNN or using LSTM or GRU (to avoid problem of vanishing gradients)

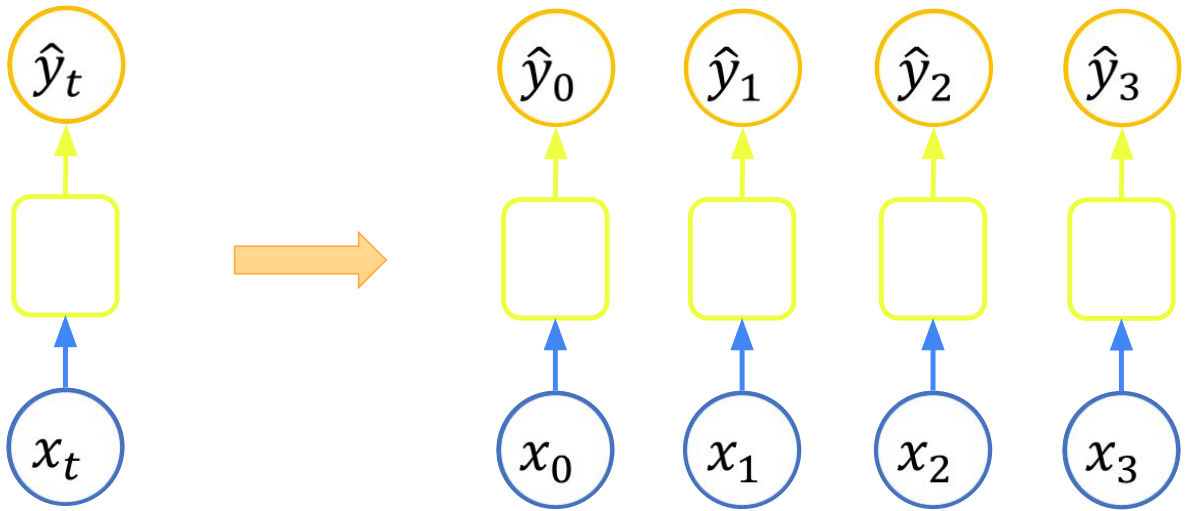
Basics of RNN

RNN

Recurrent Neural Network is a family of neural networks that:

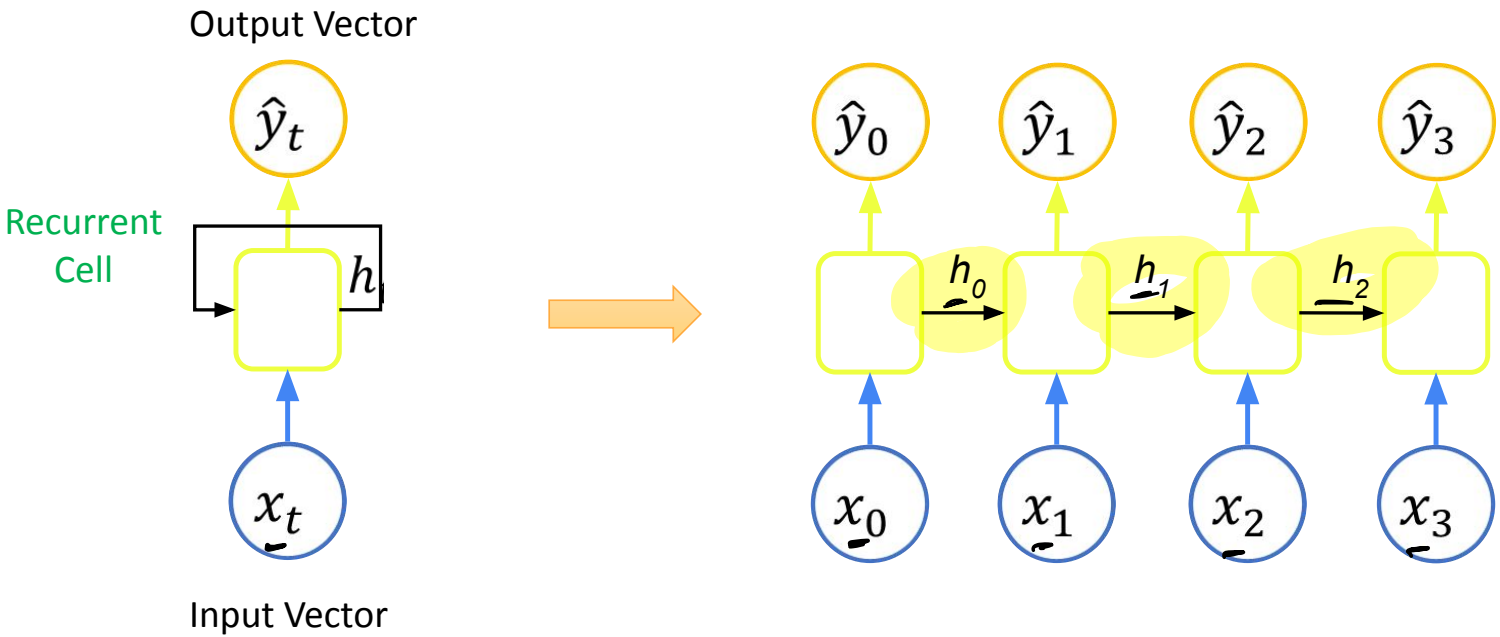
- Process sequence data
- Take sequential **input** of variable length
- Apply the same weights on each step
- Can produce **output** of variable length

Output Vector

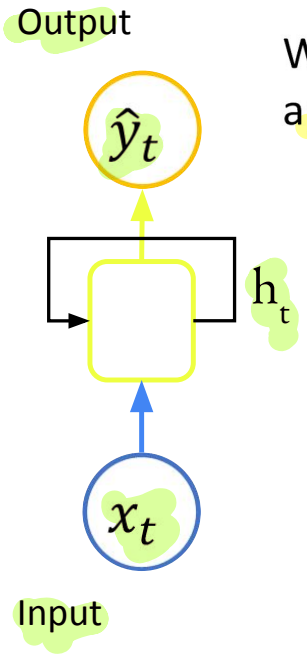


Input Vector

Recurrent Neural Networks



Recurrent Neural Networks



We can process a sequence of vectors **x** by applying a **recurrence formula** at every time step:

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

new state

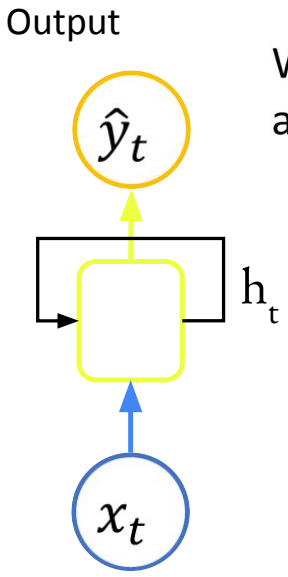
some function with parameters W

old state

input vector at some time step

Recurrent Neural Networks

Notice: the same function and the same set of parameters are used at every time step.



We can process a sequence of vectors **x** by applying a **recurrence formula** at every time step:

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

new state

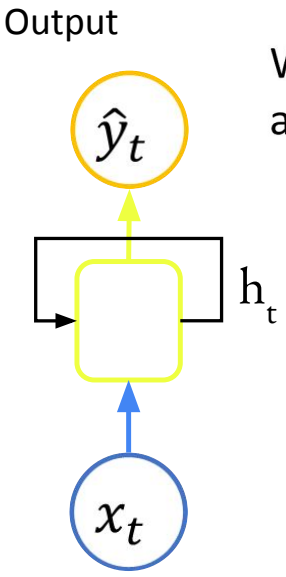
some function with parameters W

old state

input vector at some time step

Recurrent Neural Networks

Notice: the same function and the same set of parameters are used at every time step.

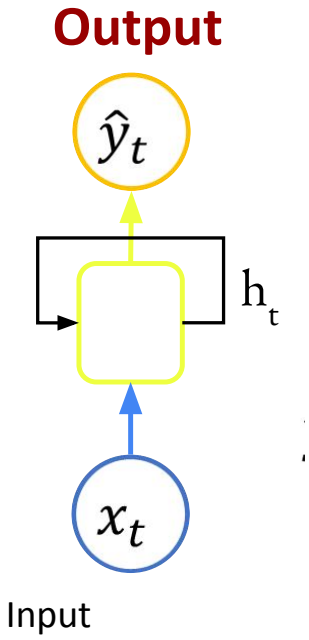


We can process a sequence of vectors **x** by applying a **recurrence formula** at every time step:

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

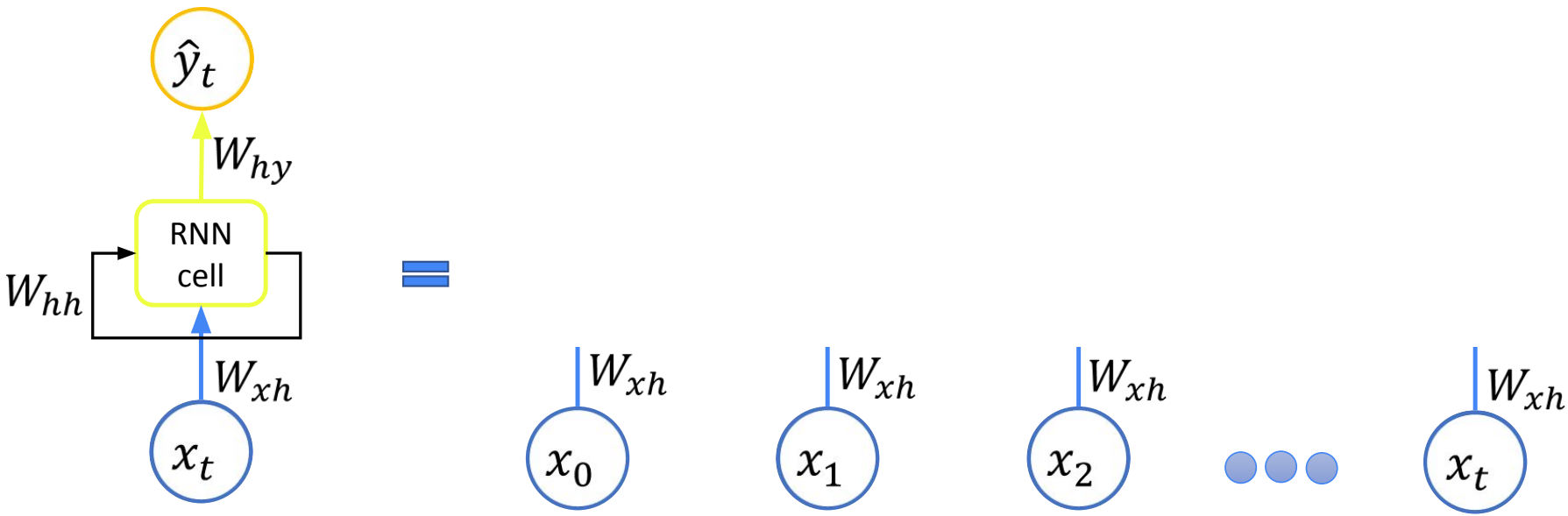
$$h_t = \tanh(W_{hh}h_{t-1} + W_{xh}x_t + b_h)$$

Recurrent Neural Networks

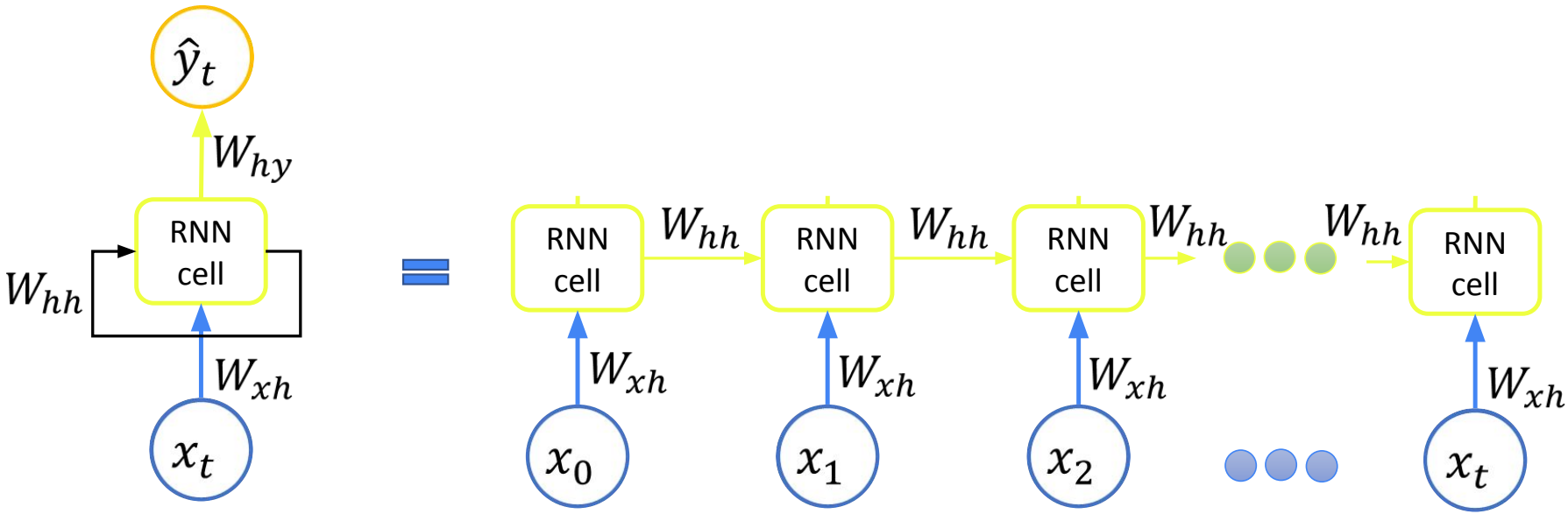


$$y_t = W_{hy} h_t + b_y$$

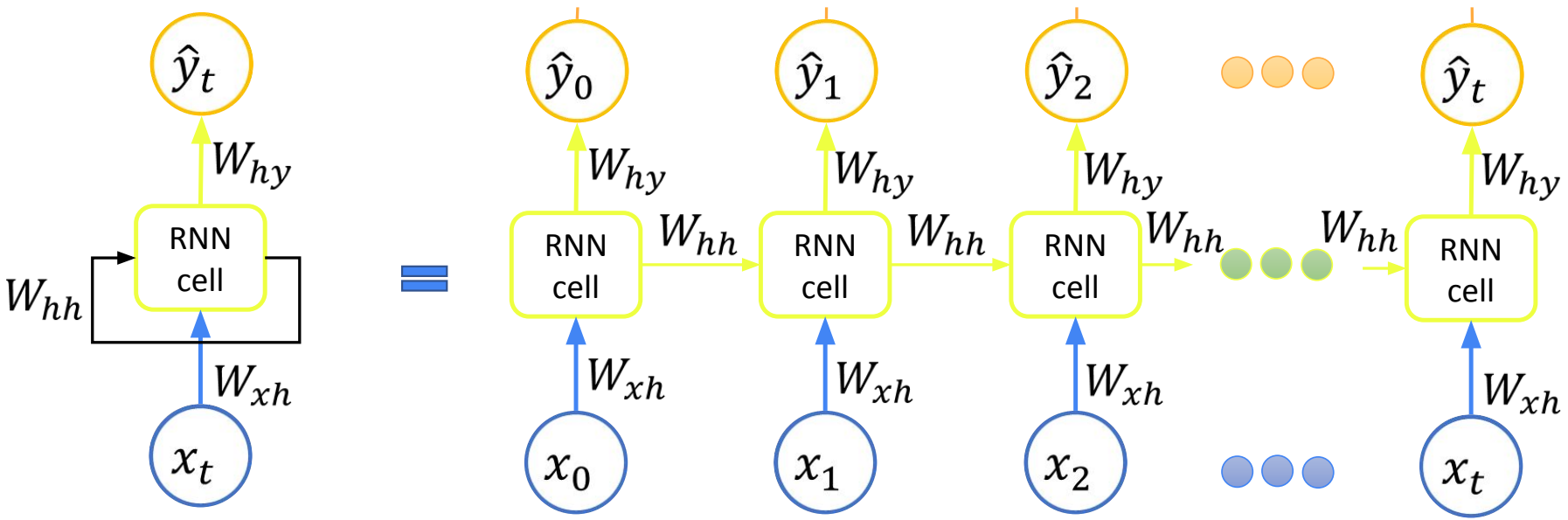
RNN Parameters



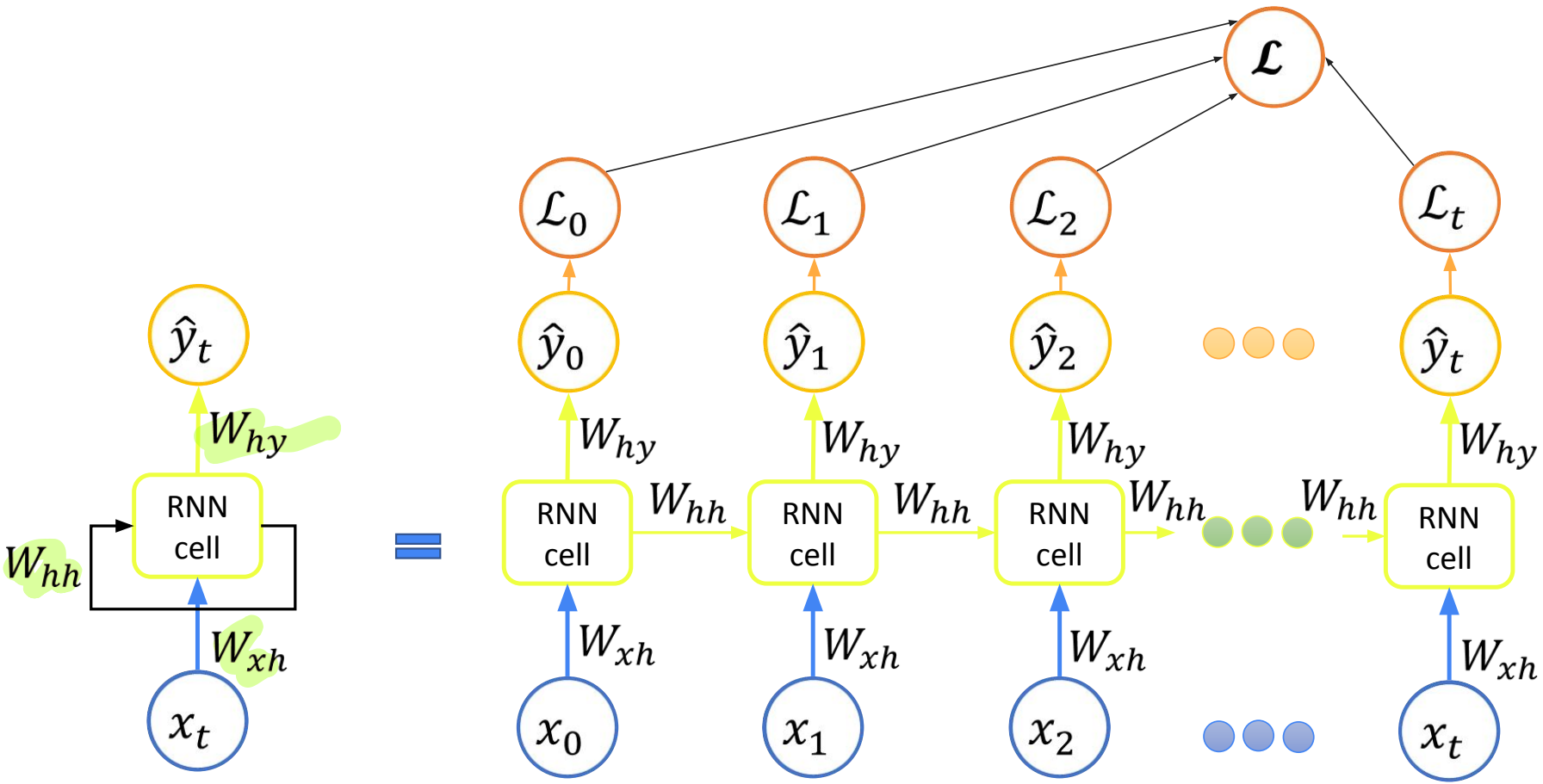
RNN Parameters



RNN Parameters



RNN Computational Graph Across Time



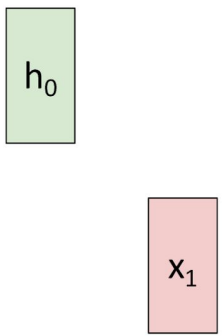
$$h_t = \tanh(W_{xh} x_t + h_{t-1} W_{hh} + b_h)$$

$$\hat{y} = W_{hy} h_t + b_y$$

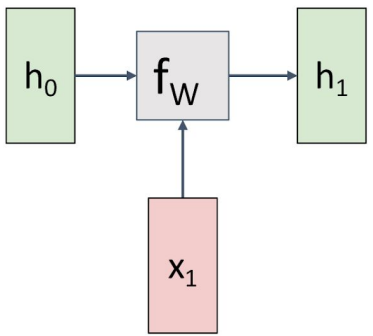
RNN Computational Graph

RNN Computational Graph

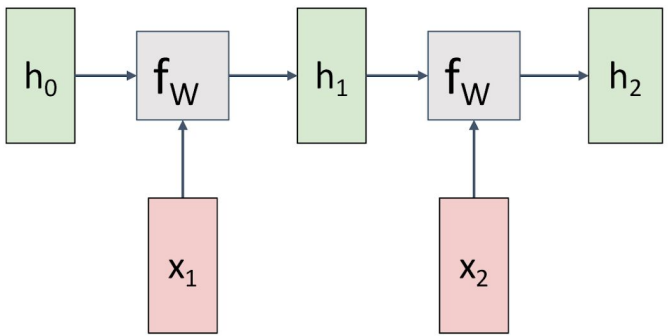
Initial hidden state
Either set to all 0,
Or learn it



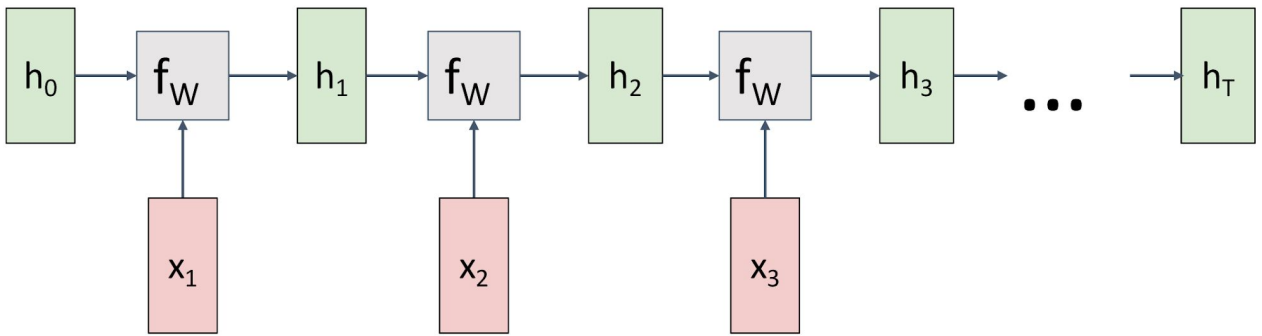
RNN Computational Graph



RNN Computational Graph

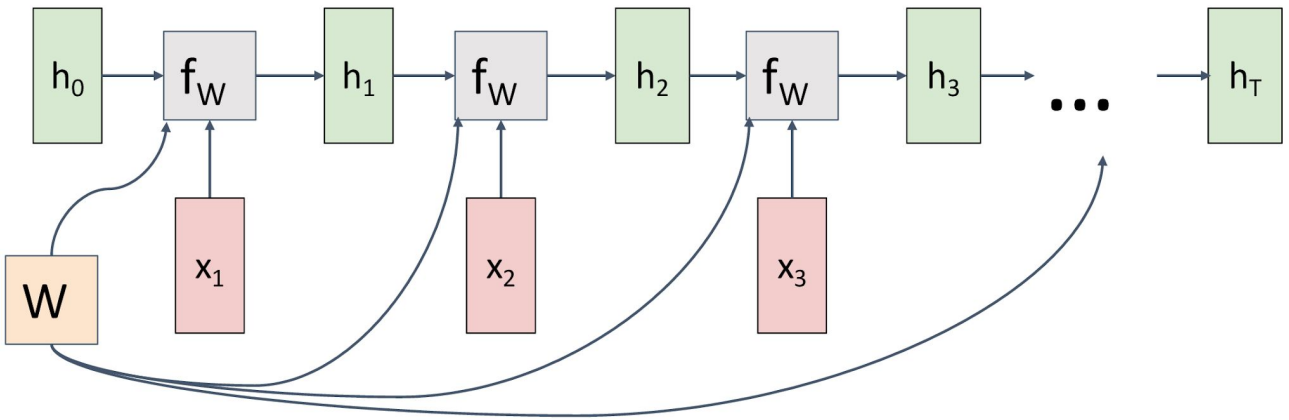


RNN Computational Graph

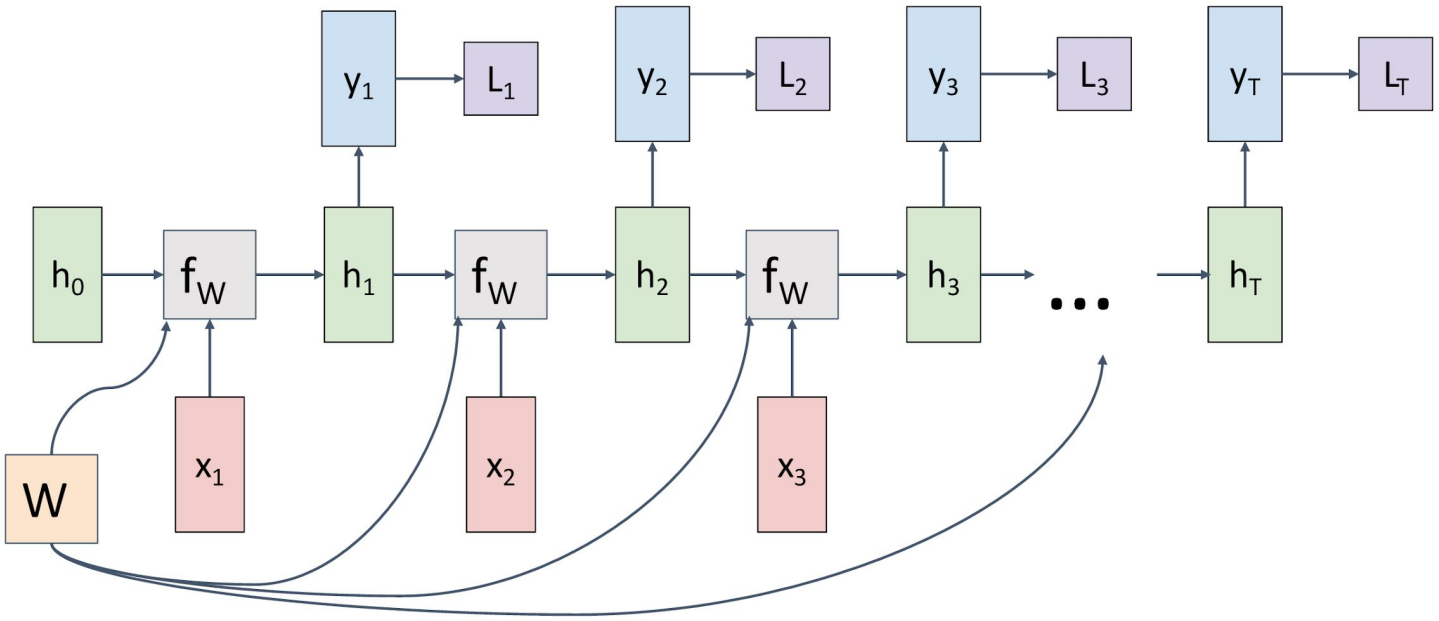


RNN Computational Graph

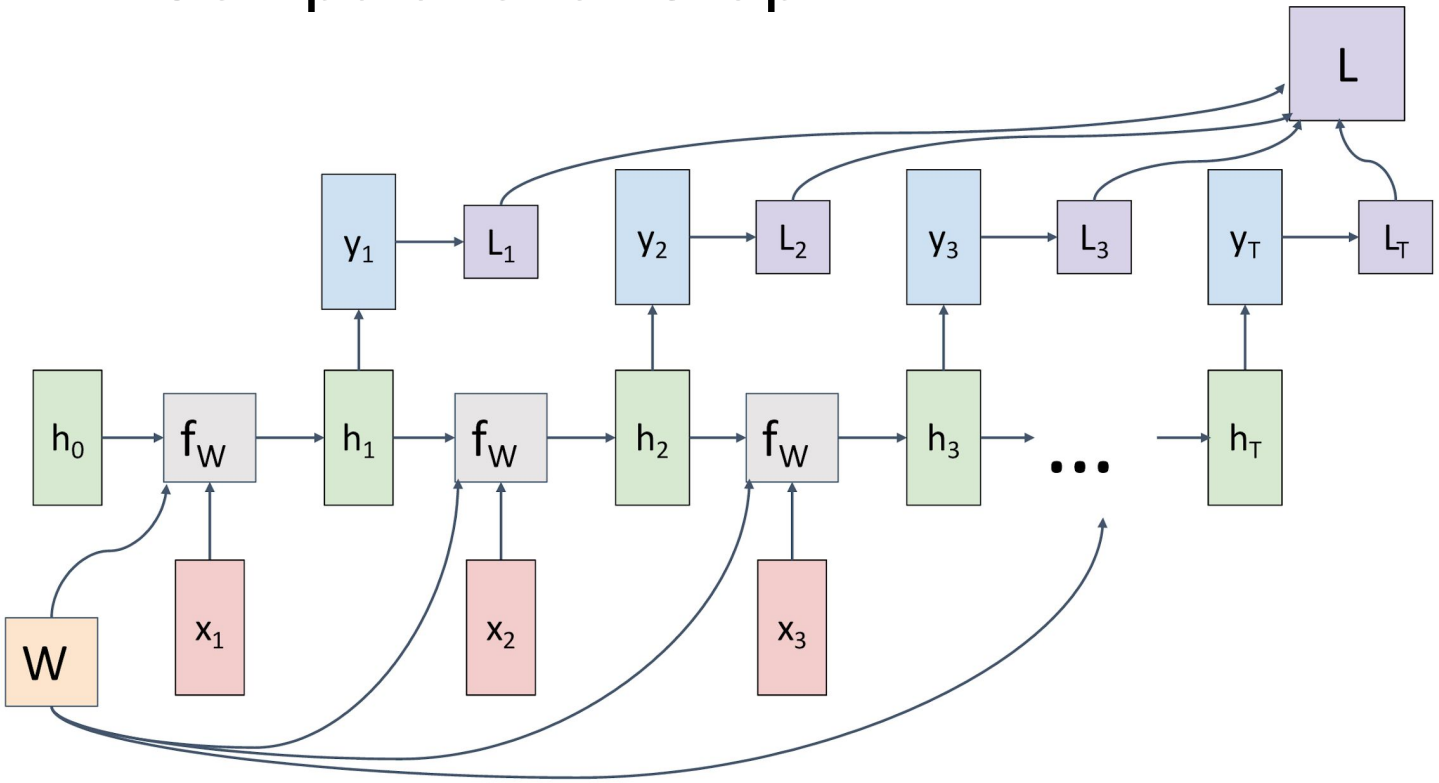
Re-use the same weight matrix at every time-step



RNN Computational Graph

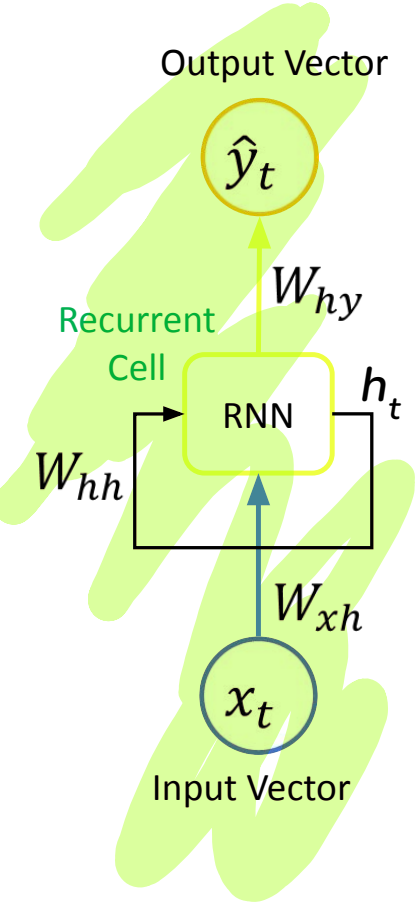


RNN Computational Graph



Training RNNs

RNN State Update

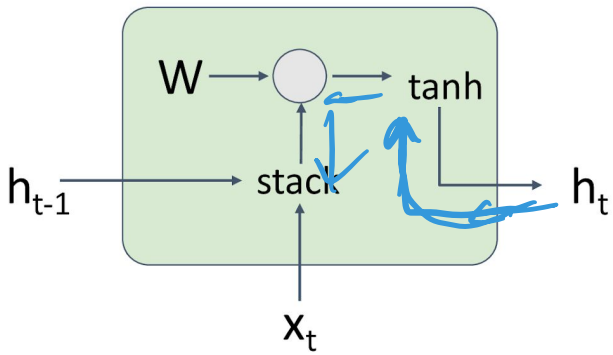


$$y_t = W_{hy}h_t + b_y$$ ← Output Vector

$$h_t = \tanh(W_{hh}h_{t-1} + W_{xh}x_t + b_h)$$ — Cell update

x_t ← Input Vector

Vanilla RNN



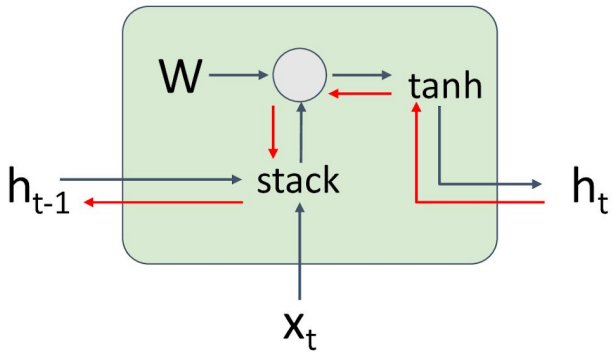
$$h_t = \tanh(W_{hh}h_{t-1} + W_{xh}x_t + b_h)$$

$$= \tanh\left((W_{hh} \quad W_{hx})\begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix} + b_h\right)$$

$$= \tanh\left(W\begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix} + b_h\right)$$

Gradient Flow

Backpropagation from h_t to h_{t-1}



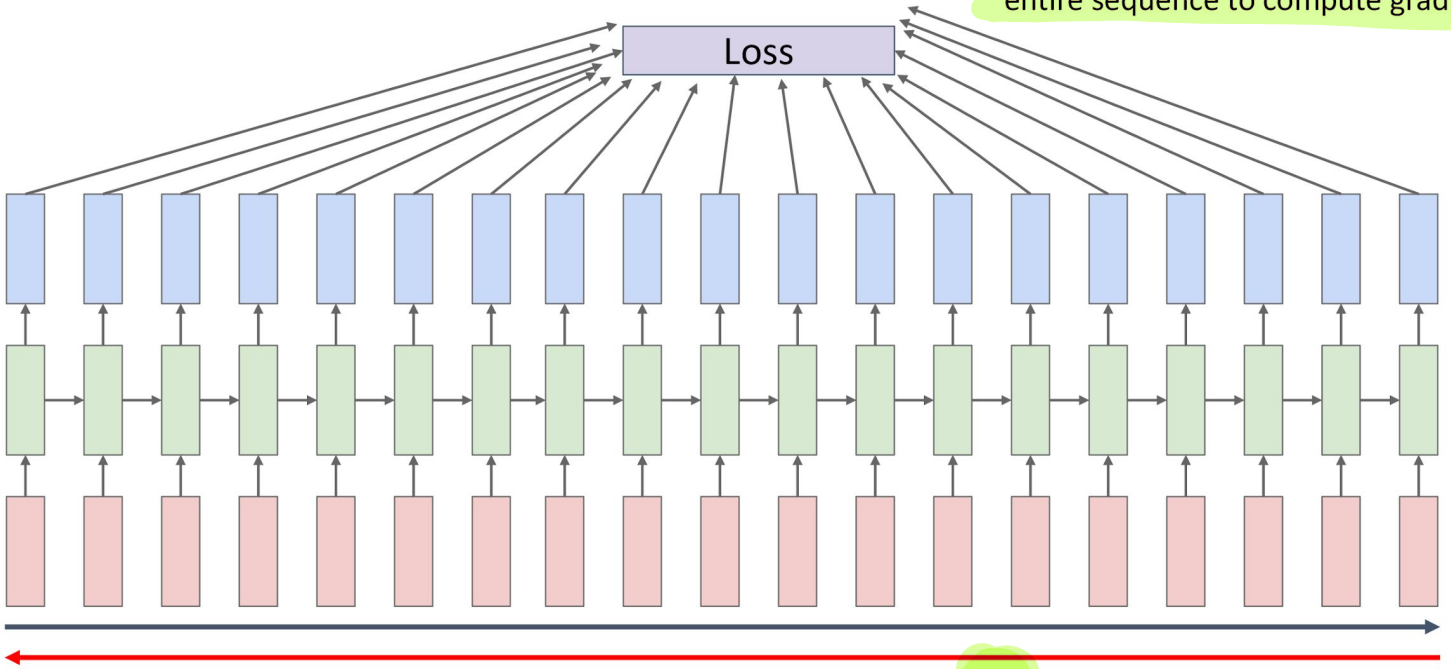
$$
\begin{aligned}
h_t &= \tanh(W_{hh}h_{t-1} + W_{xh}x_t + b_h) \\
&= \tanh\left((W_{hh} \quad W_{hx})\begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix} + b_h\right) \\
&= \tanh\left(W\begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix} + b_h\right)
\end{aligned}
$$

Backpropagation steps

- Input feedforward in the network
- Compute the Loss
- Take the derivative of the Loss with respect to each parameter
- Update parameters to minimize the Loss

Backpropagation Through Time

Forward through entire sequence to compute loss, then backward through entire sequence to compute gradient



Backpropagation Through Time

Forward through entire sequence to compute loss, then backward through entire sequence to compute gradient

Problem: Takes a lot of memory for long sequences!