18AIC301J: DEEP LEARNING TECHNIQUES

B. Tech in ARTIFICIAL INTELLIGENCE, 5th semester

Faculty: **Dr. Athira Nambiar**

Section: A, slot:D

Venue: TP 804

Academic Year: 2022-22

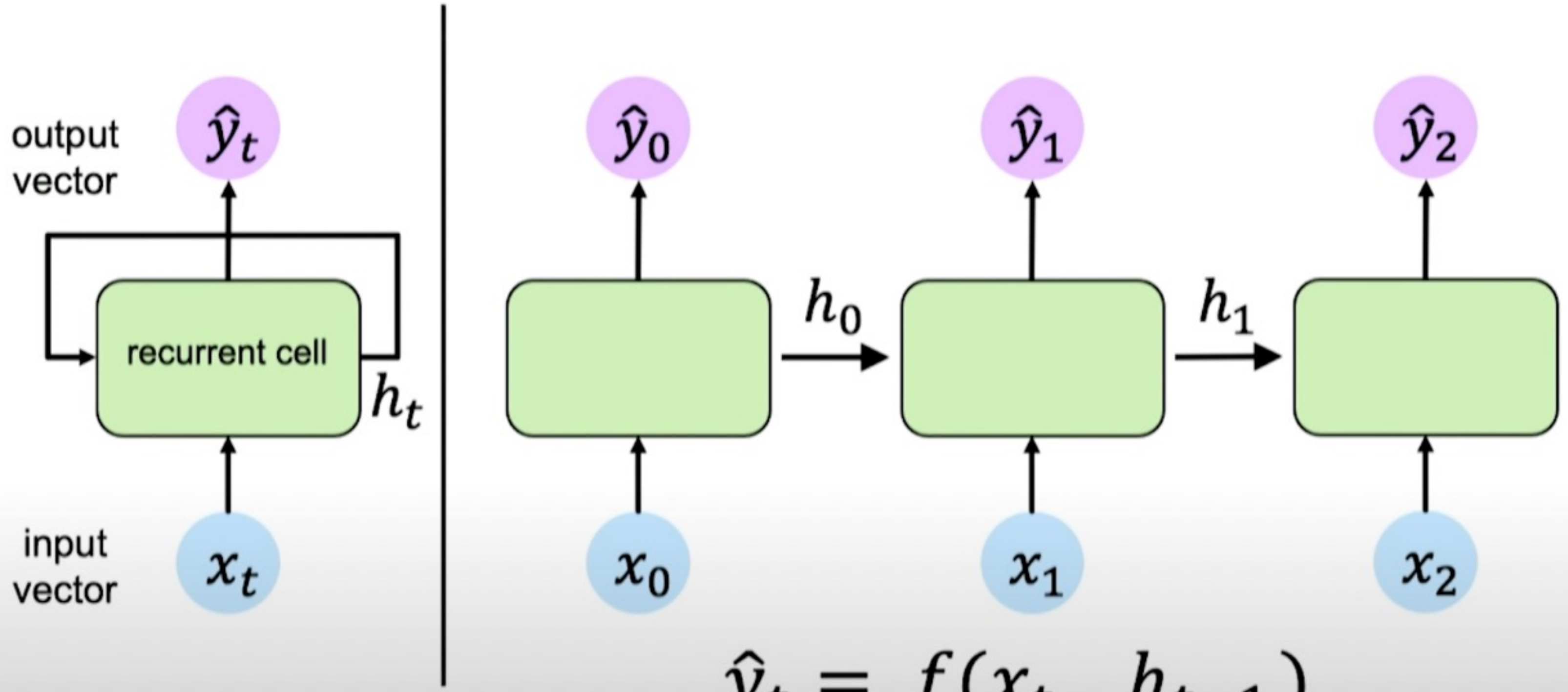
UNIT-4

DenseNet Architecture, Transfer Learning
Need for Transfer Learning, Deep Transfer Learning, Types of Deep Transfer learning, Applications of Transfer learning
Transfer learning implementation using VGG16 model to classify images
Sequence Learning Problems, Recurrent Neural Networks
Backpropagation through time, Unfolded RNN, The problem of exploding and vanishing Gradients, Seq to Seq Models
Building a RNN to perform Character level language modeling.
How gates help to solve the problem of vanishing gradients, Long-Short Term Memory architectures
Dealing with exploding gradients, Gated Recurrent Units, Introduction to Encoder Decoder Models, Applications of Encoder Decoder Models
Build a LSTM network for Named Entity recognition.

UNIT-4

DenseNet Architecture, Transfer Learning
Need for Transfer Learning, Deep Transfer Learning, Types of Deep Transfer learning, Applications of Transfer learning
Transfer learning implementation using VGG16 model to classify images
Sequence Learning Problems, Recurrent Neural Networks
Backpropagation through time, Unfolded RNN, The problem of exploding and vanishing Gradients, Seq to Seq Models
Building a RNN to perform Character level language modeling.
How gates help to solve the problem of vanishing gradients, Long-Short Term Memory architectures
Dealing with exploding gradients, Gated Recurrent Units, Introduction to Encoder Decoder Models, Applications of Encoder Decoder Models
Build a LSTM network for Named Entity recognition.

Neurons with Recurrence

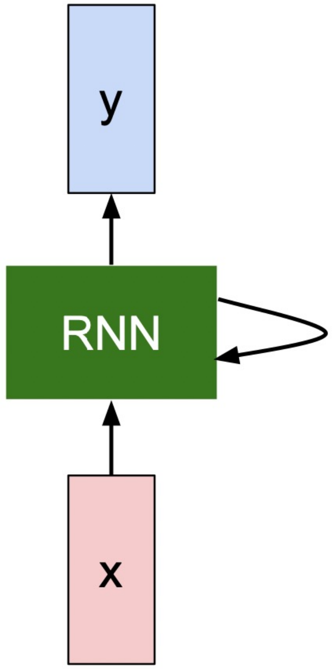


$$\underbrace{\hat{y}_t}_{\text{output}} = f(\underbrace{x_t}_{\text{input}}, \underbrace{h_{t-1}}_{\text{past memory}})$$

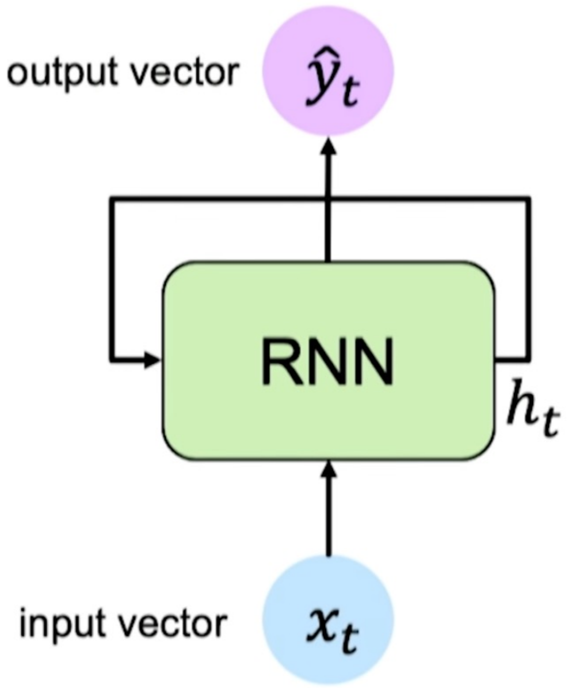
Recurrent Neural Network

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

$$\underbrace{\boxed{h_t}}_{\text{new state}} = \underbrace{\boxed{f_W}}_{\text{some function with parameters W}}(\underbrace{\boxed{h_{t-1}}}_{\text{old state}}, \underbrace{\boxed{x_t}}_{\text{input vector at some time step}})$$



Recurrent Neural Networks (RNNs)



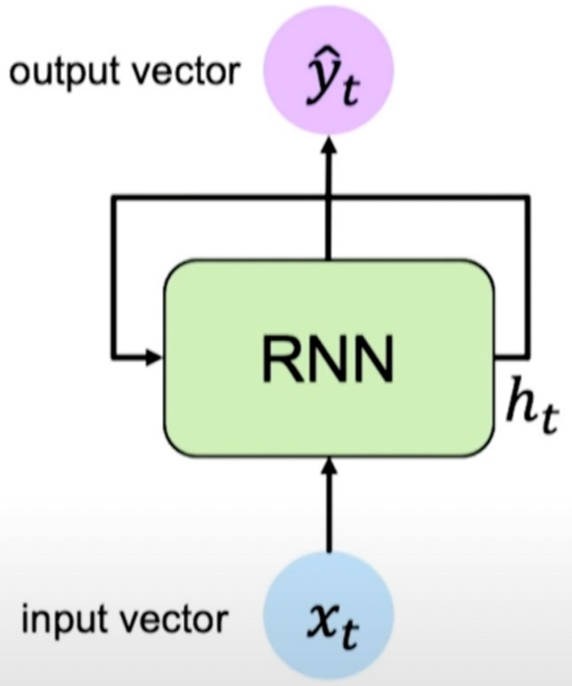
Apply a **recurrence relation** at every time step to process a sequence:

$$\underbrace{\boxed{h_t}}_{\text{cell state}} = \underbrace{\boxed{f_W}}_{\text{function with weights W}}(\underbrace{\boxed{x_t}}_{\text{input}}, \underbrace{\boxed{h_{t-1}}}_{\text{old state}})$$

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

RNNs have a **state**, h_t, that is updated **at each time step** as a sequence is processed

RNN State Update and Output



Output Vector

$$\hat{y}_t = \boldsymbol{W}_{\boldsymbol{hy}}^{\boldsymbol{T}} h_t$$

Update Hidden State

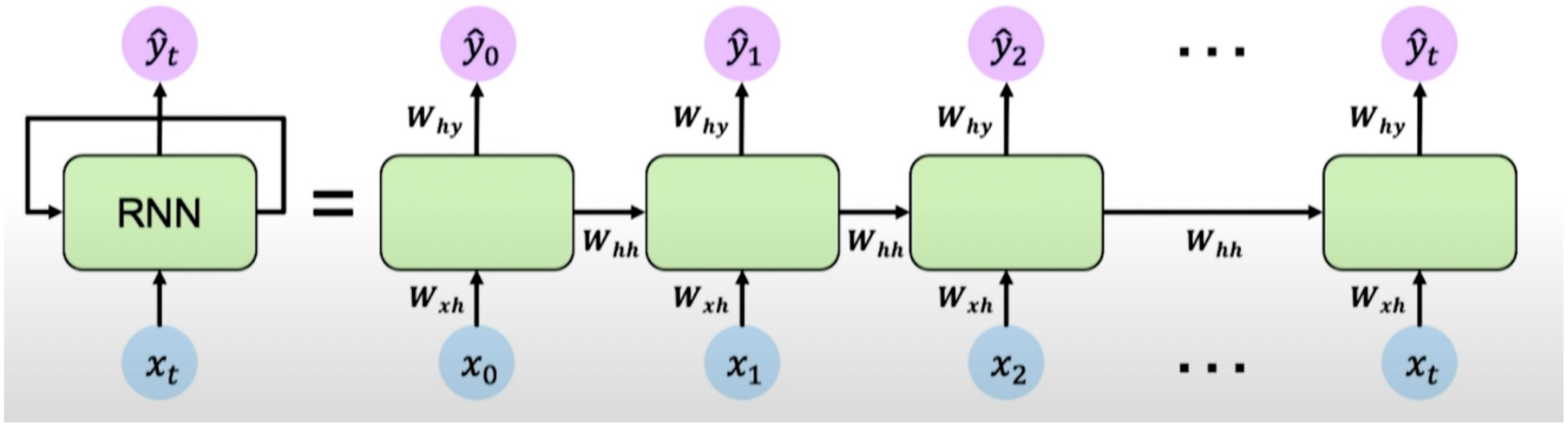
$$h_t = \tanh(\boldsymbol{W}_{\boldsymbol{hh}}^{\boldsymbol{T}} h_{t-1} + \boldsymbol{W}_{\boldsymbol{xh}}^{\boldsymbol{T}} x_t)$$

Input Vector

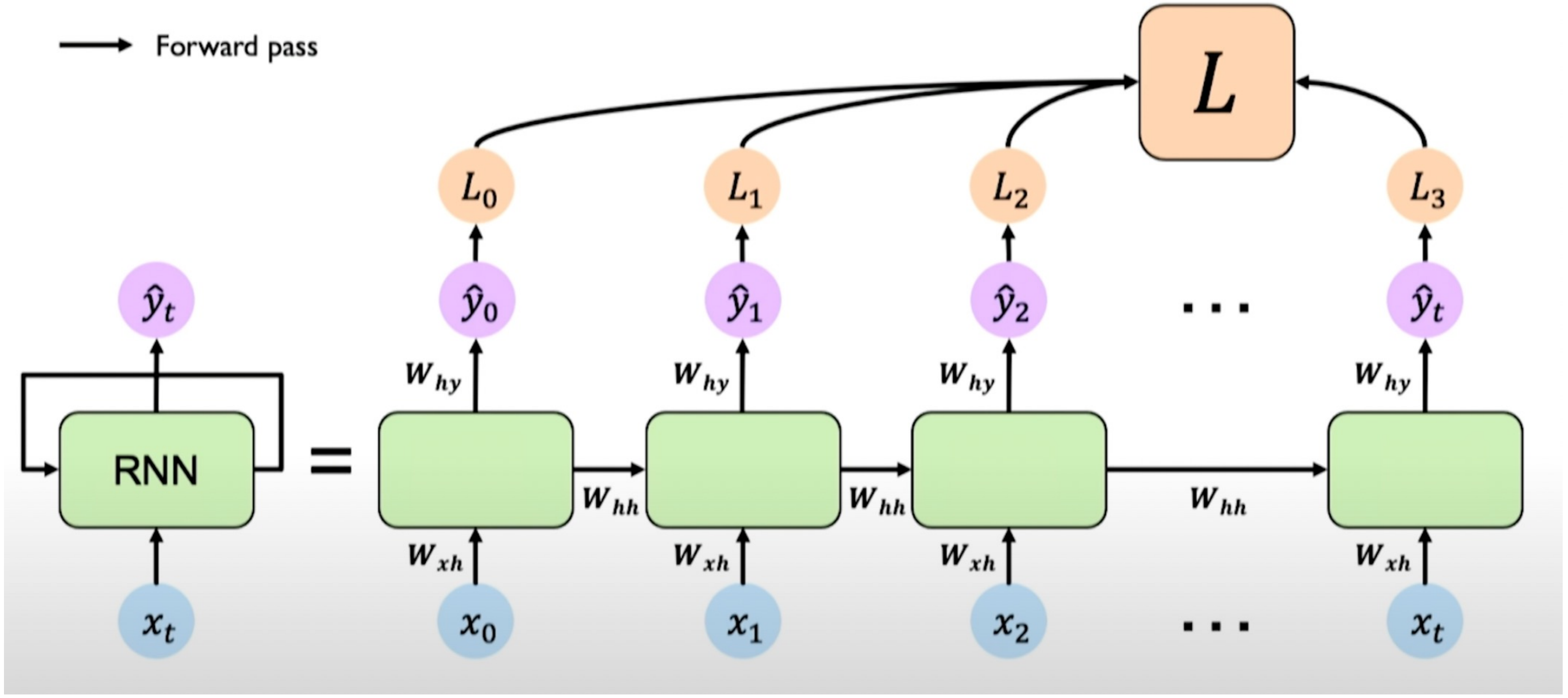
$$x_t$$

RNNs: Computational Graph Across Time

Re-use the **same weight matrices** at every time step

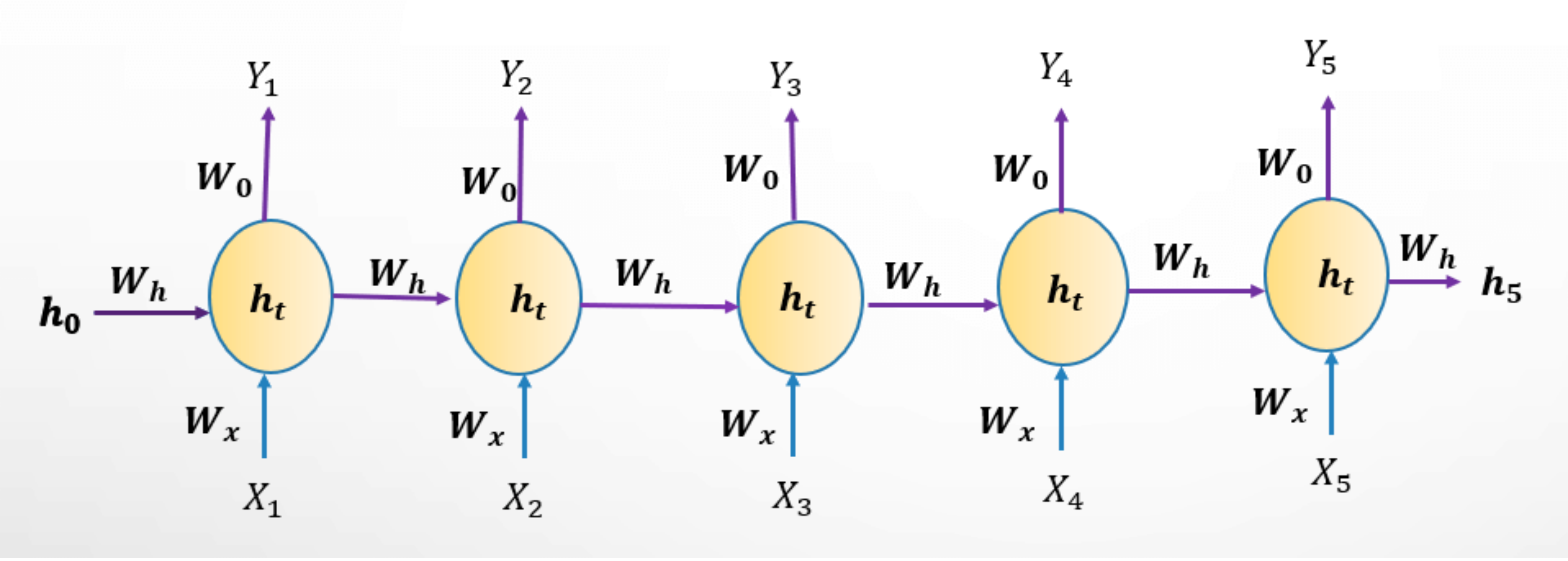


RNNs: Computational Graph Across Time



Back Propagation through time – RNN

We don't train the system on the exact time "t".
We train it according to a particular time "t" as well as everything that has occurred prior to time "t" like the following: t-1, t-2, t-3.



https://towardsmachinelearning.org/what-is-back-propagation-through-time-bptt-in-recurrent-neural-network/

Weight matrices in a single-hidden layer RNN

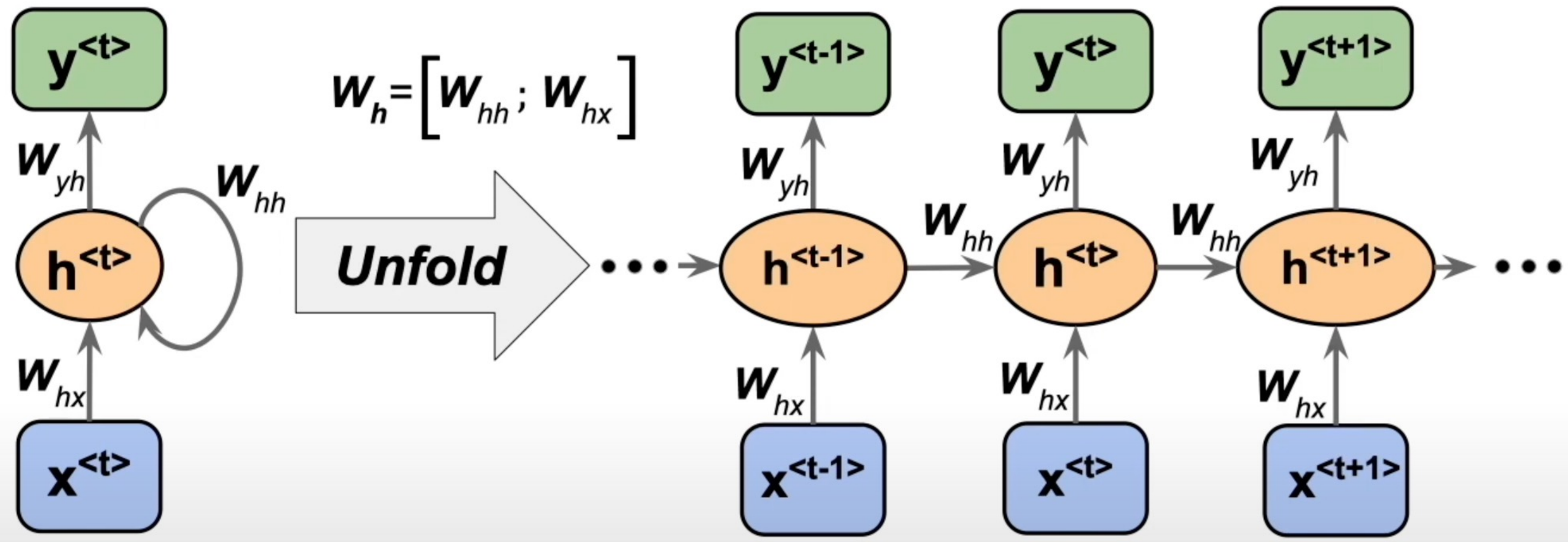


Image source: Sebastian Raschka, Vahid Mirjalili. *Python Machine Learning. 3rd Edition.* Packt, 2019

Backpropagation through time

$$L = \sum_{t=1}^{T} L^{\langle t \rangle}$$

The overall loss can be computed as the sum over all time steps

Werbos, Paul J. "Backpropagation through time: what it does and how to do it." *Proceedings of the IEEE* 78, no. 10 (1990): 1550-1560.

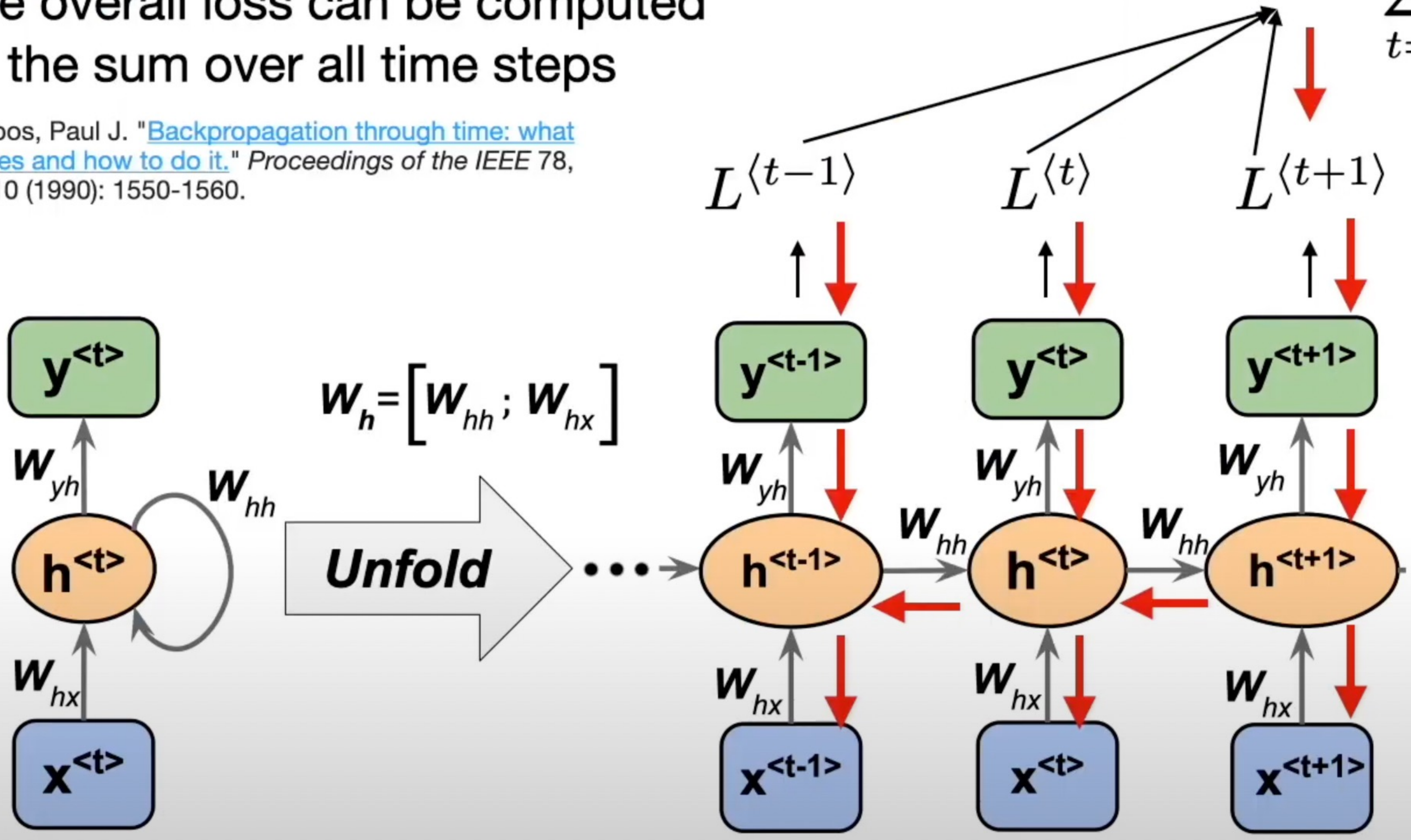
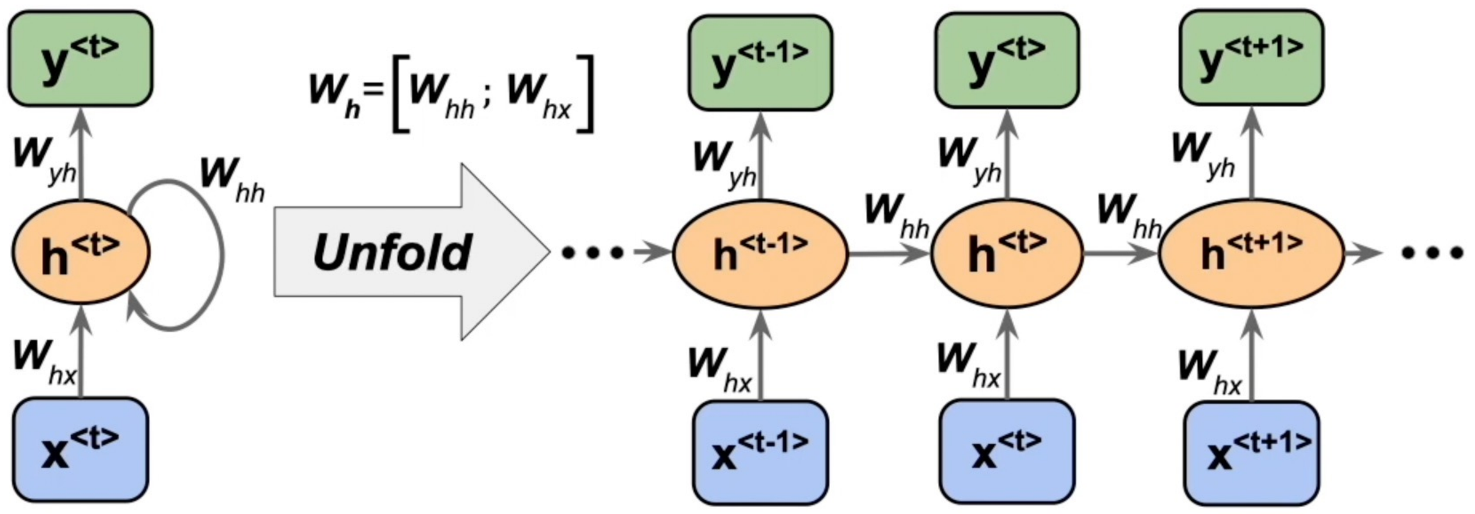


Image source: Sebastian Raschka, Vahid Mirjalili. *Python Machine Learning. 3rd Edition.* Packt, 2019

Backpropagation through time

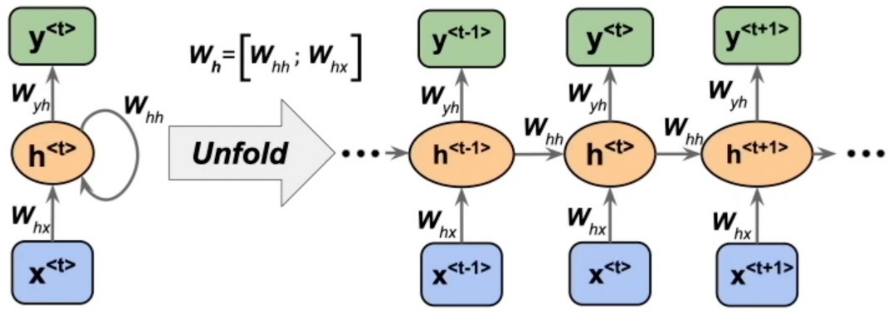


Werbos, Paul J. "Backpropagation through time: what it does and how to do it." *Proceedings of the IEEE* 78, no. 10 (1990): 1550-1560.

$$L = \sum_{t=1}^{T} L^{(t)}$$

$$\frac{\partial L^{(t)}}{\partial \mathbf{W}_{hh}} = \frac{\partial L^{(t)}}{\partial y^{(t)}} \cdot \frac{\partial y^{(t)}}{\partial \mathbf{h}^{(t)}} \cdot \left(\sum_{k=1}^{t} \frac{\partial \mathbf{h}^{(t)}}{\partial \mathbf{h}^{(k)}} \cdot \frac{\partial \mathbf{h}^{(k)}}{\partial \mathbf{W}_{hh}} \right)$$

Backpropagation through time



Werbos, Paul J. "Backpropagation through time: what it does and how to do it." *Proceedings of the IEEE* 78, no. 10 (1990): 1550-1560.

$$L = \sum_{t=1}^{T} L^{(t)} \qquad \frac{\partial L^{(t)}}{\partial \mathbf{W}_{hh}} = \frac{\partial L^{(t)}}{\partial y^{(t)}} \cdot \frac{\partial y^{(t)}}{\partial \mathbf{h}^{(t)}} \cdot \left(\sum_{k=1}^{t} \boxed{\frac{\partial \mathbf{h}^{(t)}}{\partial \mathbf{h}^{(k)}}} \cdot \frac{\partial \mathbf{h}^{(k)}}{\partial \mathbf{W}_{hh}} \right)$$

computed as a multiplication of adjacent time steps:

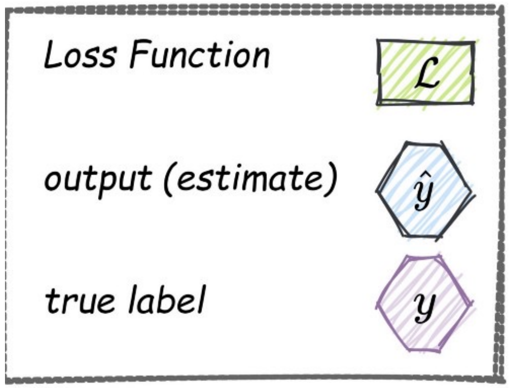
This is very problematic:
Vanishing/Exploding gradient problem!

$$\frac{\partial \mathbf{h}^{(t)}}{\partial \mathbf{h}^{(k)}} = \prod_{i=k+1}^{t} \frac{\partial \mathbf{h}^{(i)}}{\partial \mathbf{h}^{(i-1)}}$$

Backpropagation Through Time (BPTT)

Training an RNN is done by defining a loss function (L) that measures the error between the true label and the output, and minimizes it by using forward pass and backward pass. The following simple RNN architecture summarizes the entire backpropagation through time idea.

For a single time step, the following procedure is done: first, the input arrives, then it processes trough a hidden layer/state, and the estimated label is calculated. In this phase, the loss function is computed to evaluate the difference between the true label and the estimated label. The total loss function, **L**, is computed, and by that, the forward pass is finished. The second part is the backward pass, where the various derivatives are calculated.



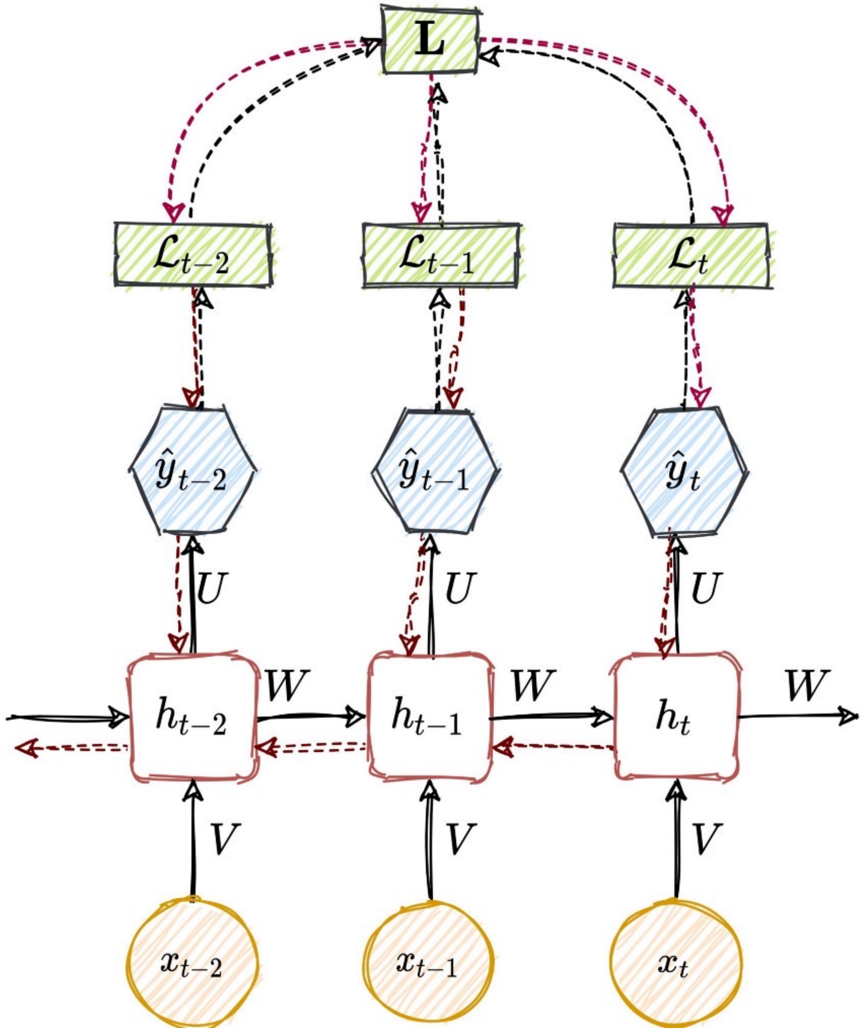
$$\mathbf{L} = \sum_i \mathcal{L}_i\left(\hat{y}_t, y_t\right)$$

Forward Pass:

$$h_t, \hat{y}_t, \mathcal{L}_t, \mathbf{L}$$

Backward Pass:

$$\frac{\partial \mathbf{L}}{\partial U}, \frac{\partial \mathbf{L}}{\partial V}, \frac{\partial \mathbf{L}}{\partial W}, \frac{\partial \mathbf{L}}{\partial b_h}, \frac{\partial \mathbf{L}}{\partial b_y}$$



https://towardsdatascience.com/the-exploding-and-vanishing-gradients-problem-in-time-series-6b87d558d22

The training of RNN is not trivial, as we backpropagate gradients **through layers** and also **through time**. Hence, in each time step we have to sum up all the previous contributions until the current one, as given in the equation:

$$\frac{\partial \mathbf{L}}{\partial W} = \sum_{i=0}^{T} \frac{\partial \mathcal{L}_i}{\partial W} \propto \sum_{i=0}^{T} \left(\prod_{i=k+1}^{y} \frac{\partial h_i}{\partial h_{i-1}} \right) \frac{\partial h_k}{\partial W}$$

Image by Author

In this equation, the contribution of a state at time step **k** to the gradient of the entire loss function **L,** at time step t=T is calculated. The challenge during the training is in the ratio of the hidden state:

$$\frac{\partial \mathbf{L}}{\partial W} \propto \sum_{i=0}^{T} \left(\prod_{i=k+1}^{y} \frac{\partial h_i}{\partial h_{i-1}} \right) \frac{\partial h_k}{\partial W}$$

The Vanishing and Exploding Gradients Problem

Two common problems that occur during the backpropagation of time-series data are the vanishing and exploding gradients. The equation above has two problematic cases:

1. ***Vanishing gradient*** $\left\| \frac{\partial h_i}{\partial h_{i-1}} \right\|_2 < 1$
2. ***Exploding gradient*** $\left\| \frac{\partial h_i}{\partial h_{i-1}} \right\|_2 > 1$

In the first case, the term goes to zero exponentially fast, which makes it difficult to learn some long period dependencies. This problem is called the *vanishing gradient*.

In the second case, the term goes to infinity exponentially fast, and their value becomes a NaN due to the unstable process. This problem is called the *exploding gradient.*

The Vanishing and Exploding Gradients Problem

Two common problems that occur during the backpropagation of time-series data are the vanishing and exploding gradients. The equation above has two problematic cases:

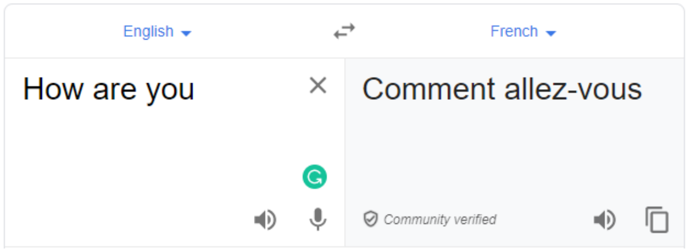
https://towardsmachinelearning.org/what-is-back-propagation-through-time-bptt-in-recurrent-neural-network/

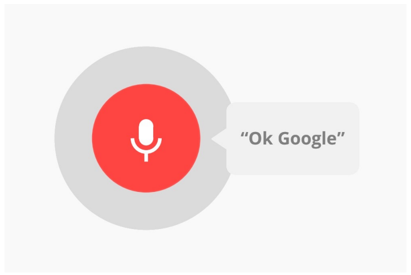
Seq to Seq Models

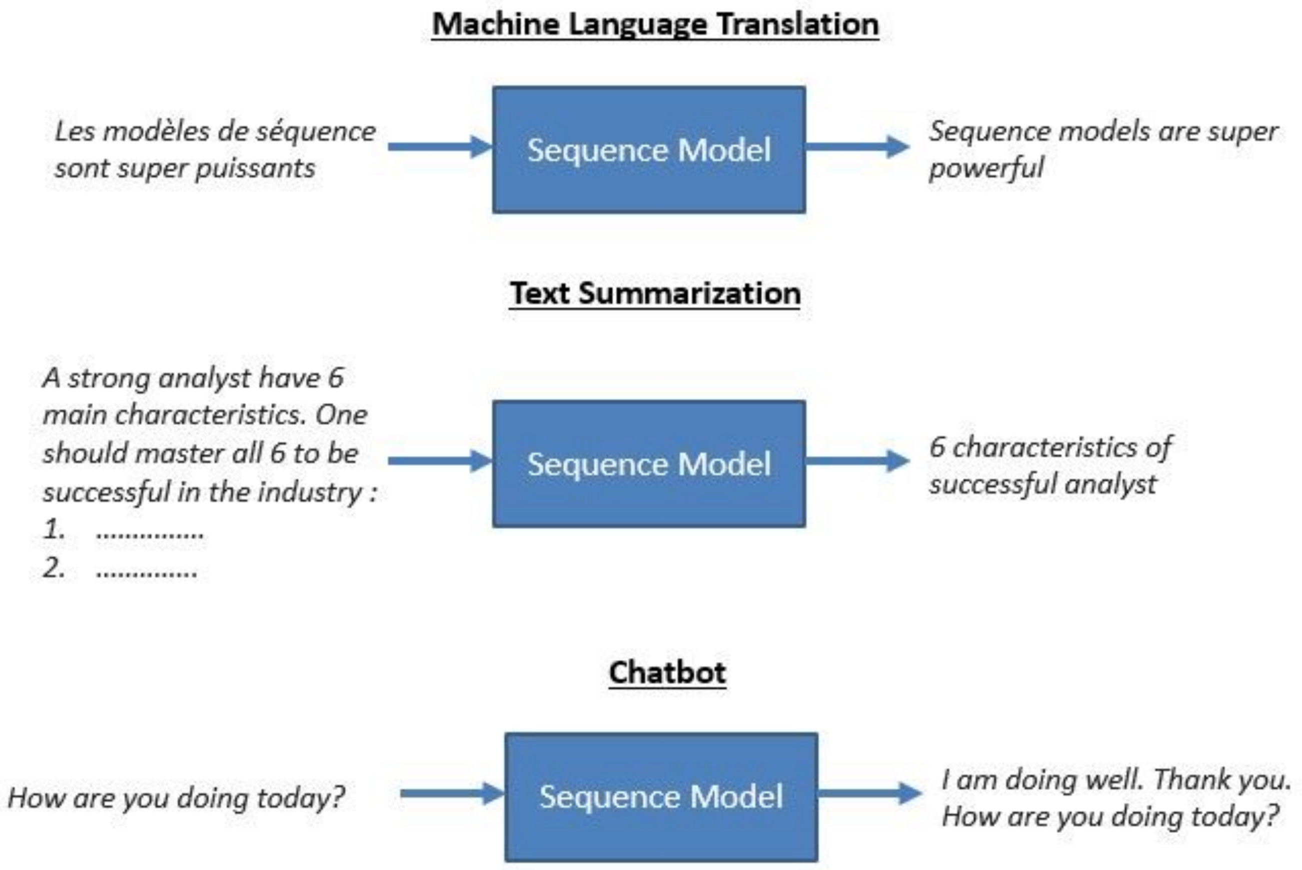
Sequence-to-sequence (abrv. Seq2Seq) models are deep learning models that have achieved a lot of success in tasks like machine translation, text summarization, and image captioning.

Sequence to Sequence (often abbreviated to seq2seq) models is a special class of Recurrent Neural Network architectures that we typically use (but not restricted) to solve complex Language problems like Machine Translation, Question Answering, creating Chatbots, Text Summarization, etc.

https://www.analyticsvidhya.com/blog/2020/08/a-simple-introduction-to-sequence-to-sequence-models/

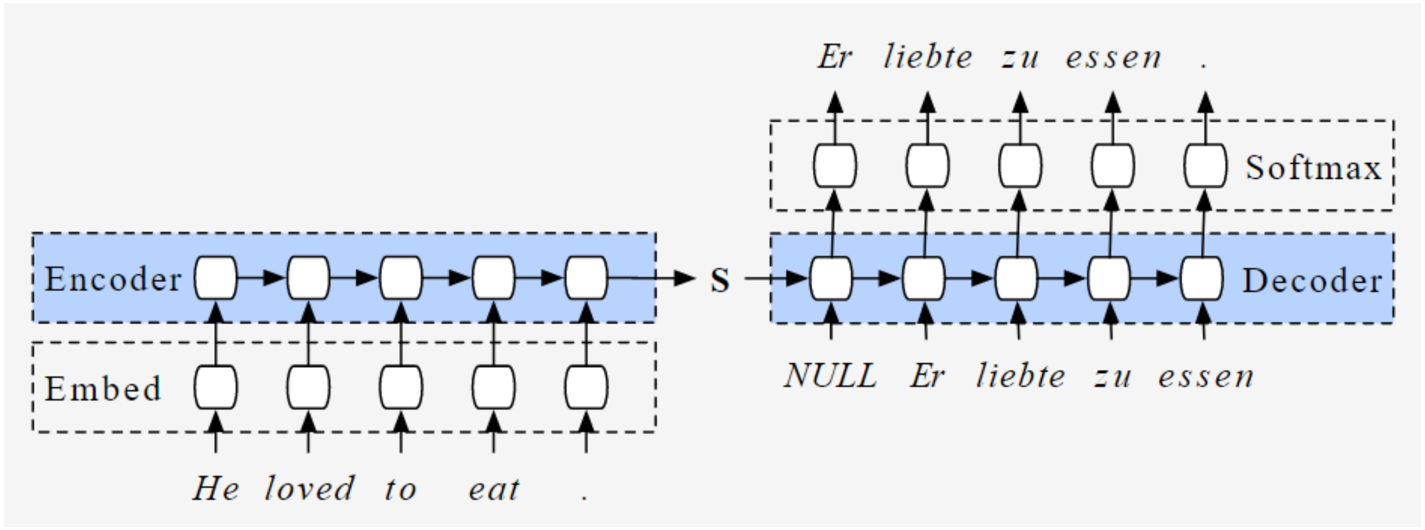

English
French
How are you
Comment allez-vous



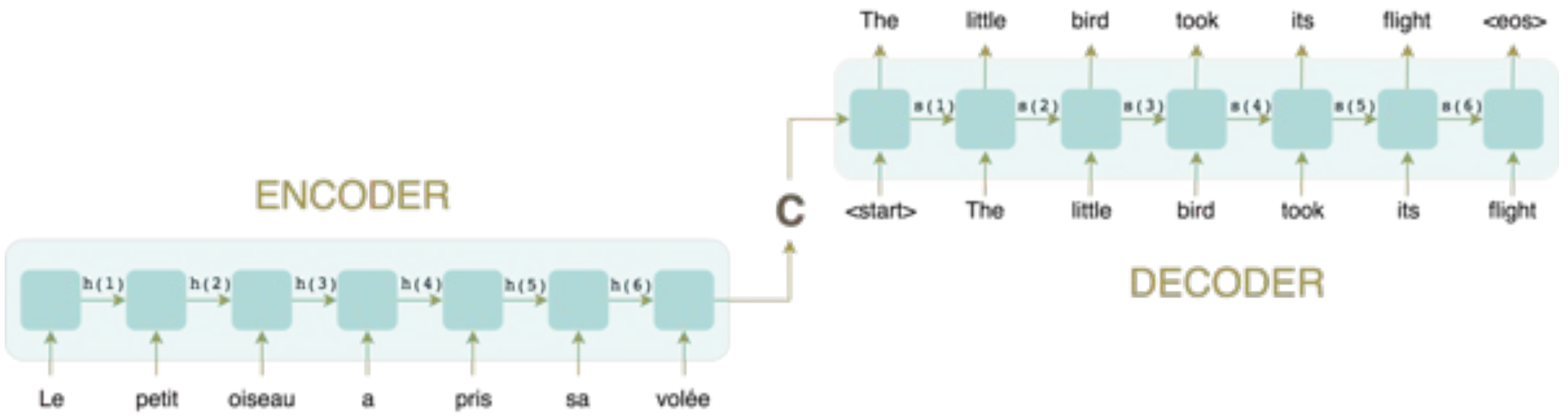

Machine Language Translation
Les modèles de séquence sont super puissants
Sequence Model
Sequence models are super powerful
Text Summarization
A strong analyst have 6 main characteristics. One should master all 6 to be successful in the industry :
1. ....
2. ....
Sequence Model
6 characteristics of successful analyst
Chatbot
How are you doing today?
Sequence Model
I am doing well. Thank you. How are you doing today?

https://www.analyticsvidhya.com/blog/2020/08/a-simple-introduction-to-sequence-to-sequence-models/

Encoder-Decoder Architecture:
The most common architecture used to build Seq2Seq models is Encoder-Decoder architecture.



https://www.youtube.com/watch?v=kNhGWjjWQFk



- Both the encoder and decoder are generally variants of recurrent neural networks (RNNs)

- The encoder RNN receives an input sentence and reads it one token at a time: each cell receives an input word and produces a hidden state as output, which is then fed as input to the next RNN cell, until all the words in the sentence are processed.

- The last-generated hidden state will hopefully capture the gist of all the information contained in every word of the input sentence. This vector, called the context vector, will then be fed as input to the decoder RNN, which will produce the translated sentence one word at a time.

Learning Resources

- Charu C. Aggarwal, Neural Networks and Deep Learning, Springer, 2018.
- Eugene Charniak, Introduction to Deep Learning, MIT Press, 2018.
- Ian Goodfellow, Yoshua Bengio, Aaron Courville, Deep Learning, MIT Press, 2016.
- Michael Nielsen, Neural Networks and Deep Learning, Determination Press, 2015.
- Deng & Yu, Deep Learning: Methods and Applications, Now Publishers, 2013.
- http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture10.pdf
- https://www.youtube.com/watch?v=6niqTuYFZLQ

Thank you