RNN and LSTM: Introduction

James Loy. 2019. Neural Network Projects with Python. Packt Publishing

Stephan Jensen. 2020. Machine Learning for Algorithmic Trading. Packt Publishing

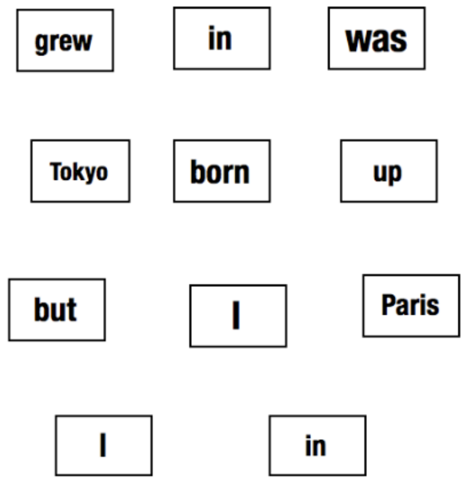
Karthiek Bokka et al. 2019. Deep Learning for Natural Language Processing. Packt Publishing

Sequential Data

- NLP: sentiment analysis, language translation, text prediction
- Time Series prediction

> "I WAS BORN IN PARIS BUT I GREW UP IN TOKYO. THEREFORE, I SPEAK FLUENT ______."

Text prediction problem



Bag of Words

People's opinions can change significantly over time.

Movie Rating

using temporal dynamics led to more accurate movie recommendations (Kohen, 2019)

- Seasonality, Holidays
- Anchoring – Oscar ratings affect a movie rating for a few month (Wu et al., 2017)
- Hedonic adaptation: expectations depend on previous good or bad movies.

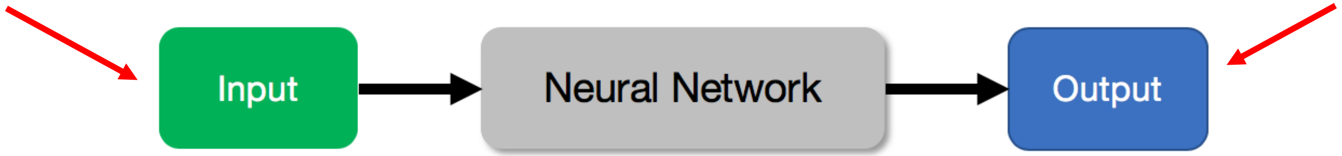
James Loy. Neural network with Python. 2020 Aston Zhang et al. 2020. Dive into Deep Learning. d21.ai

Neural Network

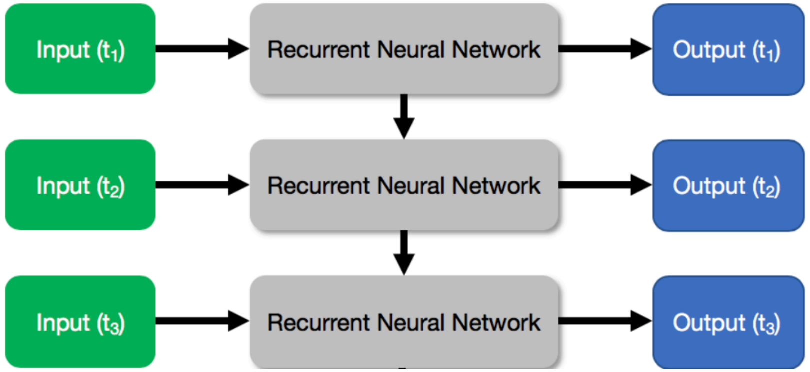
Neural network (CNN, MLP) architecture constraint: no sequential data

Fixed input vector

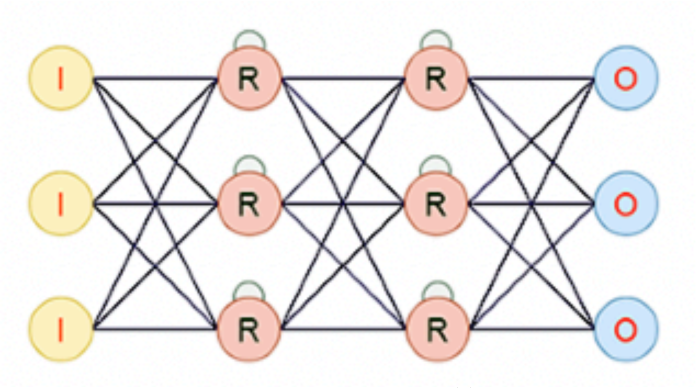
Fixed output vector



Recurrent Neural Network



word = time step



recurrent cells

James Loy. Neural network with Python. 2020

Stefan Leijnen and Fjodor van Veen. 2020. The Neural Network Zoo. In MDPI Proceedings, 47, 9.

RNN Applications

- Speech Recognition: Alexa, Siri, Cortana
- Time Series Predictions: website traffic, Google Maps, Call center traffic
- NLP: Google translate, chatbots (Slack and Google), Question-answering

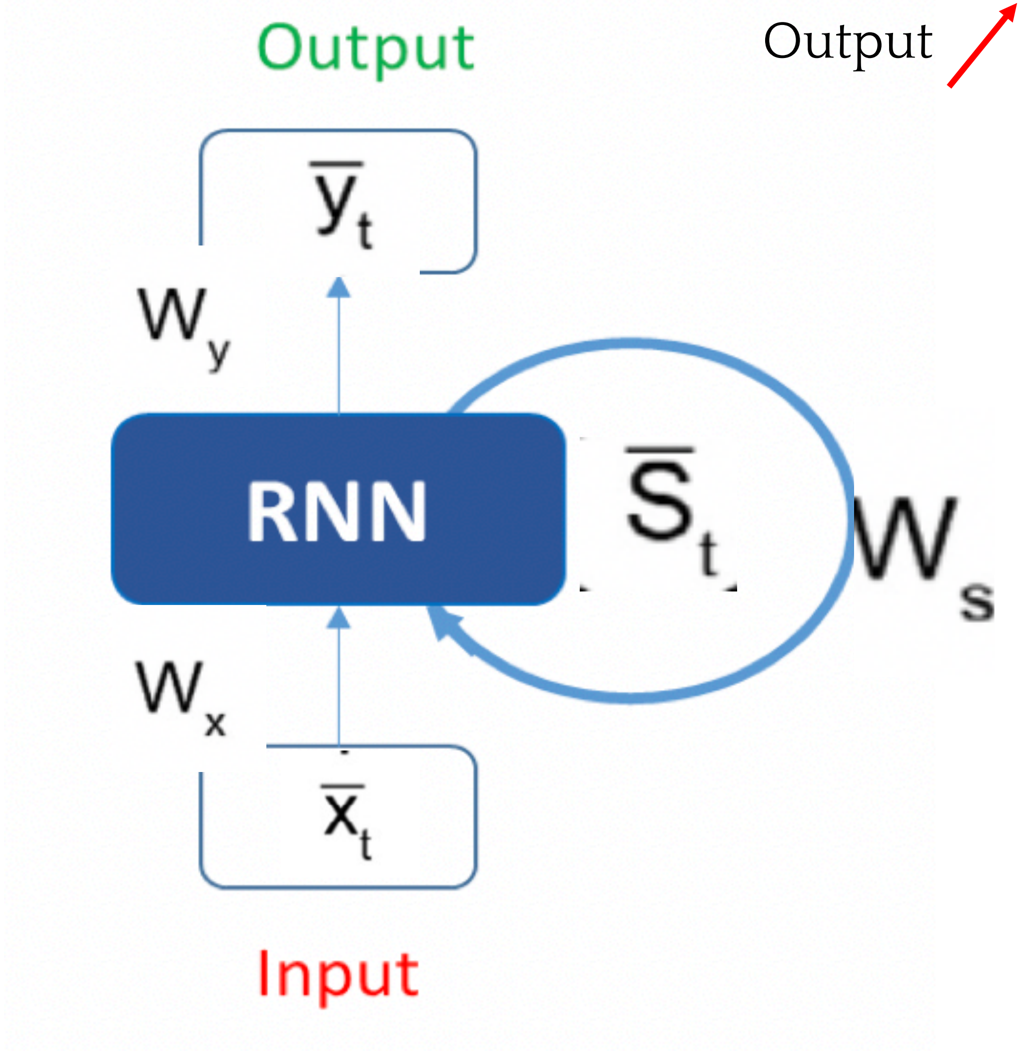
Simple RNN Structure

Previous input

$$\bar{y}_t = F(\bar{x}_t, \bar{x}_{t-1}, \bar{x}_{t-2}, \cdots, \bar{x}_{t-t_0}, W)$$

Output

Function of input and weights



Xt : Current input vector in the input sequence

Yt: $\bar{x}_t$ ıt output vector in the output sequence

St: Current state vector

Wx: Weight matrix connecting the input vector to the state vector

Wy: Weight matrix connecting the state vector to the output vector

Ws: Weight matrix connecting the state vector of previous timestep to the next one

Karthiek Bokka et al. 2019. Ch.5. Deep Learning for Natural Language Processing. Packt Publishing

Text RNN Architecture

indirect

synchronized/
direct

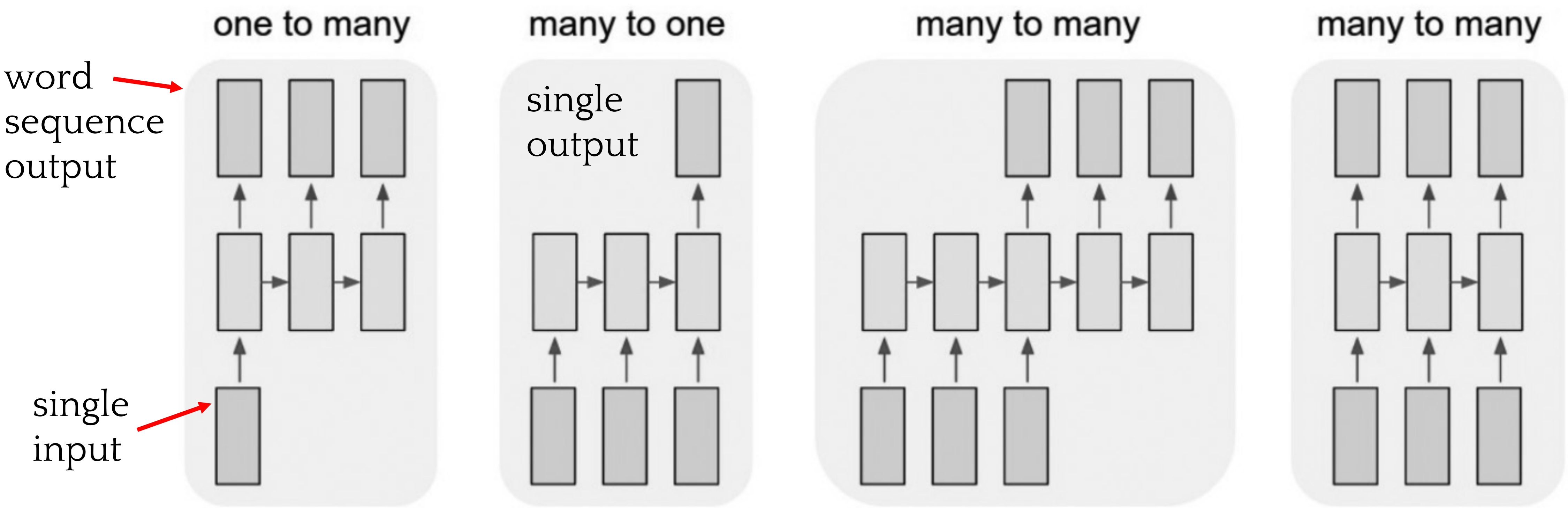


Image captioning

Time series
prediction

Sentiment
Analysis

Machine
translation

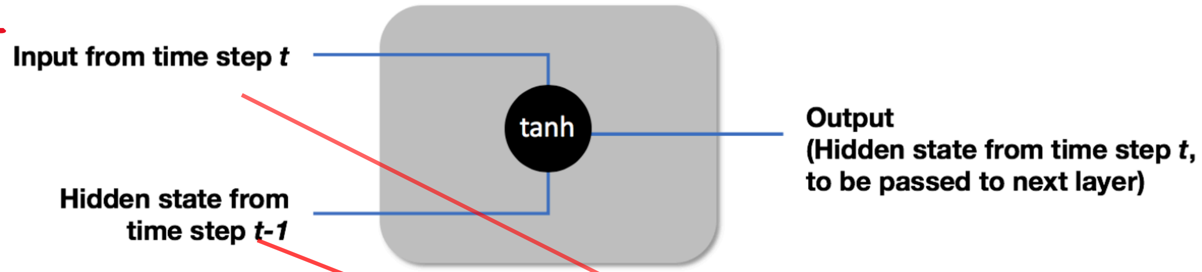
Speech
enhancement

Name Entity
Recognition

Karthiek Bokka et al. 2019. Ch.5. Deep Learning for Natural Language Processing. Packt Publishing

Commonly Used Activation Functions

Each layer = SUM of TWO inputs



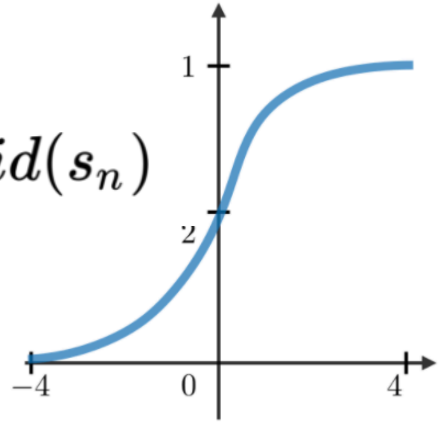
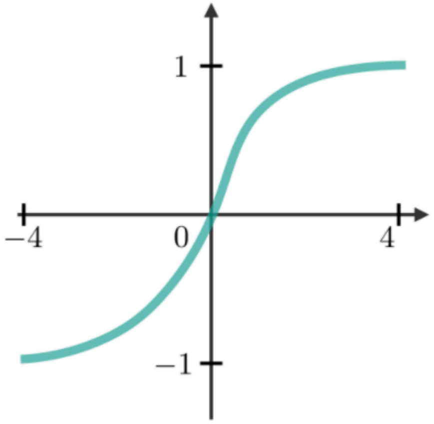
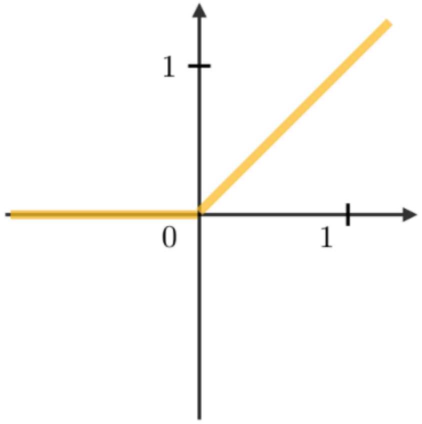
Hidden state for the next layer

$$s_t = tanh(s_{t-1} + x_t)$$

Activation functions

Final output

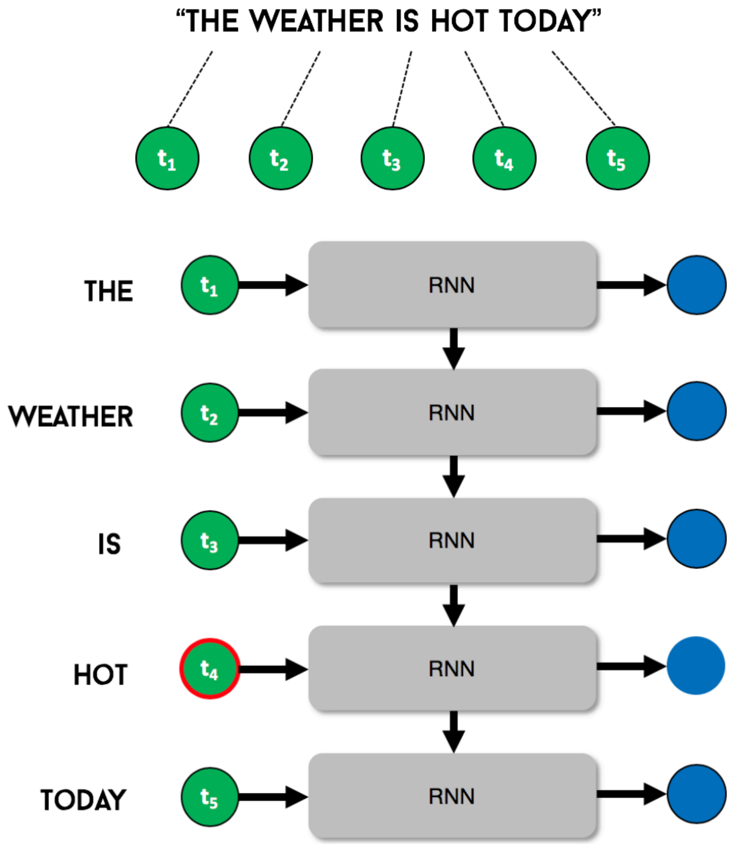
$$O_n = sigmoid(s_n)$$

Sigmoid	Tanh	RELU
$g(z) = \frac{1}{1+e^{-z}}$	$g(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$	$g(z) = \max(0, z)$
		

Afshine Amidi a

RNN: Length Dependency

Short Term Dependency

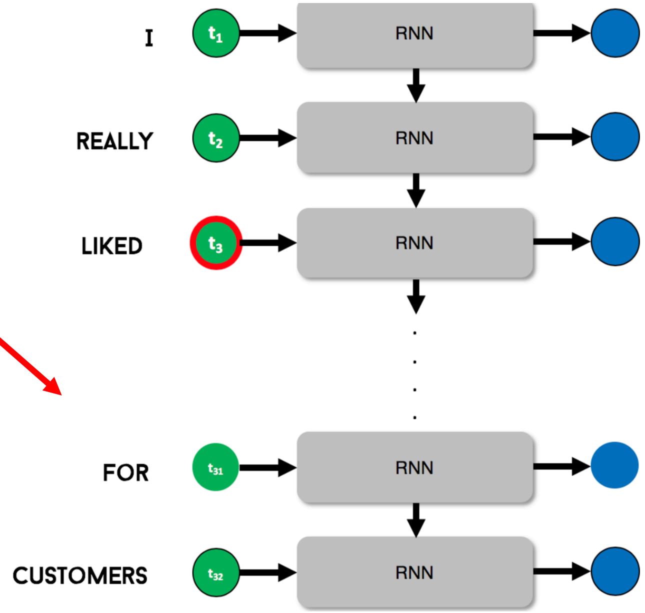


Output: **Not Snowing**

Long Term Dependency

"I really liked the movie but I was disappointed in the service and cleanliness of the cinema. The cinema should be better maintained in order to provide a better experience for customers."

Vanishing Gradient Problem



Output: **?**

James Loy. Neural network with Python. 2020

The Vanishing Gradient Problem

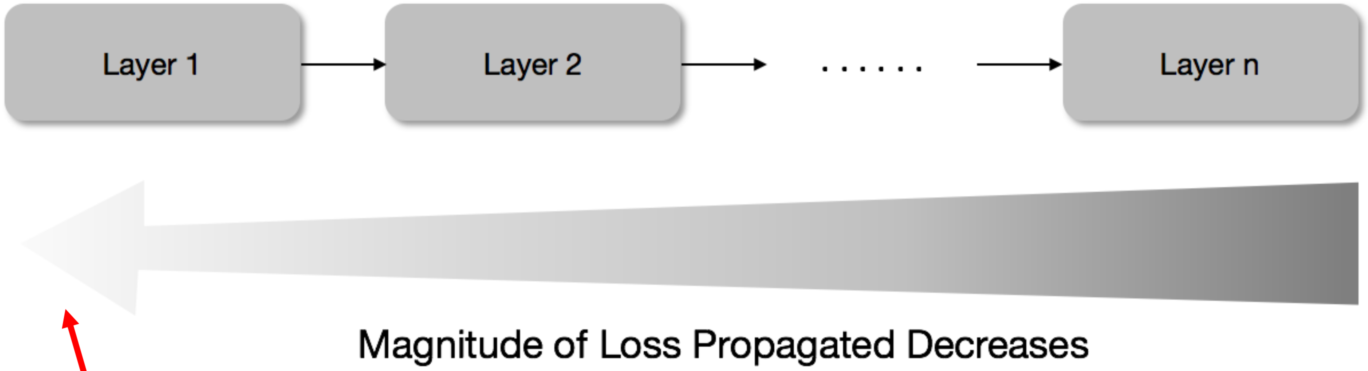
LOSS function

Error metric for evaluating the goodness of predictions:
- Mean Squared Error MSE (regression)
- Categorical Crossentropy (multiclass classification)
- Binary Crossentropy (binary classification)

Backpropagation

Propagating error metrics back and adjusting weights

The LOSS propagated back tends to decrease



Impossible to adjust weighs for the first layers (loss is very small)

James Loy. Neural network with Python. 2020

LSTM: Long Short-Term Memory (Hochreiter & Schmidhuber, 1997)

"I loved this movie! The action sequences were on point and the acting was terrific. Highly recommended!"

Movie rating is positive

Cell States

the current memory

Hidden States

the overall memory (both important and unimportant)

LSTM The ability to selectively remember important inputs

- Forget gate
- Input gate
- Output gate

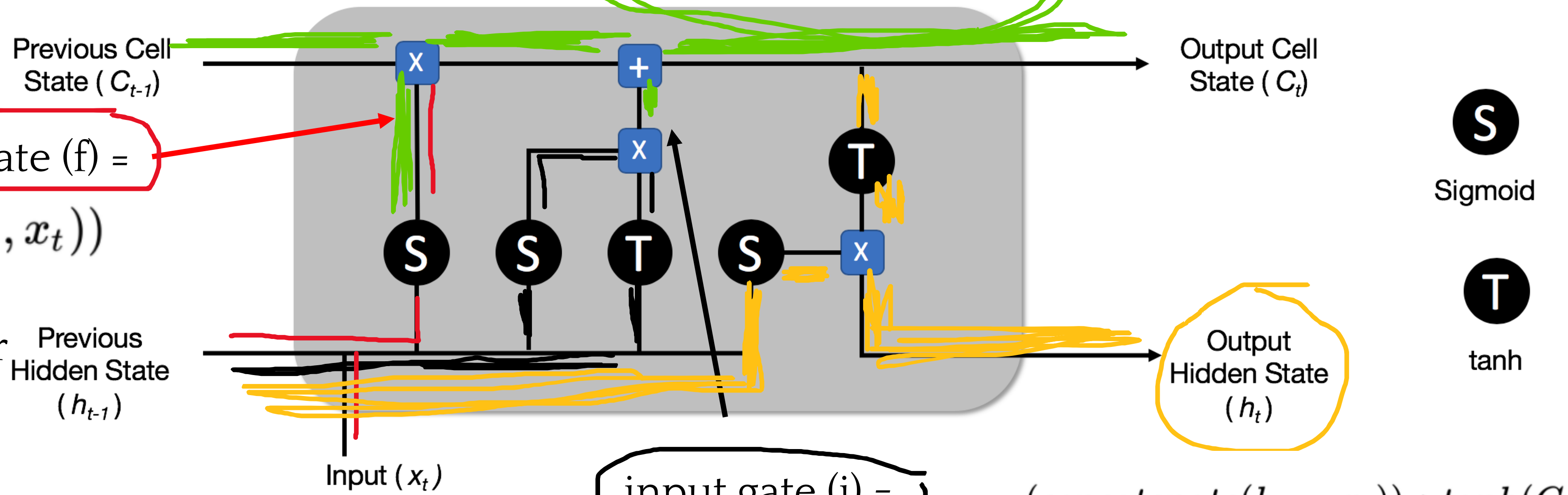
current cell state C = $C_t = (f * C_t) + i$

forget gate (f) =

$$\sigma(concatenate(h_{t-1}, x_t))$$

if (f) = 0 -> forget

if (f) = 1 -> remember



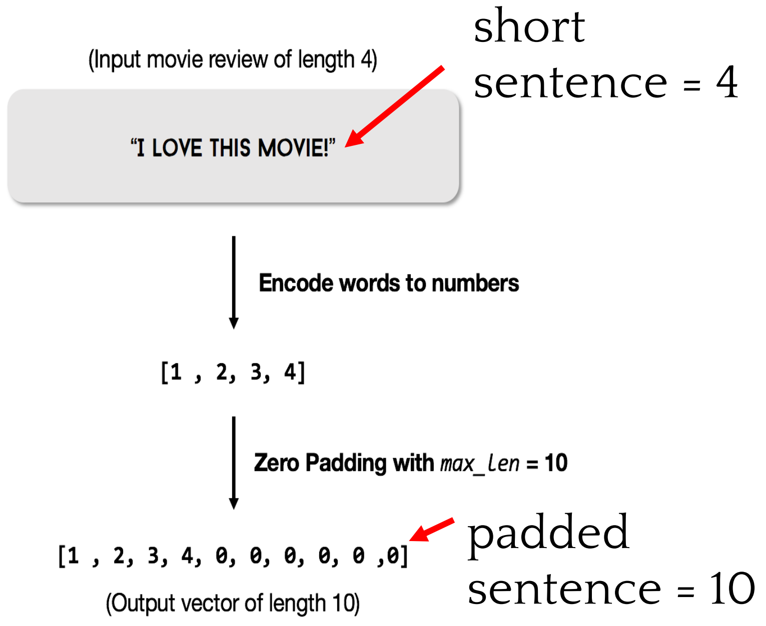
input gate (i) =

$$\sigma(concatenate(h_{t-1}, x_t)) * tanh(concatenate(h_{t-1}, x_t))$$

$$\sigma(concatenate(h_{t-1}, x_t)) * tanh(C_t)$$

James Loy. Neural network with Python. 2020

Movie Review IMDb (Keras)



1. Word vectors must be of the same length. Define MaxLen to truncate long sentences and to pad short sentences with zeros

2. Define Sequential class

```
from keras.models import Sequential
model = Sequential()
```

3. Add embedding layer

```
from keras.layers import Embedding
model.add(Embedding(input_dim = 10000, output_dim = 128))
```

number of unique words

4. Add LSTM layer

```
from keras.layers import LSTM
model.add(LSTM(units=128))
```

5. Add Dense layer

```
from keras.layers import Dense
model.add(Dense(units=1, activation='sigmoid'))
```

6. Summary

```
model.summary()
```

Layer (type)	Output Shape	Param #
embedding_3 (Embedding)	(None, None, 128)	1280000
lstm_3 (LSTM)	(None, 128)	131584
dense_3 (Dense)	(None, 1)	129

James Loy. Neural network with Python. 2020

Full code: https://downloads.packtpub.com/code/9781789138900.zip