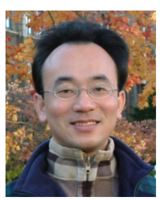
Tamkang
University

Big Data Mining

Recurrent Neural Networks (RNN)

1071BDM12
TLVXM1A (M2244) (8619) (Fall 2018)
(MBA, DBETKU) (3 Credits, Required) [Full English Course]
(Master's Program in Digital Business and Economics)
Mon, 9, 10, 11, (16:10-19:00) (B206)



Min-Yuh Day, Ph.D.
Assistant Professor
Department of Information Management
Tamkang University

http://mail.tku.edu.tw/myday

2018-12-10

Course Schedule (1/2)

Tamkang University

Week	Date	Subject/Topics
1	2018/09/10	Course Orientation for Big Data Mining
2	2018/09/17	ABC: AI, Big Data, Cloud Computing
3	2018/09/24	Mid-Autumn Festival (Day off)
4	2018/10/01	Data Science and Big Data Analytics: Discovering, Analyzing, Visualizing and Presenting Data
5	2018/10/08	Fundamental Big Data: MapReduce Paradigm, Hadoop and Spark Ecosystem
6	2018/10/15	Foundations of Big Data Mining in Python
7	2018/10/22	Supervised Learning: Classification and Prediction
8	2018/10/29	Unsupervised Learning: Cluster Analysis
9	2018/11/05	Unsupervised Learning: Association Analysis

Course Schedule (2/2)

Tamkang University

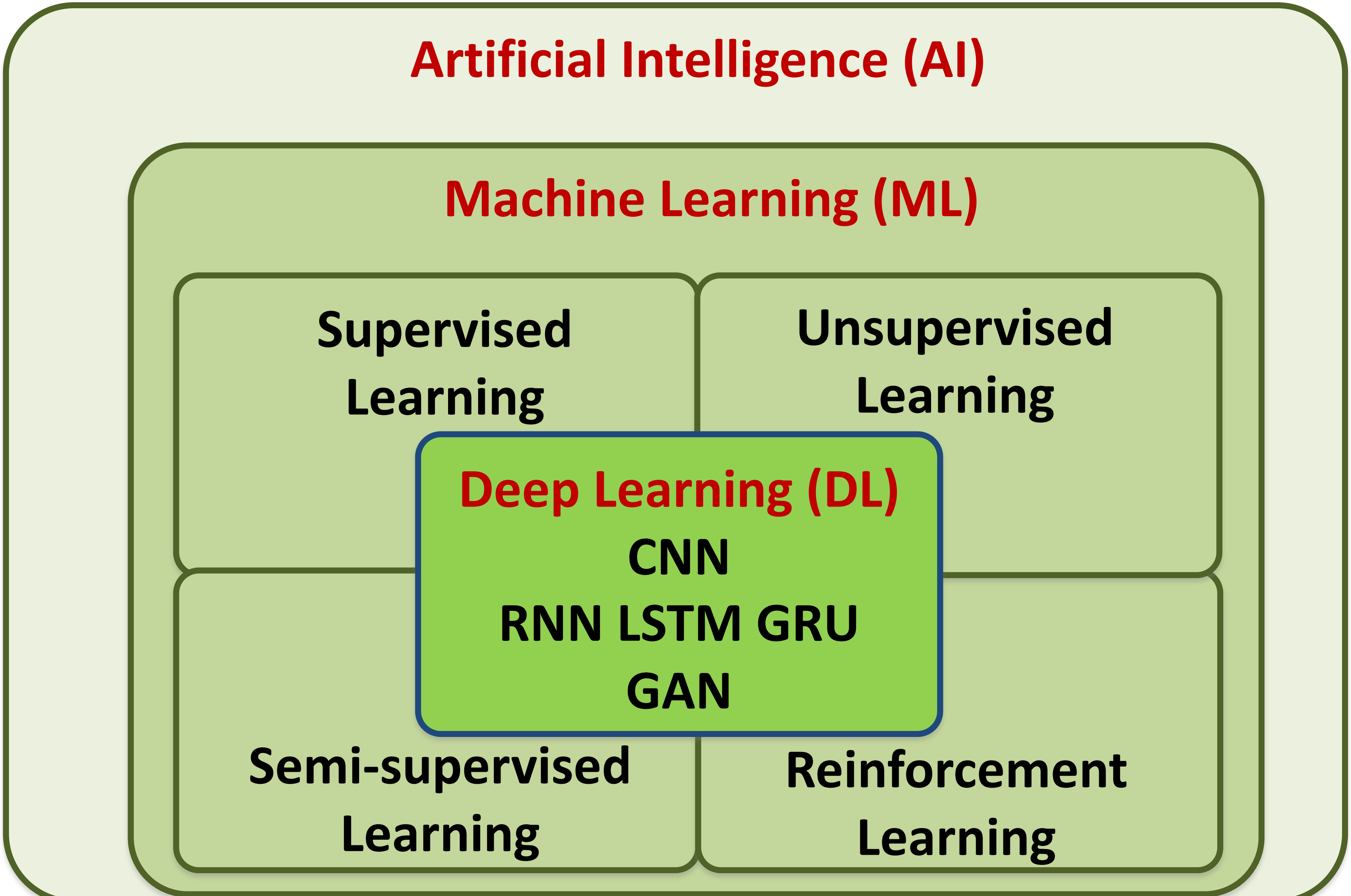
Week	Date	Subject/Topics
10	2018/11/12	Midterm Project Report
11	2018/11/19	Machine Learning with Scikit-Learn in Python
12	2018/11/26	Deep Learning for Finance Big Data with TensorFlow
13	2018/12/03	Convolutional Neural Networks (CNN)
14	2018/12/10	Recurrent Neural Networks (RNN)
15	2018/12/17	Reinforcement Learning (RL)
16	2018/12/24	Social Network Analysis (SNA)
17	2018/12/31	Bridge Holiday (Extra Day Off)
18	2019/01/07	Final Project Presentation

Recurrent Neural Networks (RNN)

Outline

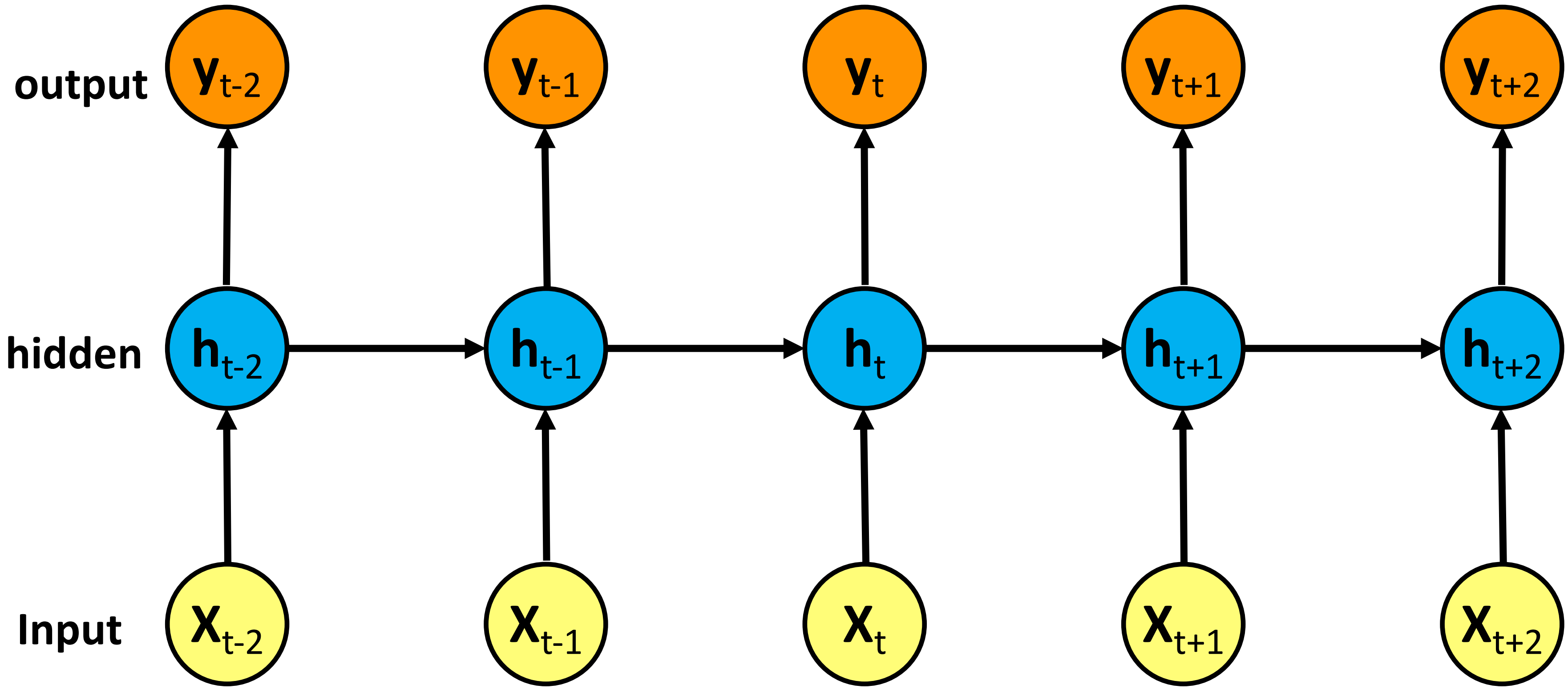
- Recurrent Neural Networks (RNN)
- Long Short Term Memory (LSTM)
- Gated Recurrent Unit (GRU)
- Deep Learning (RNN) for
  Text Analytics (NLP)
- Deep Learning (RNN) for
  Time Series Prediction

AI, ML, DL

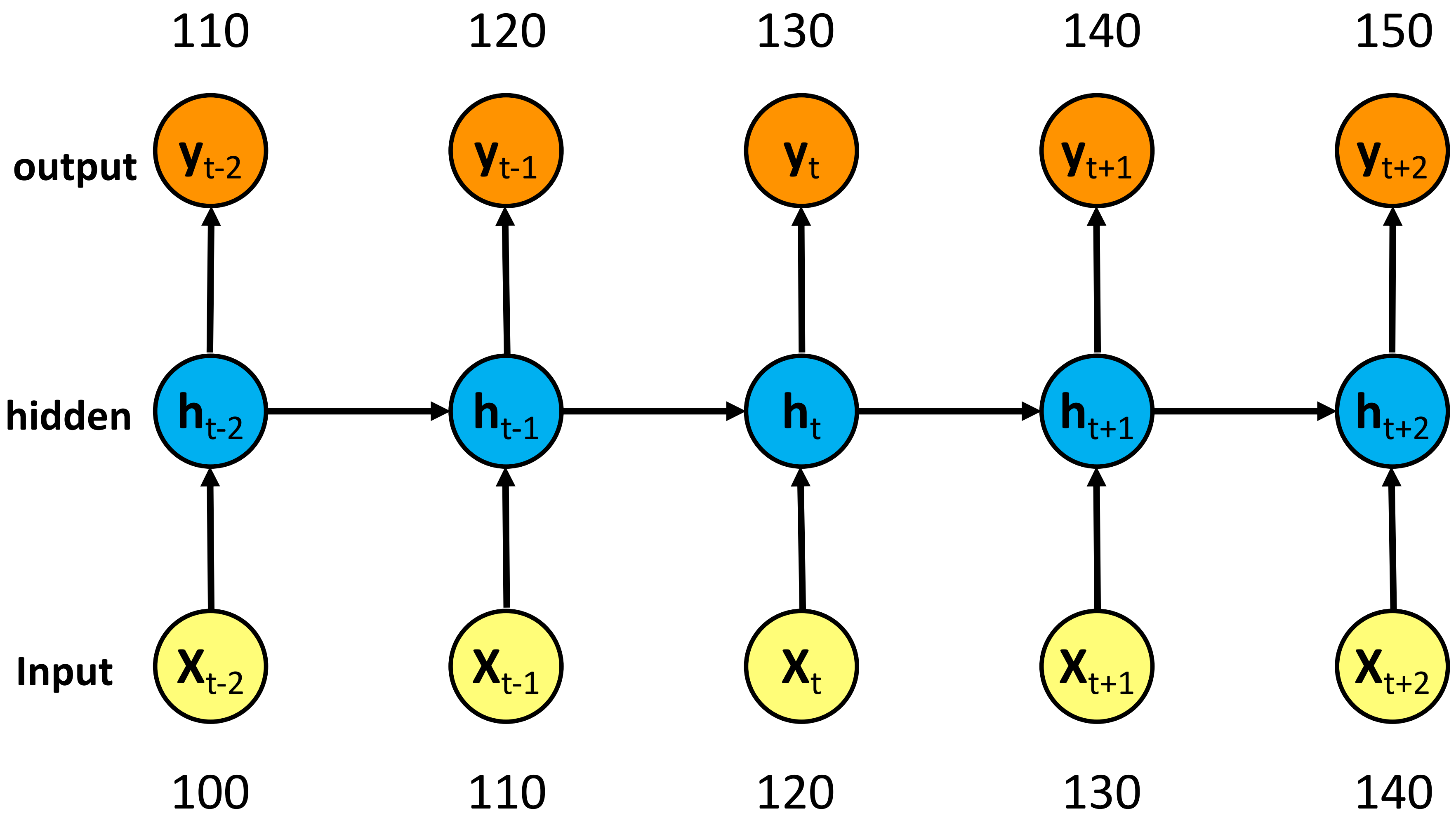


Source: https://leonardoaraujosantos.gitbooks.io/artificial-inteligence/content/deep_learning.html

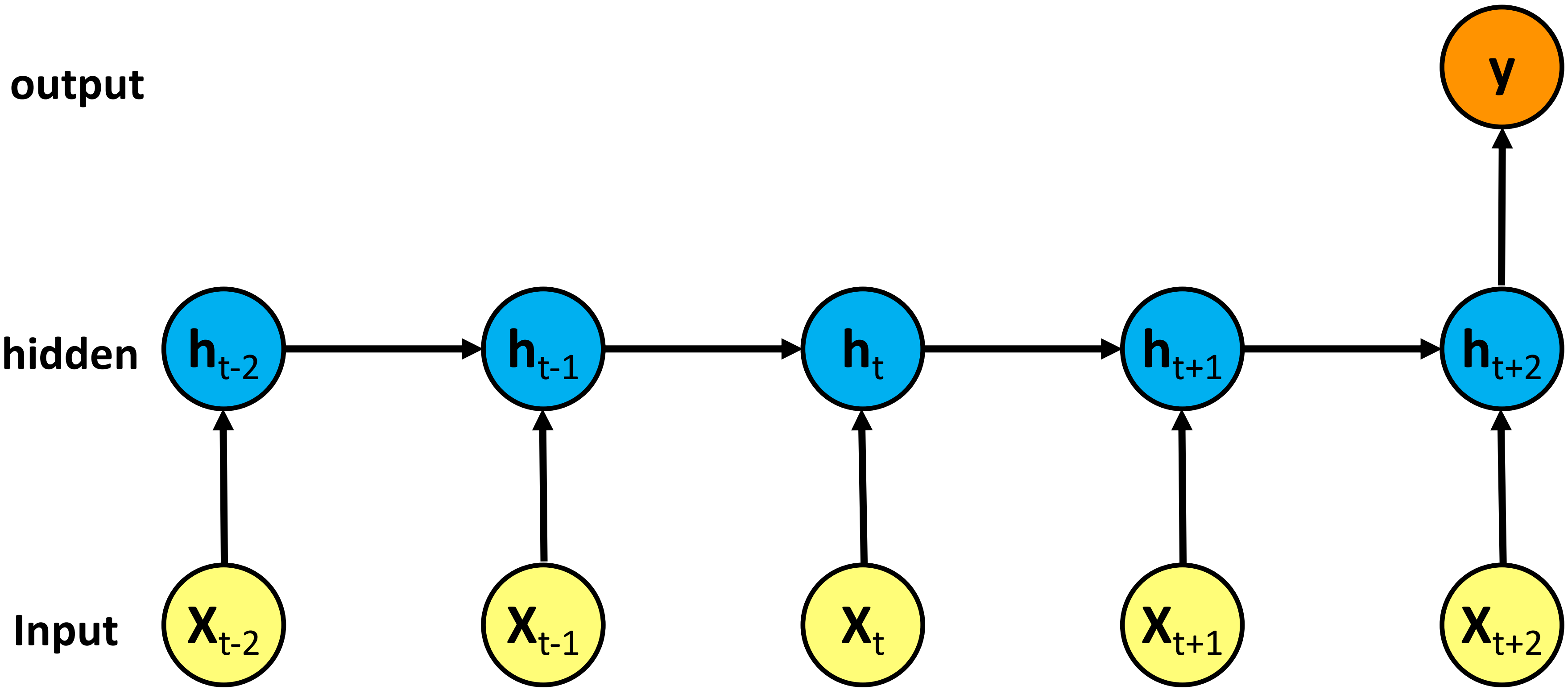
Recurrent Neural Networks (RNN)



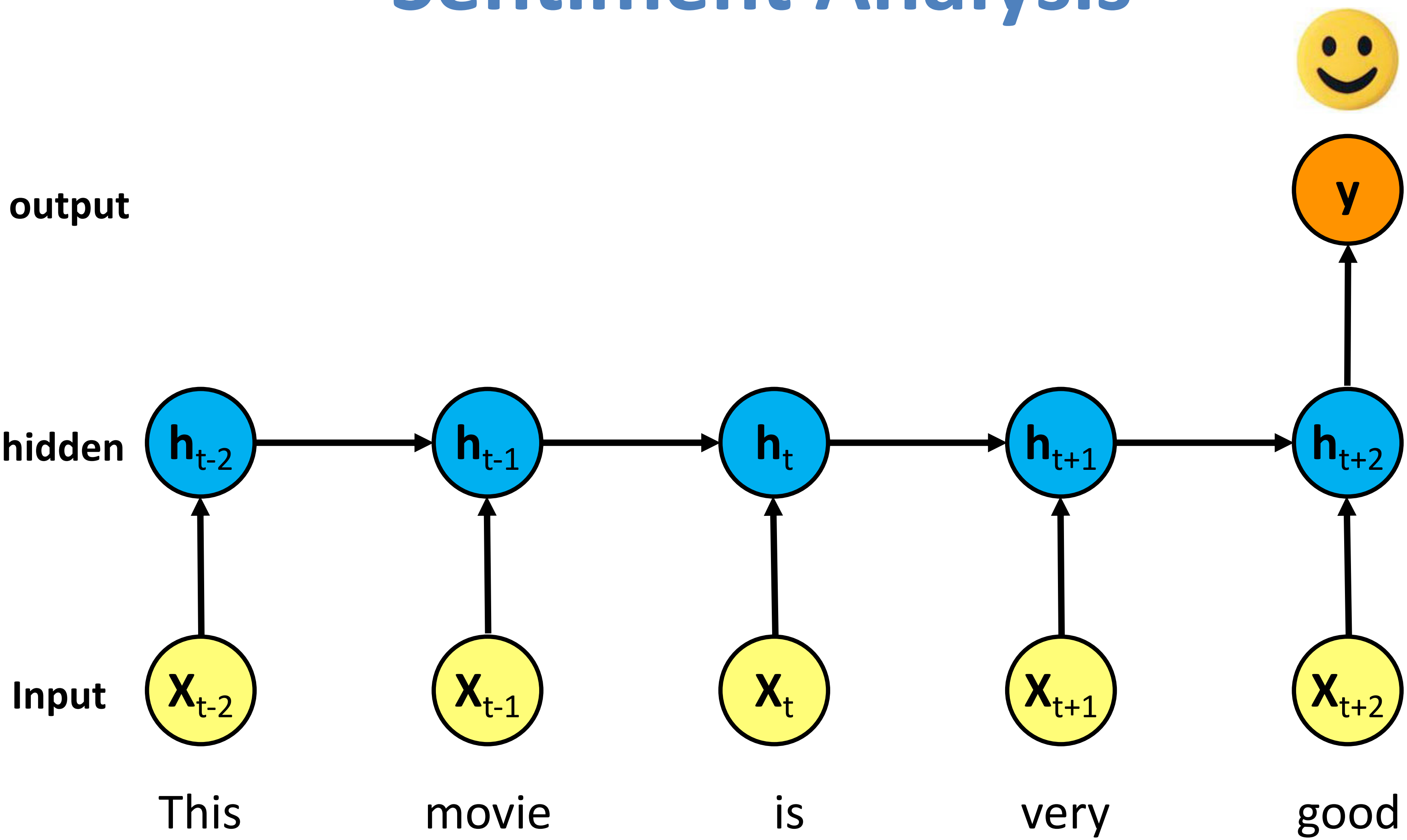
Recurrent Neural Networks (RNN) Time Series Forecasting



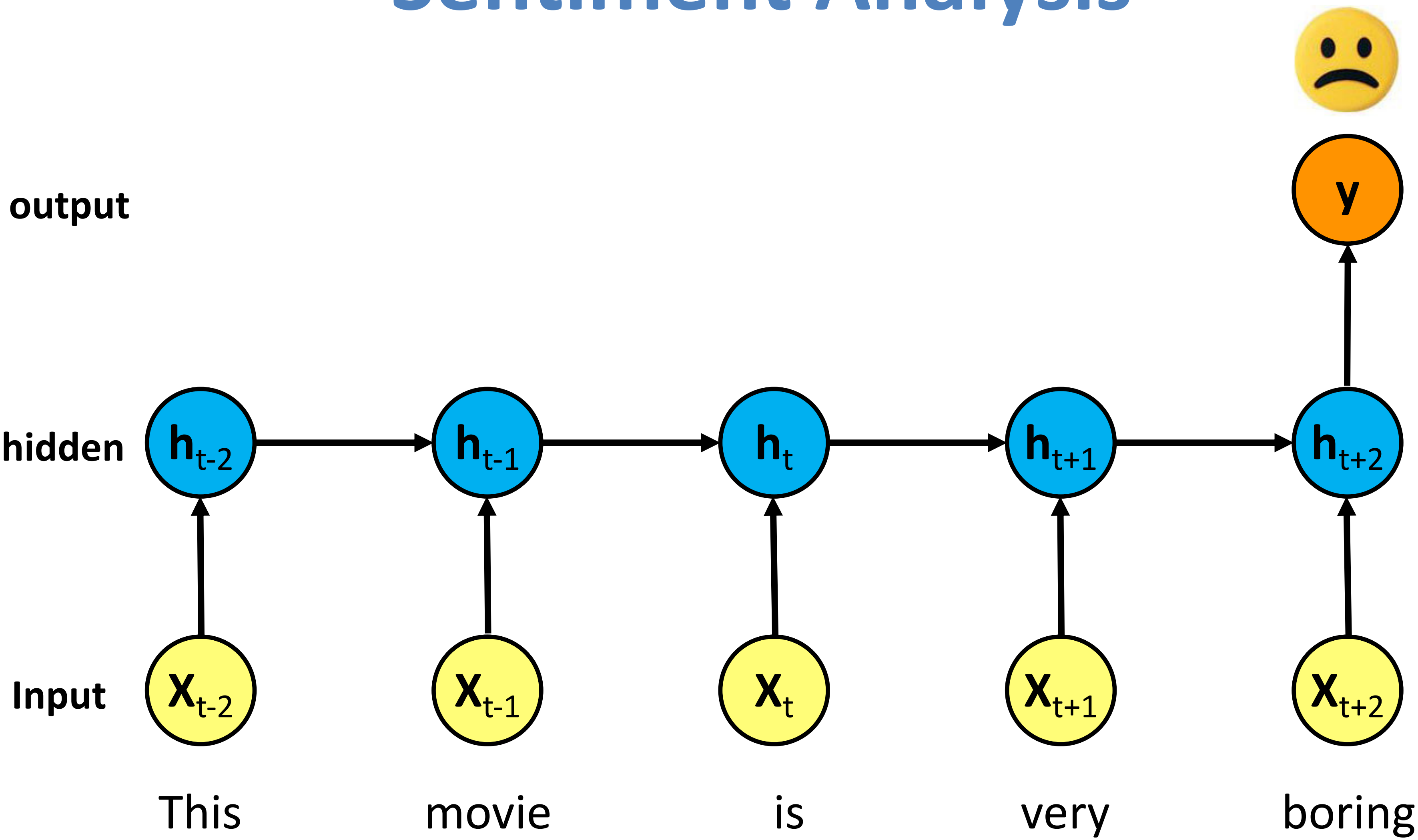
Recurrent Neural Networks (RNN)



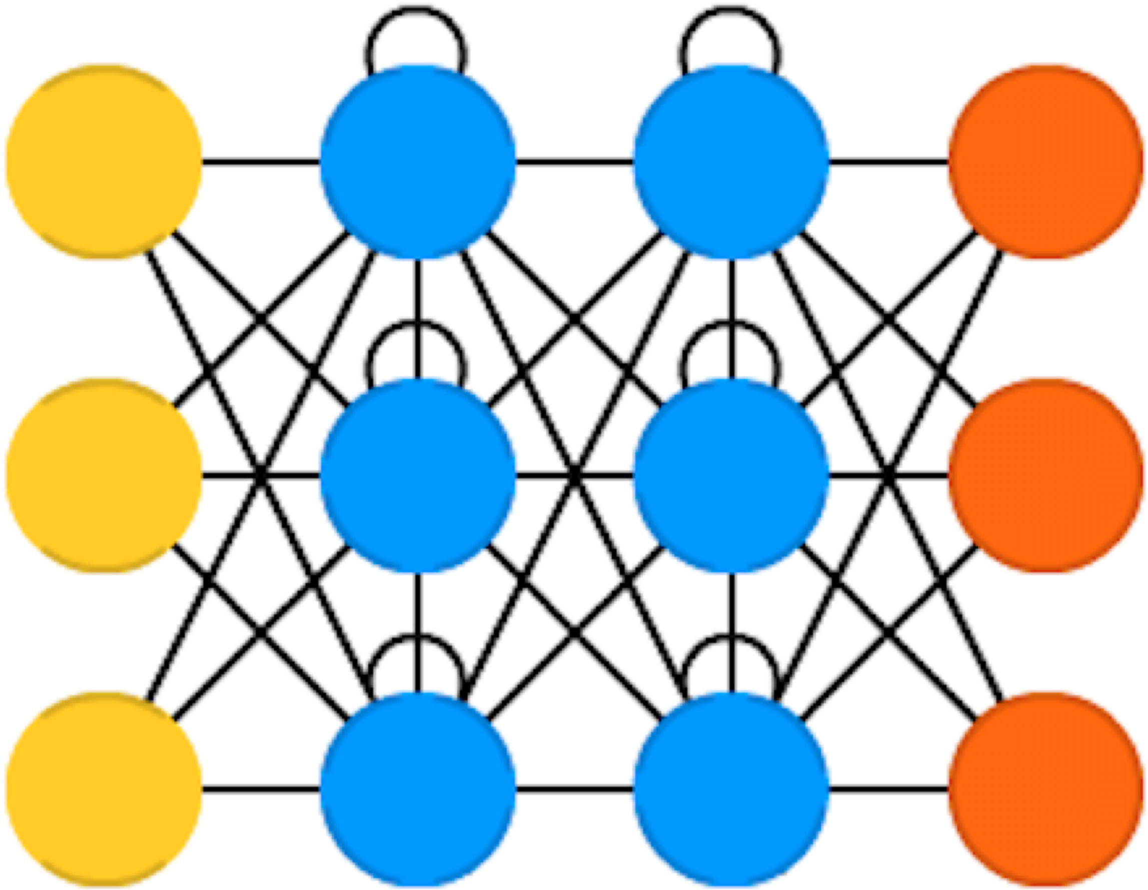
Recurrent Neural Networks (RNN) Sentiment Analysis



Recurrent Neural Networks (RNN)
Sentiment Analysis

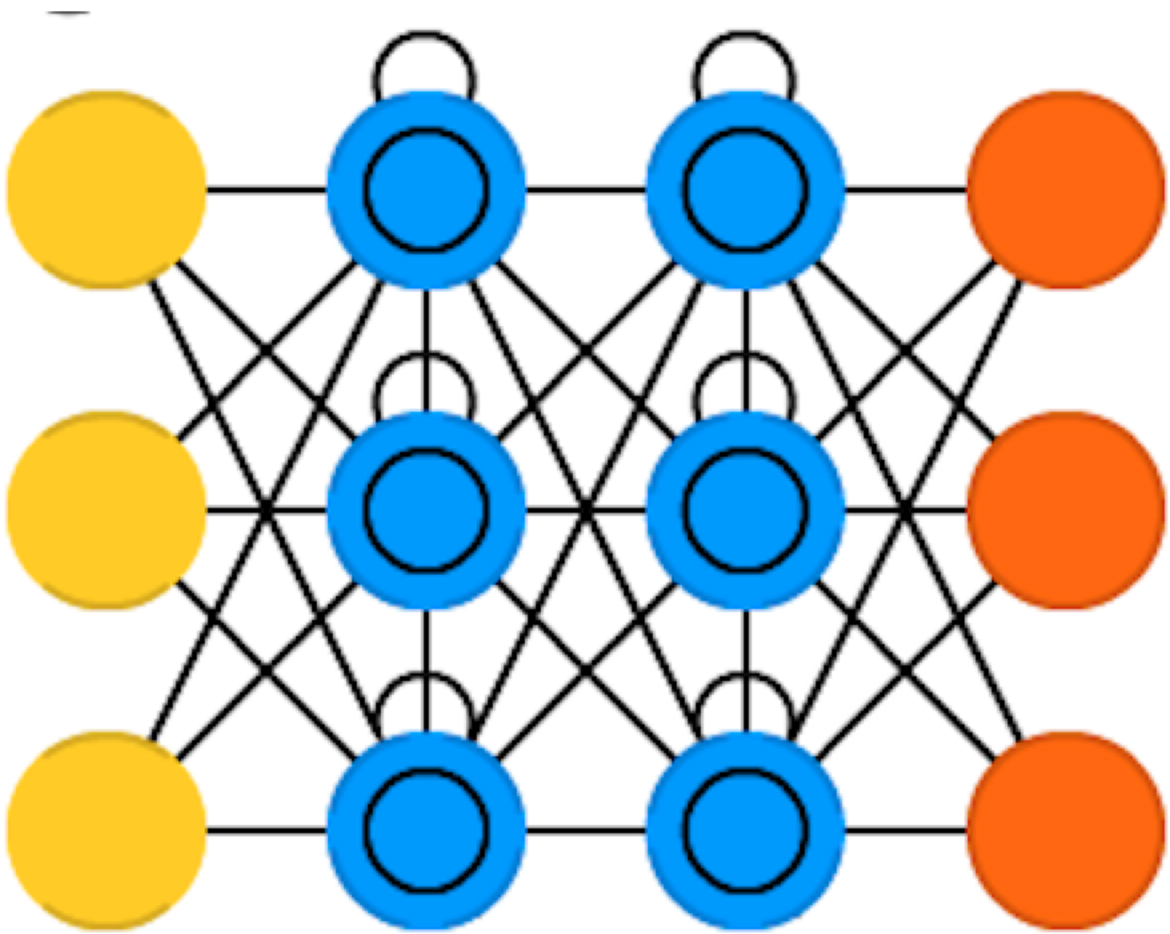


Recurrent Neural Networks (RNN)



Elman, Jeffrey L. "Finding structure in time." Cognitive science 14.2 (1990): 179-211
Source: http://www.asimovinstitute.org/neural-network-zoo/

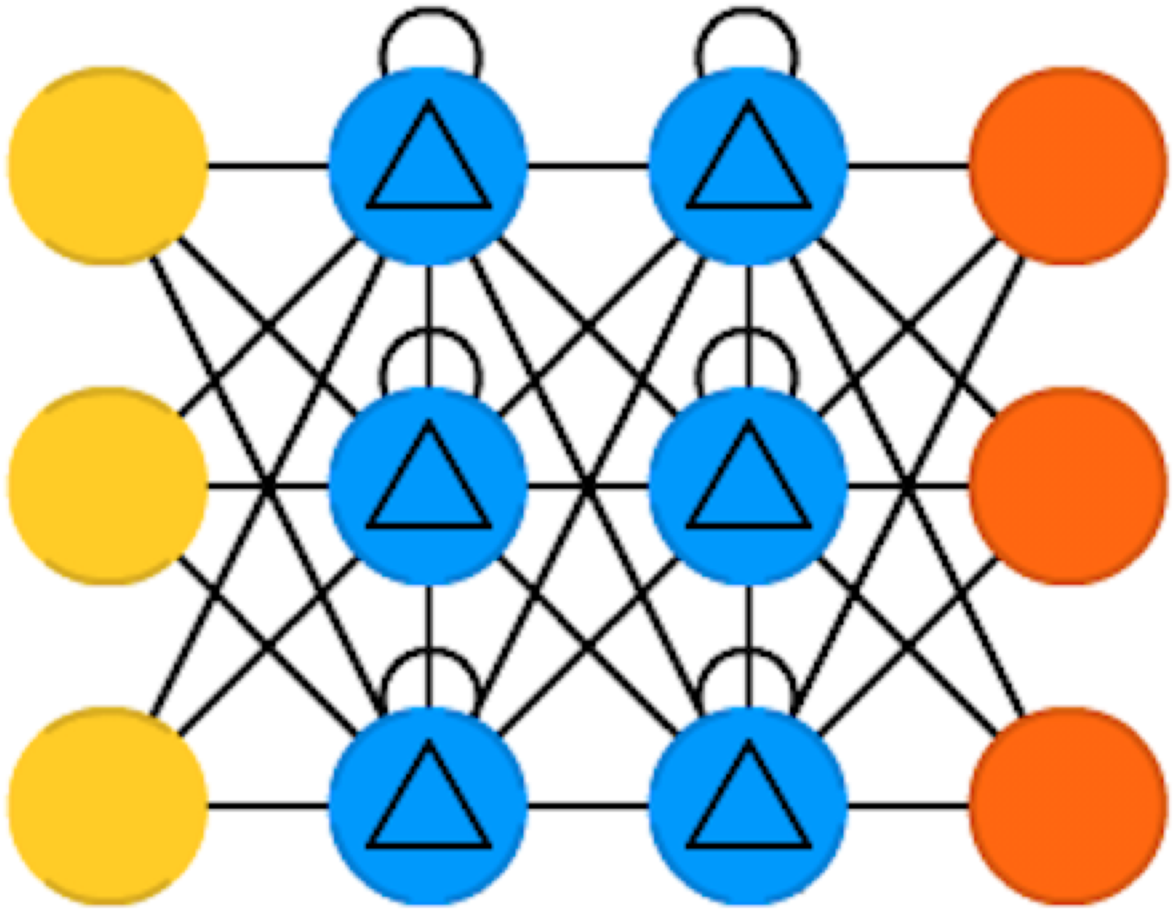
Long / Short Term Memory (LSTM)



Hochreiter, Sepp, and Jürgen Schmidhuber. “Long short-term memory.” Neural computation 9.8 (1997): 1735-1780.

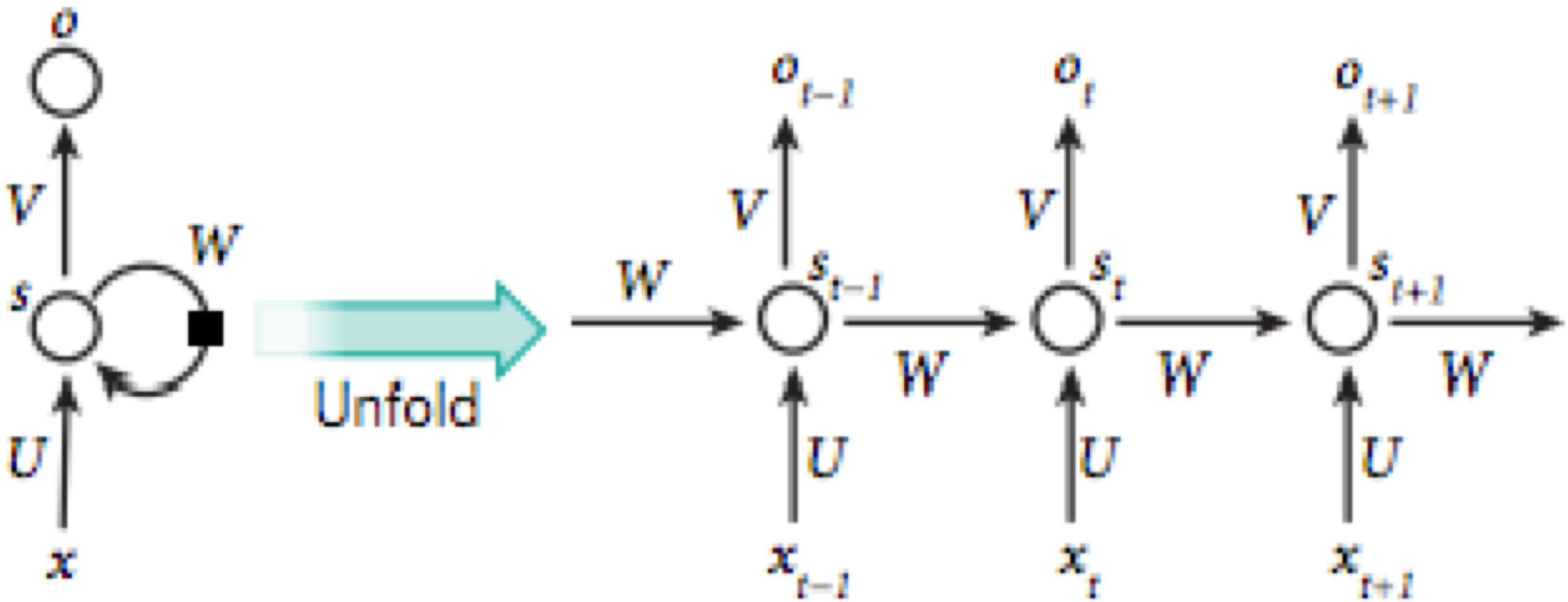
Source: http://www.asimovinstitute.org/neural-network-zoo/

Gated Recurrent Units (GRU)



Chung, Junyoung, et al. "Empirical evaluation of gated recurrent neural networks on sequence modeling." arXiv preprint arXiv:1412.3555 (2014).
Source: http://www.asimovinstitute.org/neural-network-zoo/

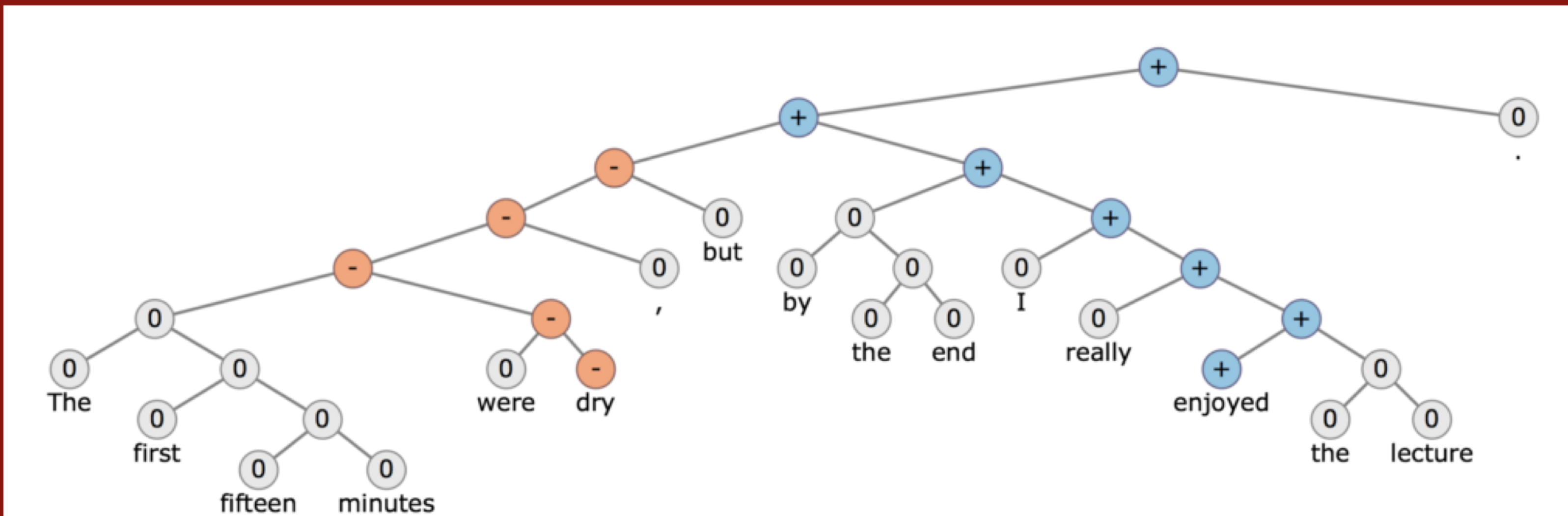
Recurrent Neural Network (RNN)



Source: LeCun, Yann, Yoshua Bengio, and Geoffrey Hinton. "Deep learning." Nature 521, no. 7553 (2015): 436-444.

CS224d: Deep Learning for Natural Language Processing

CS224d: Deep Learning for Natural Language Processing

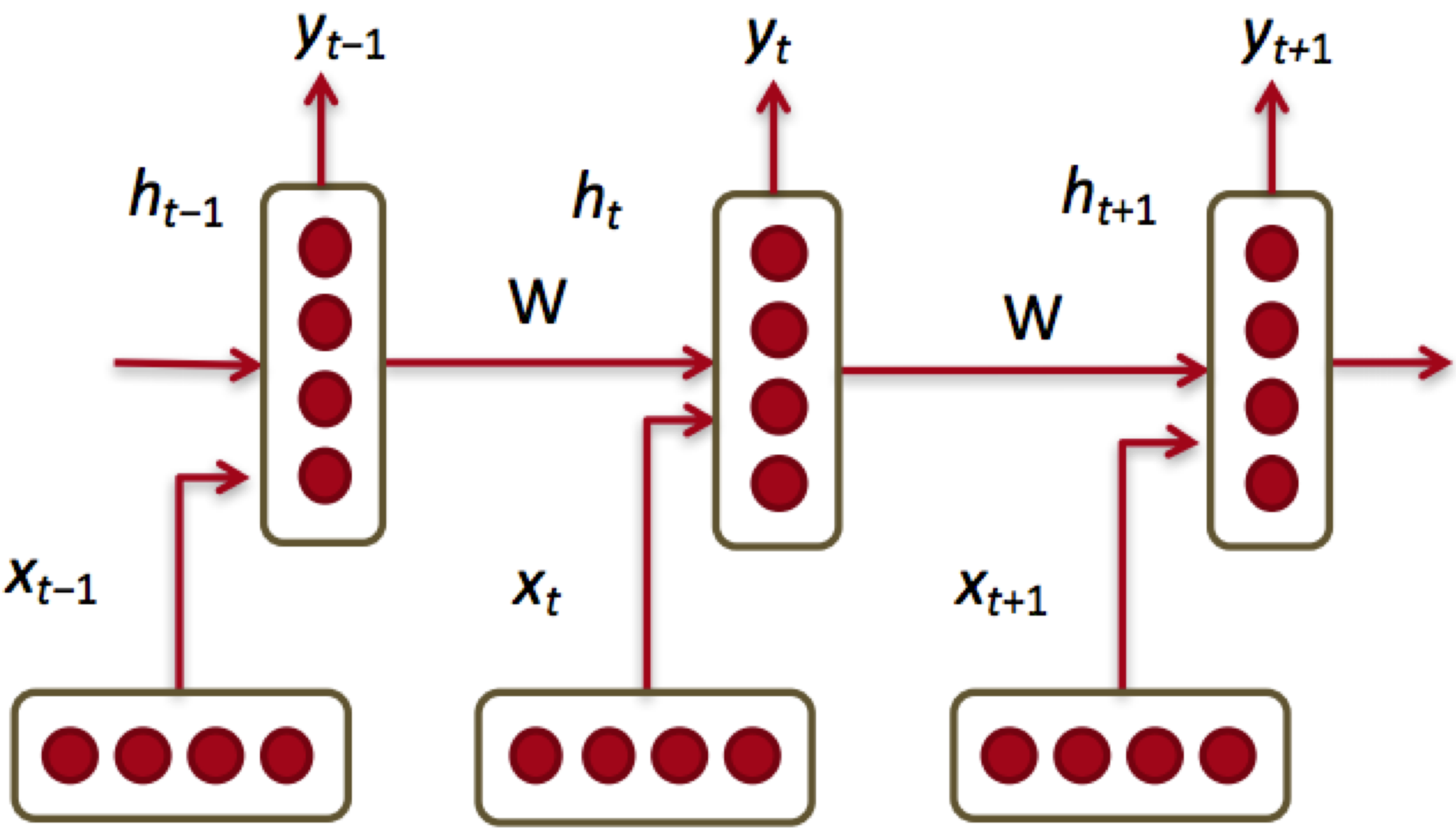


Course Description

Natural language processing (NLP) is one of the most important technologies of the information age. Understanding complex language utterances is also a crucial part of artificial intelligence. Applications of NLP are everywhere because people communicate most everything in language: web search, advertisement, emails, customer service, language translation, radiology reports, etc. There are a large variety of underlying tasks and machine learning models powering NLP applications. Recently, deep learning approaches have obtained very high performance across many different NLP tasks. These models can often be trained with a single end-to-end model and do not require traditional, task-specific feature engineering. In this spring quarter course students will learn to implement, train, debug, visualize and invent their own neural network models. The course provides a deep excursion into cutting-edge research in deep learning applied to NLP. The final project will involve training a complex recurrent neural network and applying it to a large scale NLP problem. On the model side we will cover word vector representations,

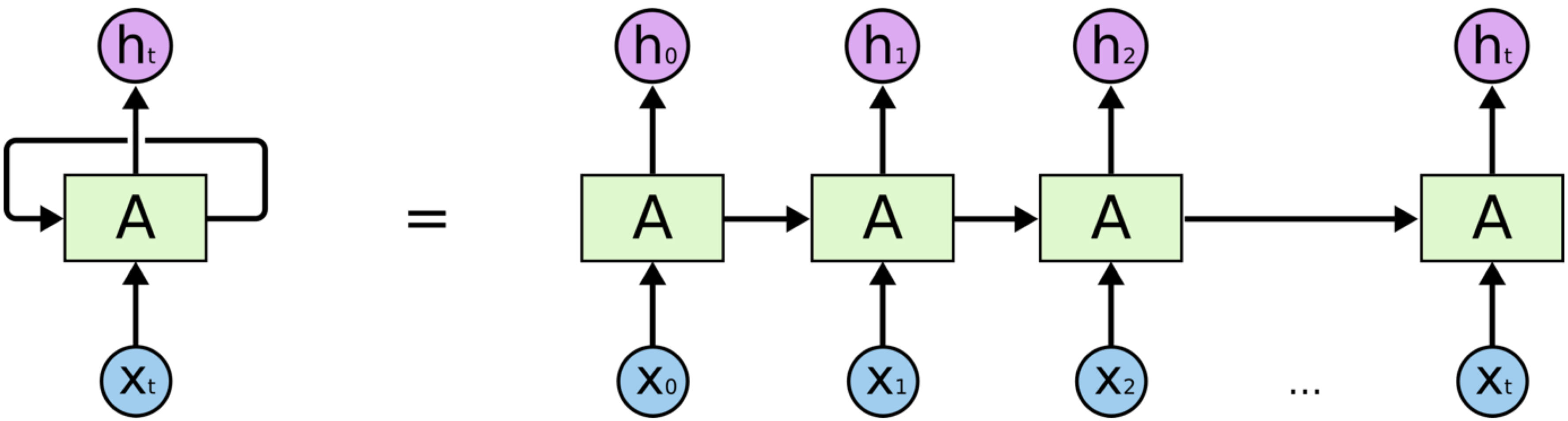
http://cs224d.stanford.edu/

Recurrent Neural Networks (RNNs)



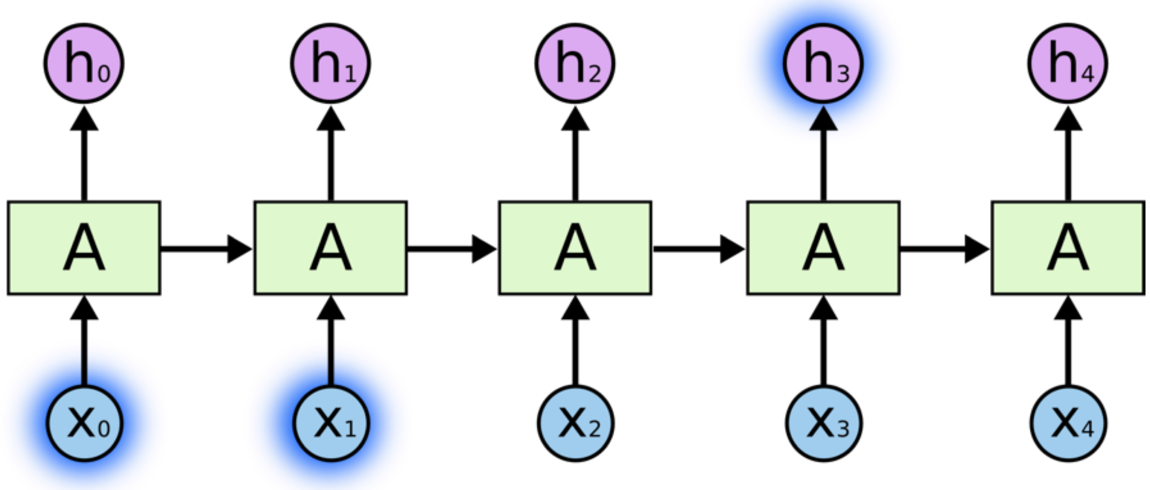
Source: http://cs224d.stanford.edu/lectures/CS224d-Lecture8.pdf

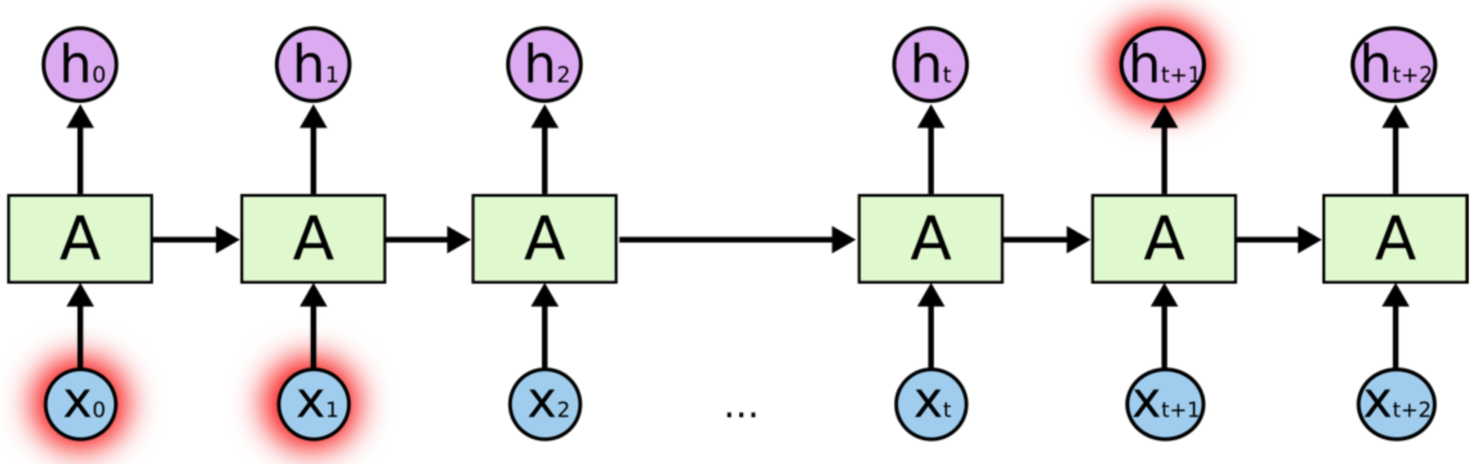
RNN



Source: Christopher Olah, (2015) Understanding LSTM Networks, http://colah.github.io/posts/2015-08-Understanding-LSTMs/

RNN long-term dependencies

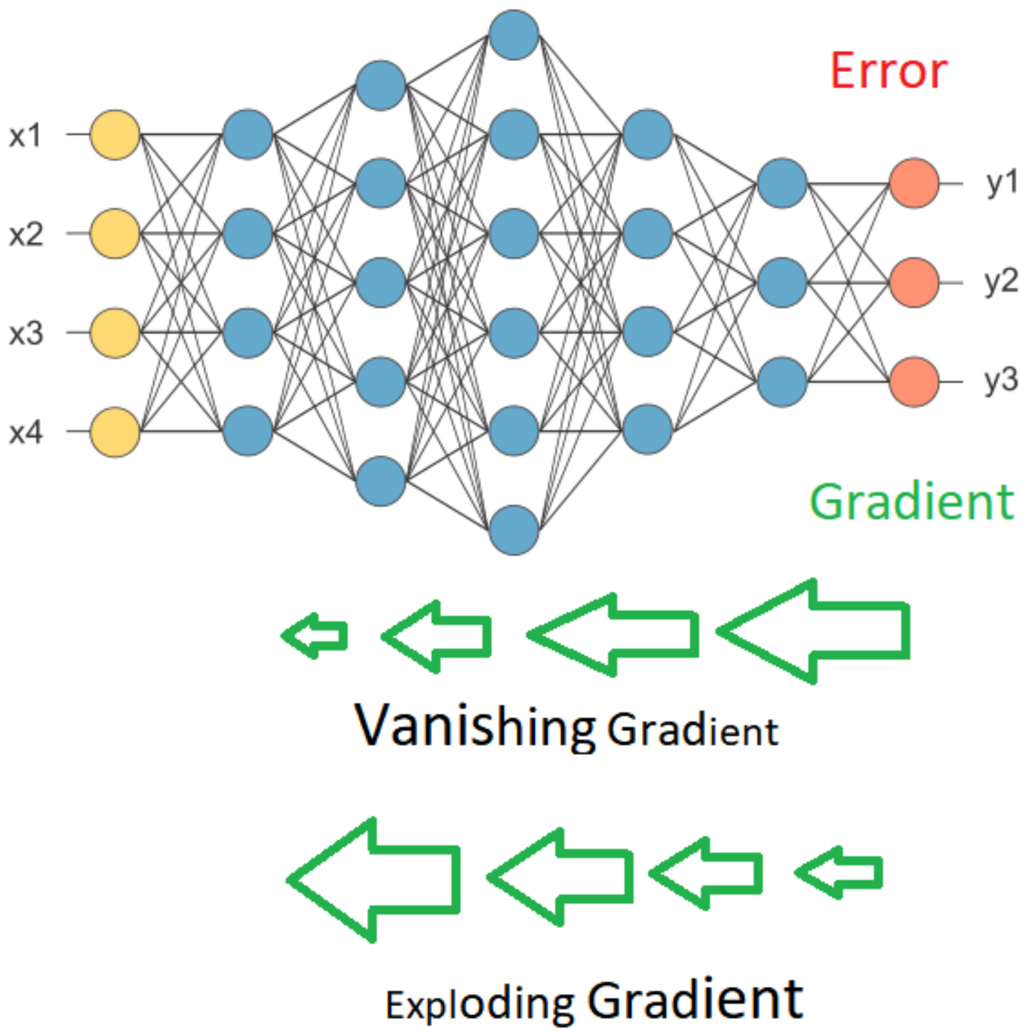




I grew up in France… I speak fluent French.

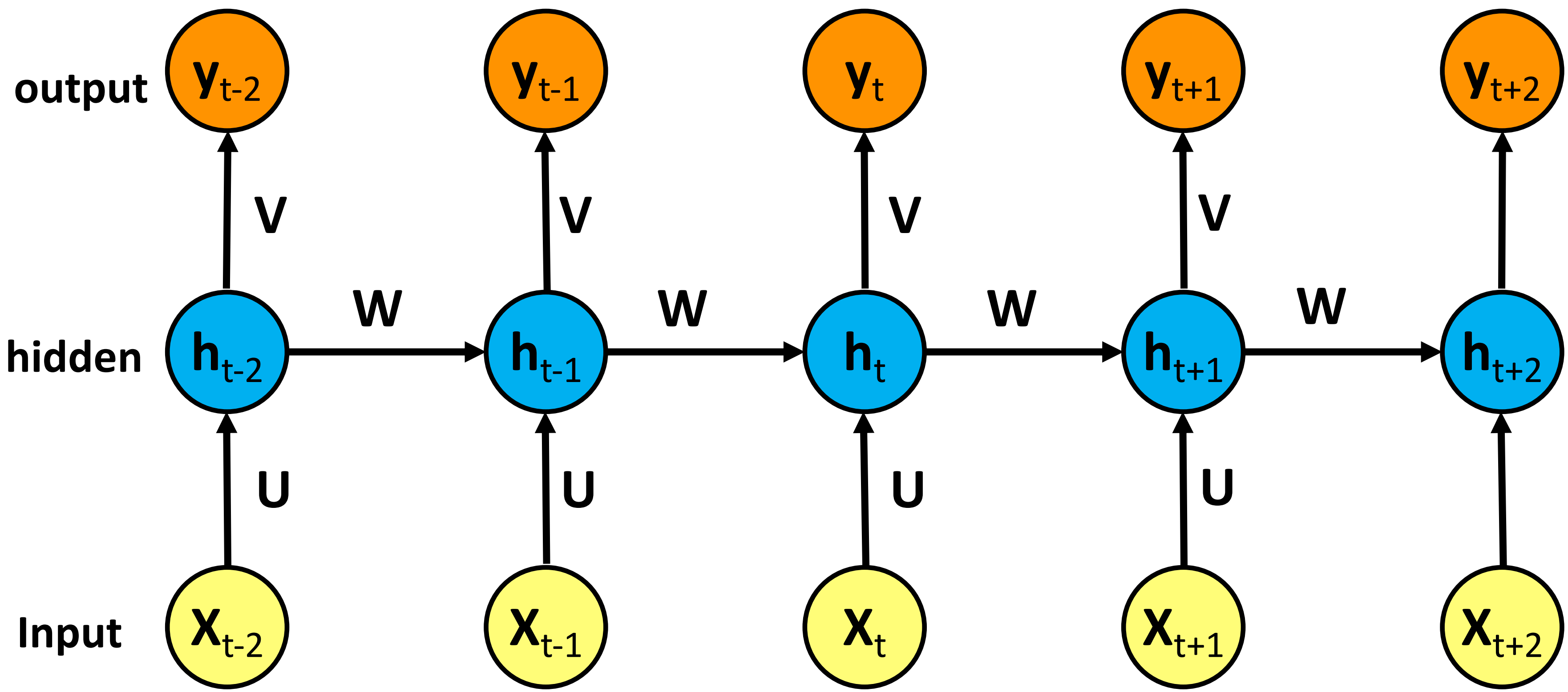
Source: Christopher Olah, (2015) Understanding LSTM Networks, http://colah.github.io/posts/2015-08-Understanding-LSTMs/

Vanishing Gradient
Exploding Gradient

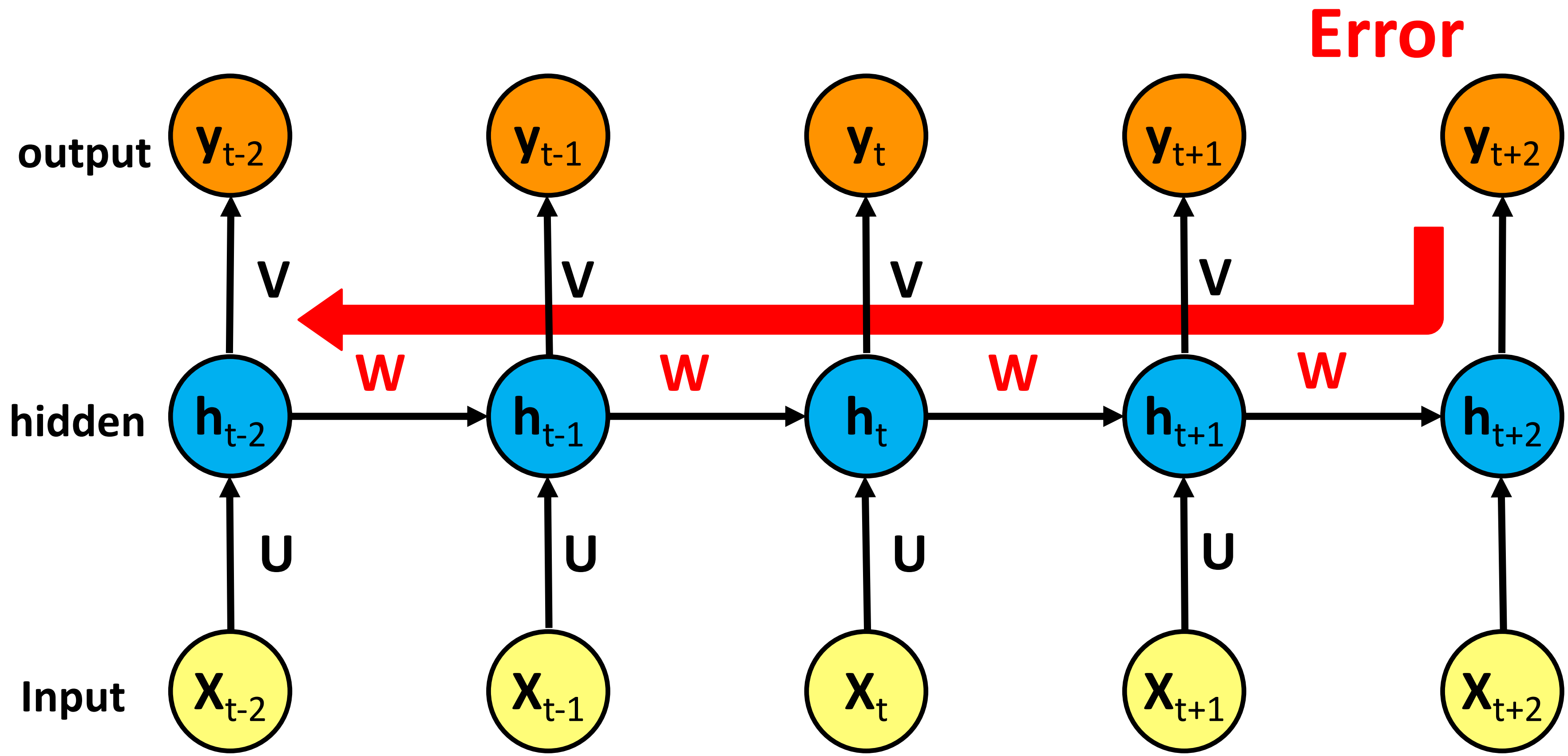


Source: https://medium.com/deep-math-machine-learning-ai/chapter-10-1-deepnlp-lstm-long-short-term-memory-networks-with-math-21477f8e4235

Recurrent Neural Networks (RNN)



RNN
Vanishing Gradient problem
Exploding Gradient problem

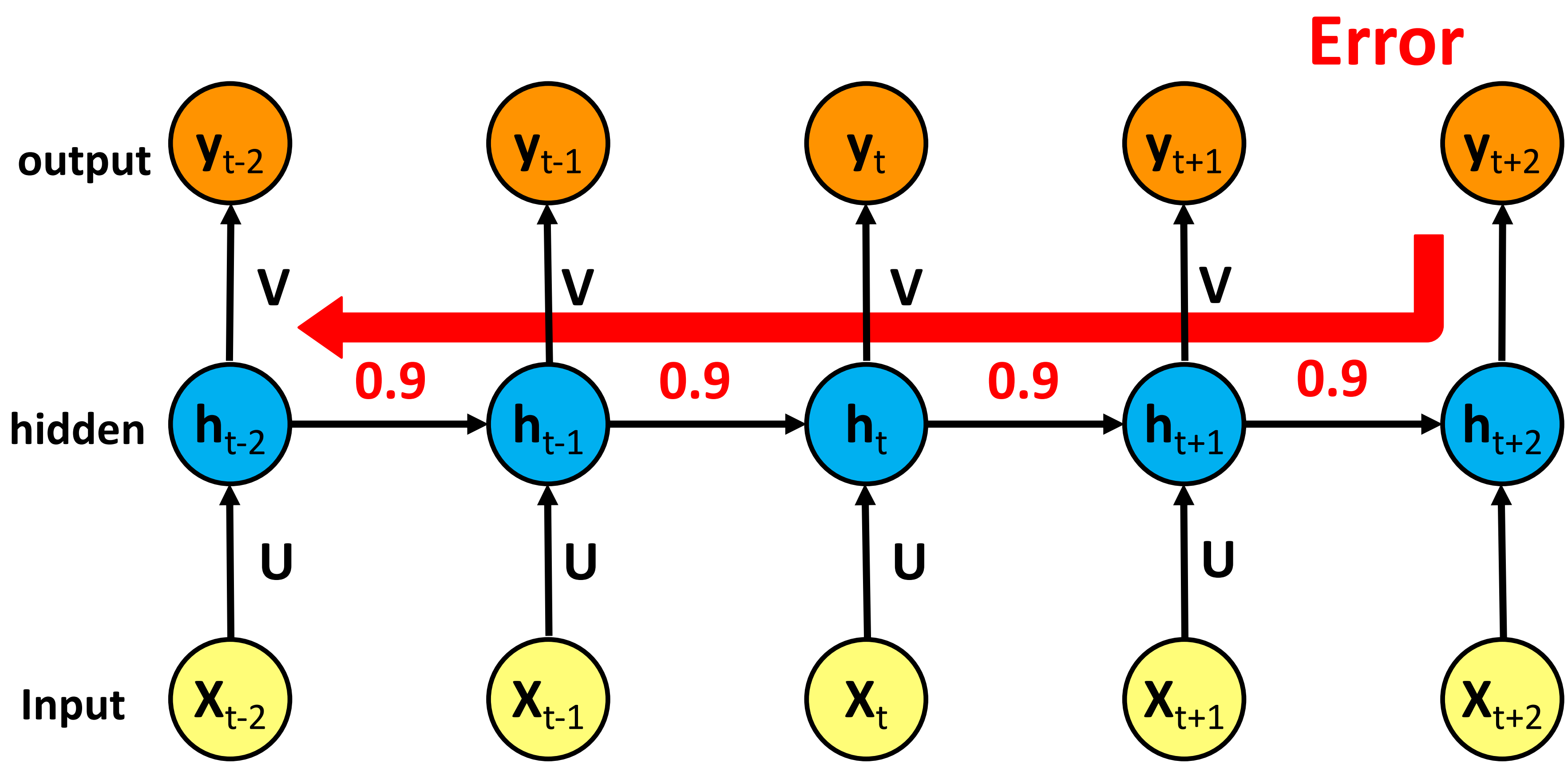


if $|W| < 1$ (Vanishing)

if $|W| > 1$ (Exploding)

Source: https://medium.com/deep-math-machine-learning-ai/chapter-10-1-deepnlp-lstm-long-short-term-memory-networks-with-math-21477f8e4235

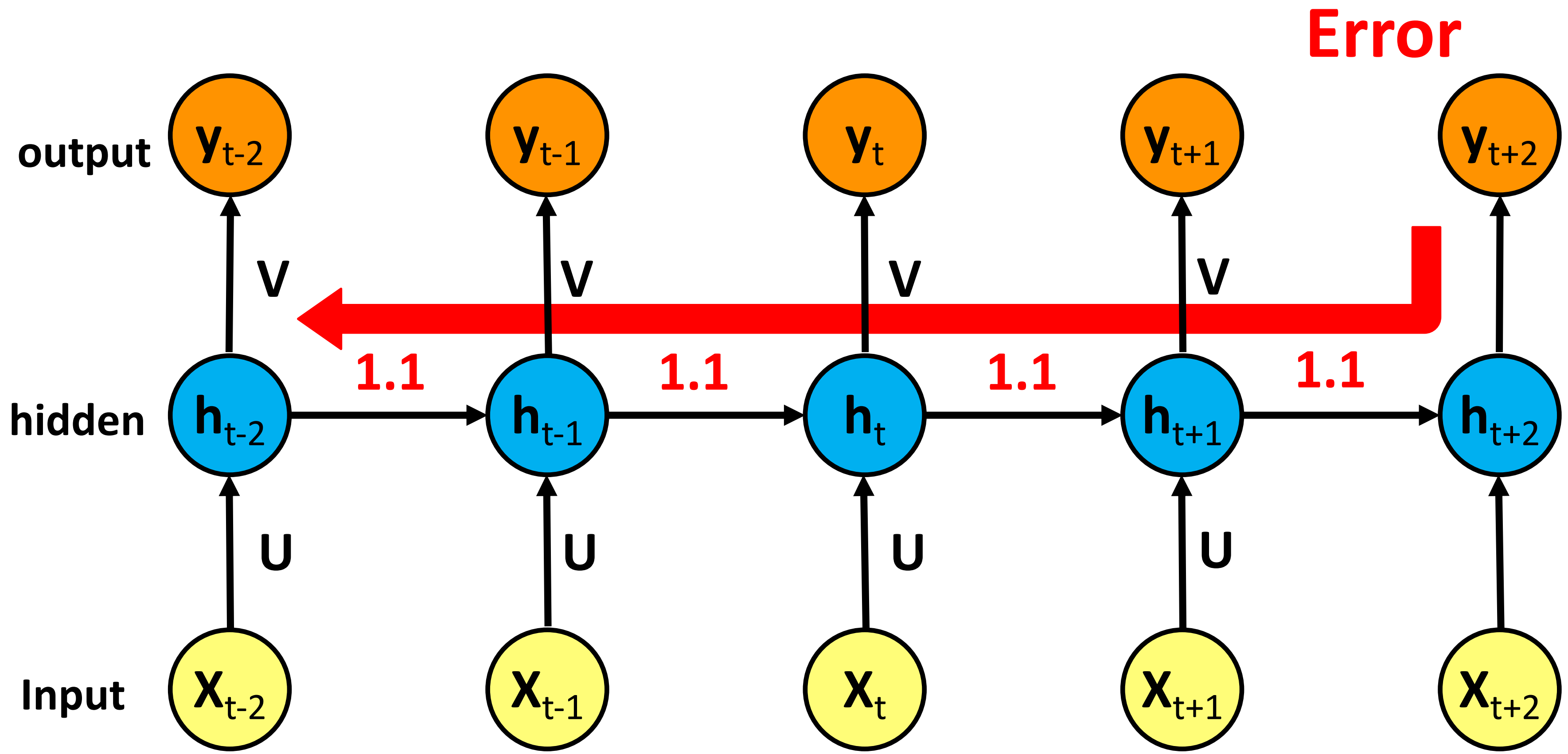
RNN
Vanishing Gradient problem



W = 0.9 < 1 (Vanishing)

Source: https://medium.com/deep-math-machine-learning-ai/chapter-10-1-deepnlp-lstm-long-short-term-memory-networks-with-math-21477f8e4235

RNN
Exploding Gradient problem



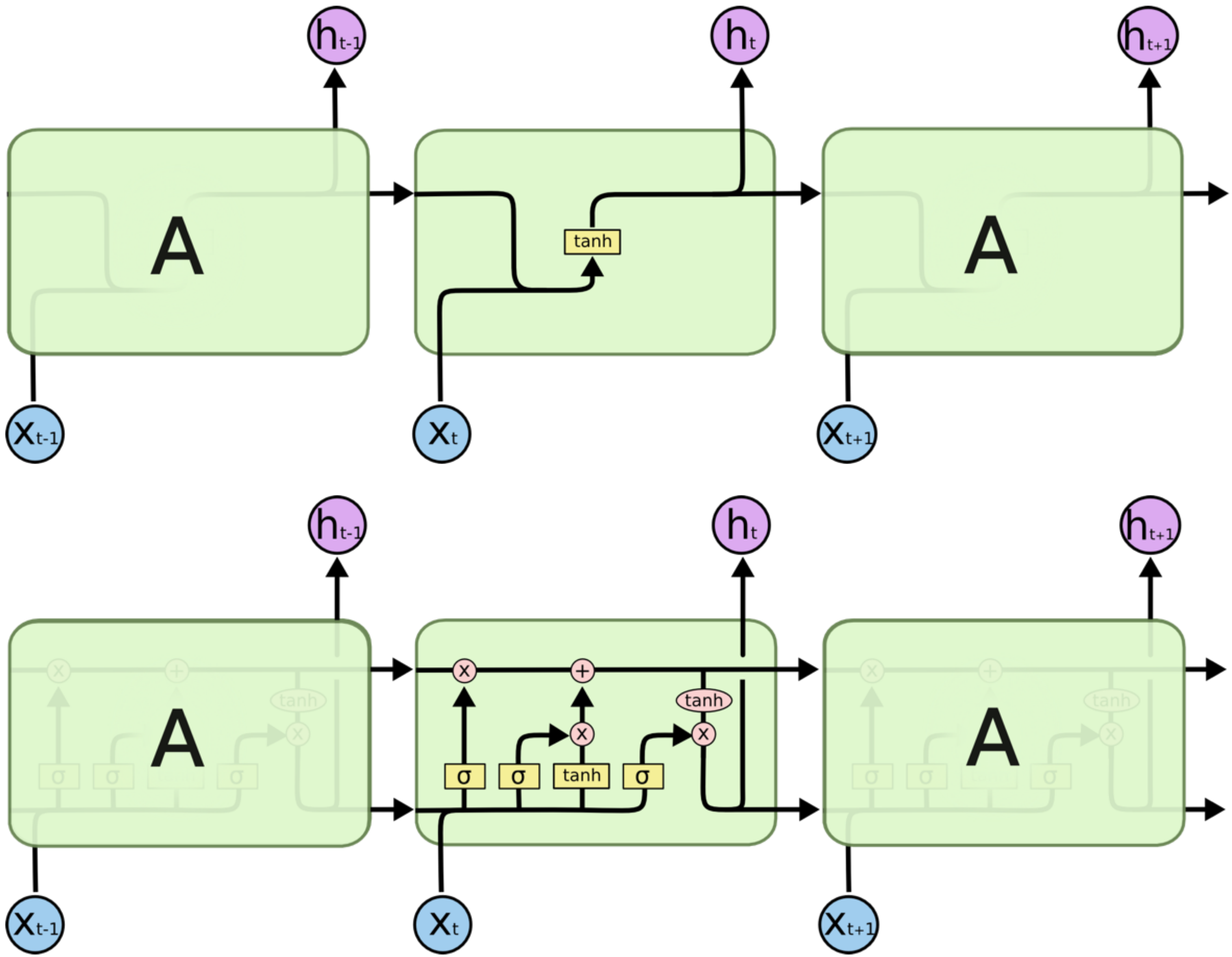
W = 1.1 > 1 (Exploding)

Source: https://medium.com/deep-math-machine-learning-ai/chapter-10-1-deepnlp-lstm-long-short-term-memory-networks-with-math-21477f8e4235

RNN LSTM

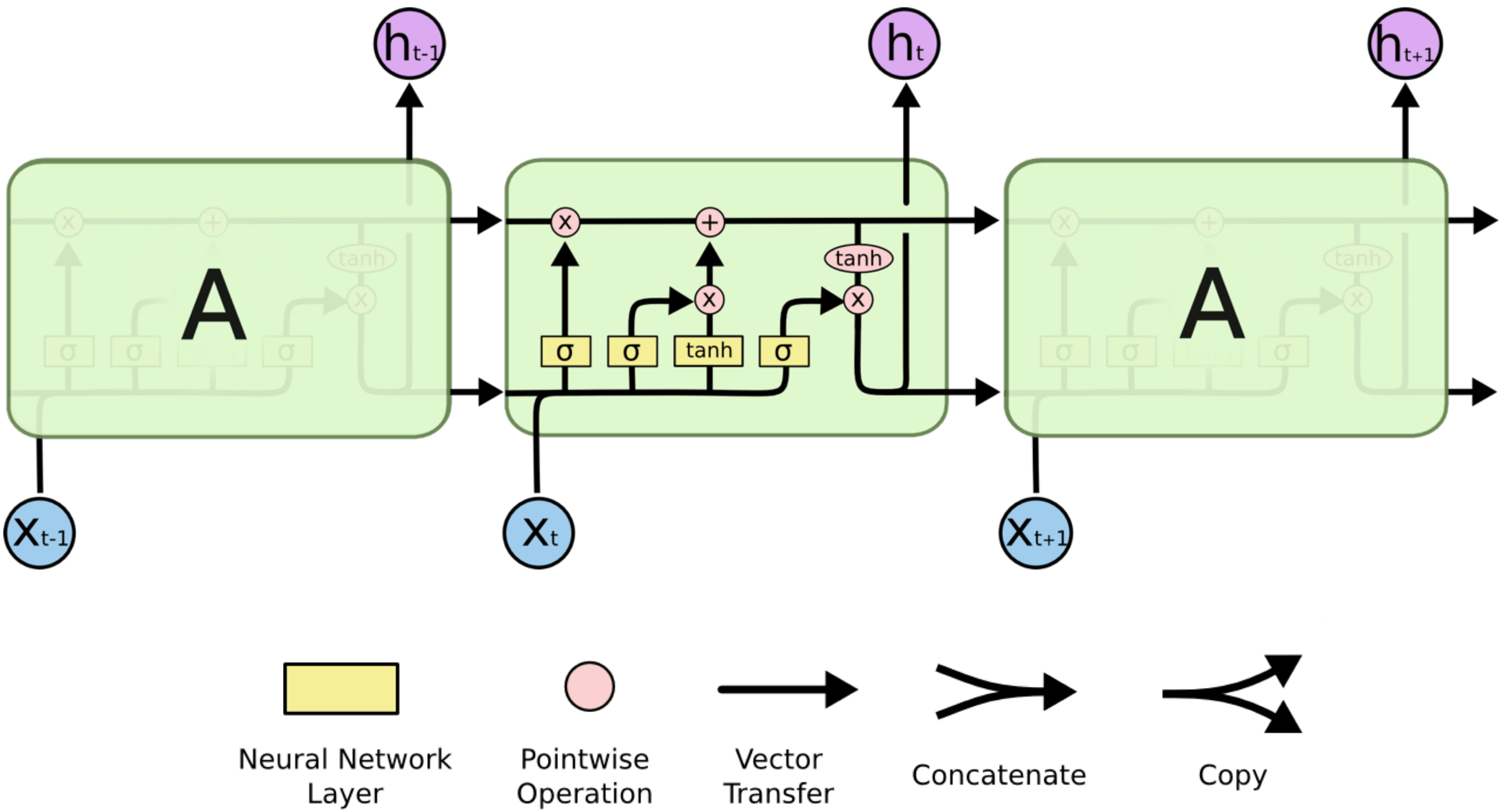
RNN

LSTM



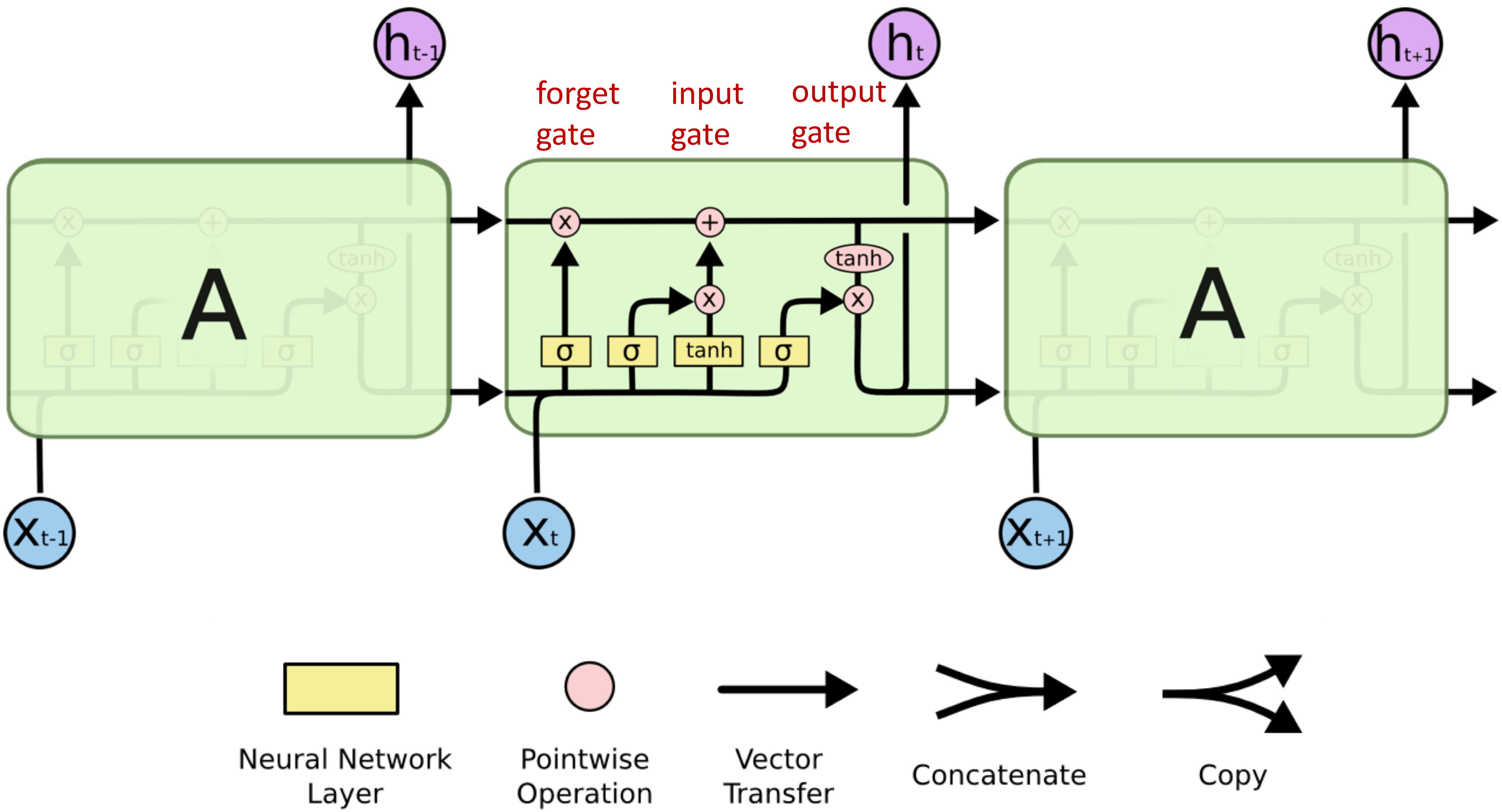
Source: Christopher Olah, (2015) Understanding LSTM Networks, http://colah.github.io/posts/2015-08-Understanding-LSTMs/

Long Short Term Memory (LSTM)



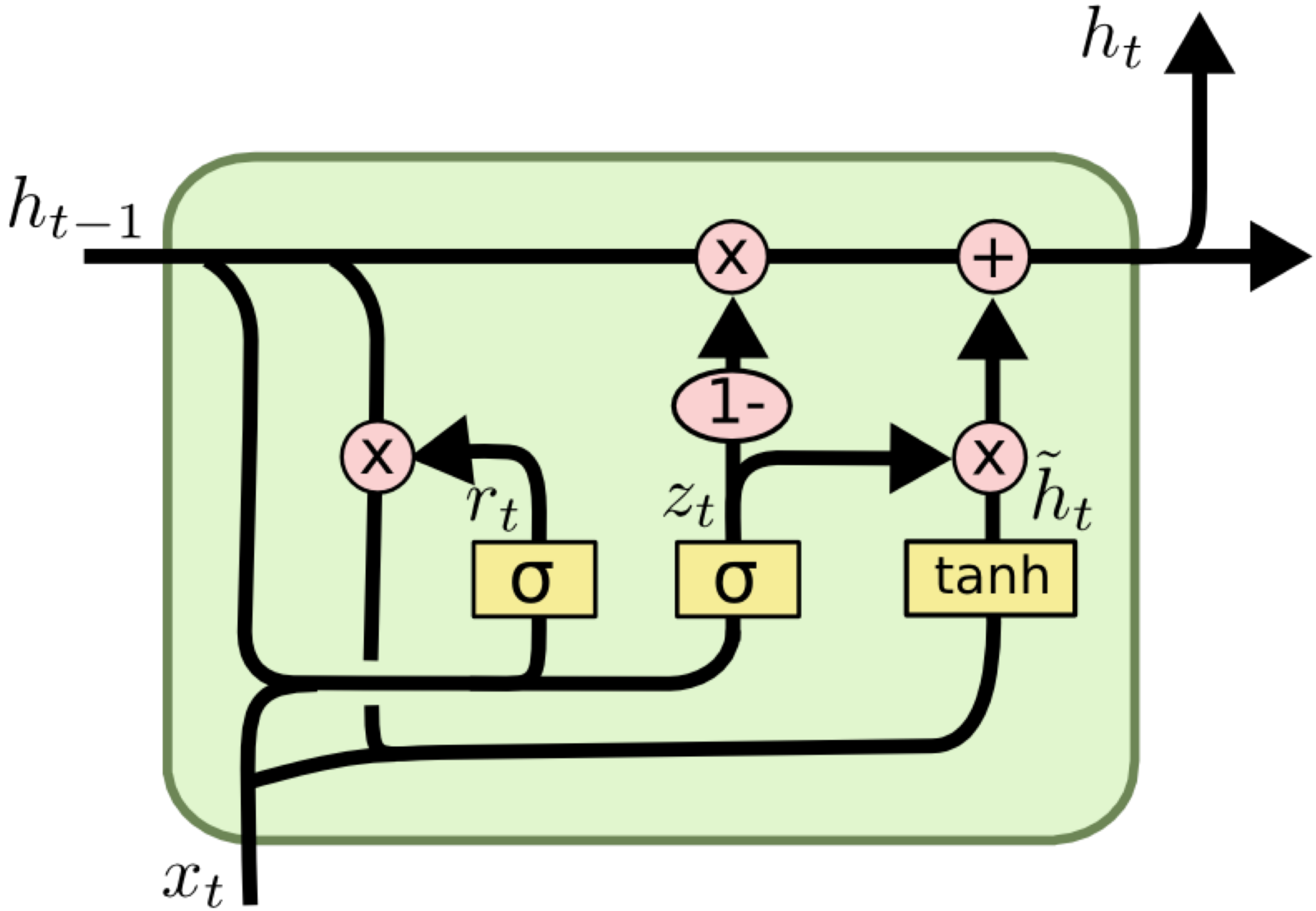
Source: Christopher Olah, (2015) Understanding LSTM Networks, http://colah.github.io/posts/2015-08-Understanding-LSTMs/

Long Short Term Memory (LSTM)



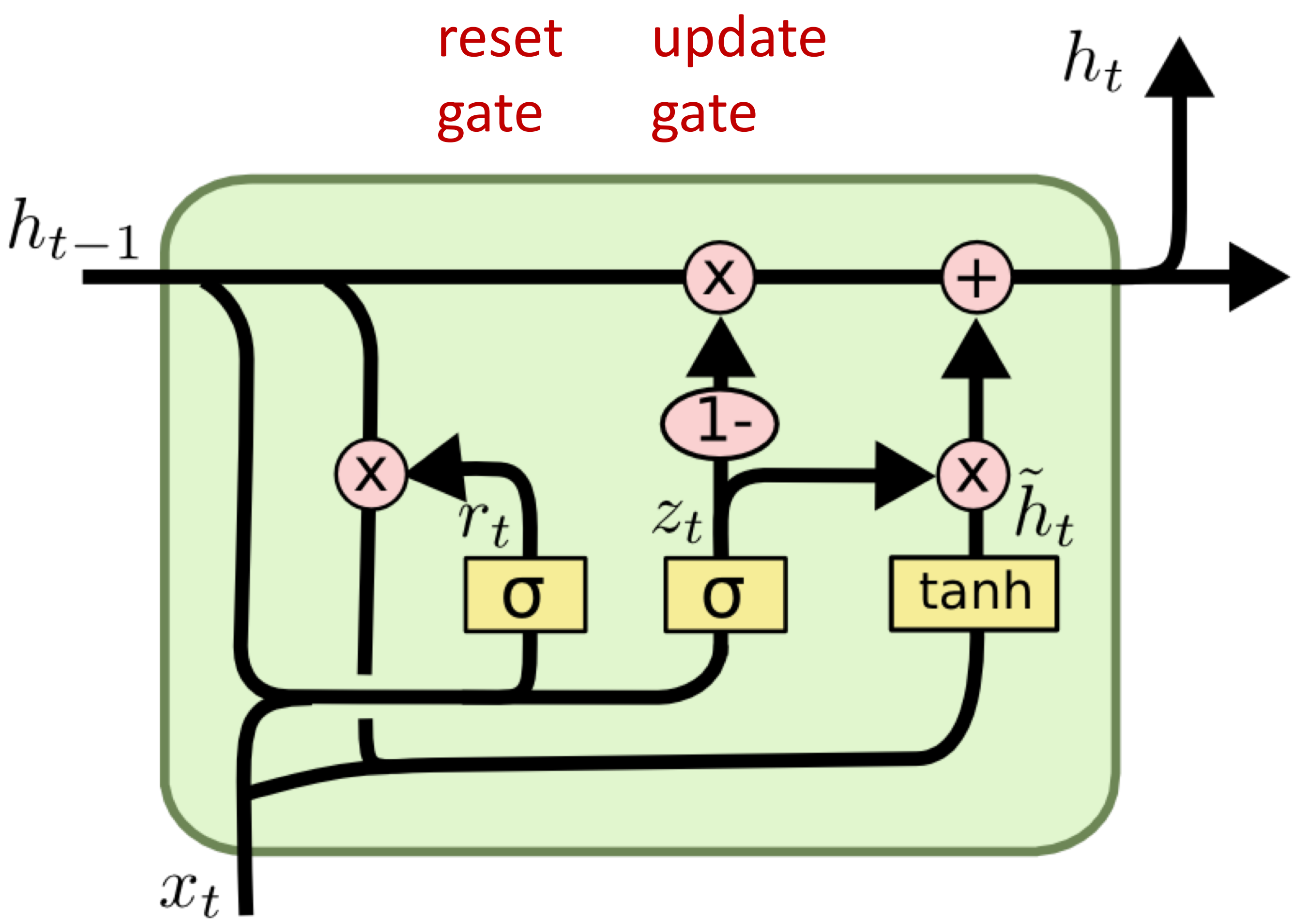
Source: Christopher Olah, (2015) Understanding LSTM Networks, http://colah.github.io/posts/2015-08-Understanding-LSTMs/

Gated Recurrent Unit (GRU)



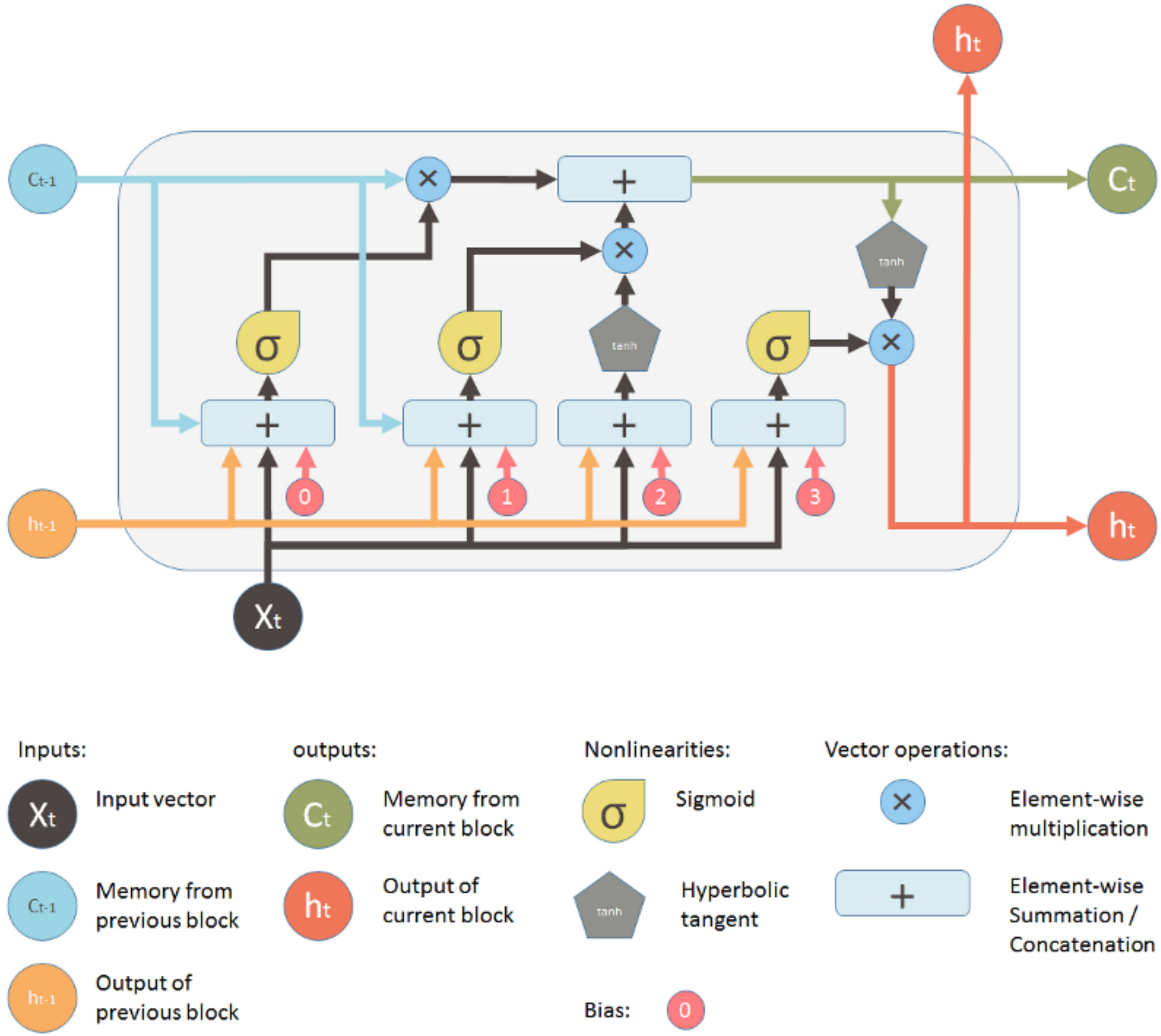
Source: Christopher Olah, (2015) Understanding LSTM Networks, http://colah.github.io/posts/2015-08-Understanding-LSTMs/

Gated Recurrent Unit (GRU)



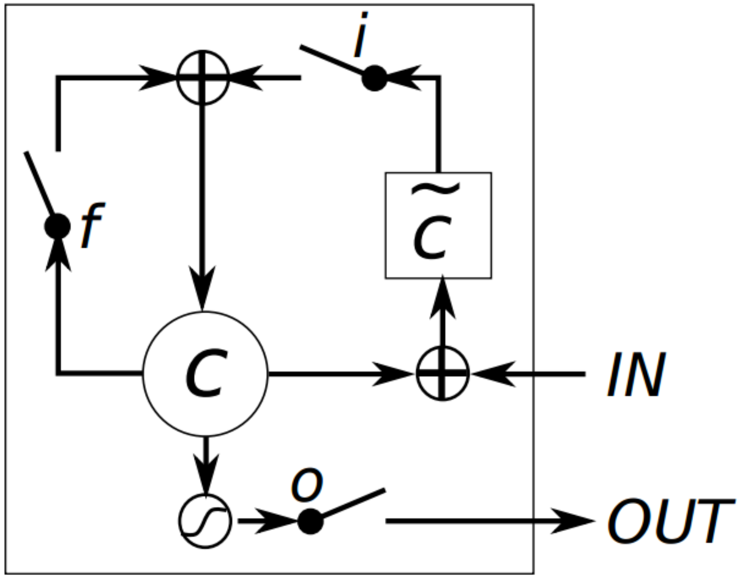
Source: Christopher Olah, (2015) Understanding LSTM Networks, http://colah.github.io/posts/2015-08-Understanding-LSTMs/

LSTM



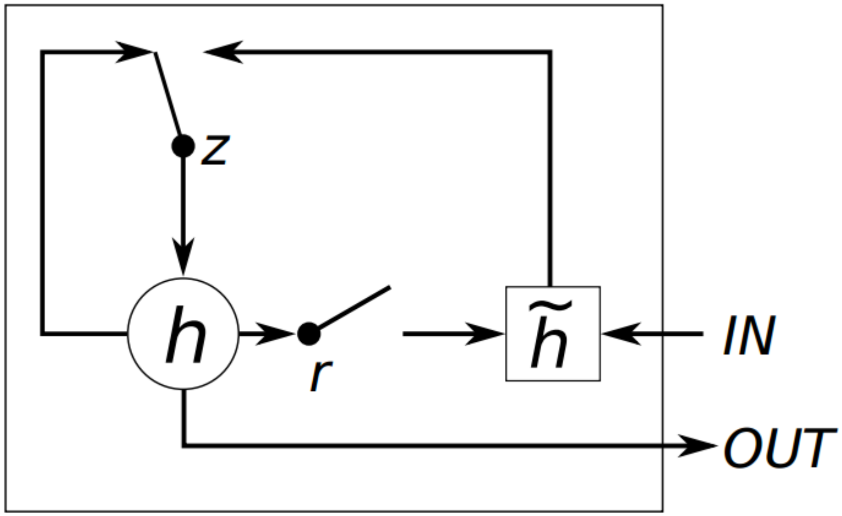
Source: Shi Yan (2016), Understanding LSTM and its diagrams, https://medium.com/mlreview/understanding-lstm-and-its-diagrams-37e2f46f1714

LSTM vs GRU



LSTM

i, f and o are the input, forget and output gates, respectively.
c and c˜ denote the memory cell and the new memory cell content.

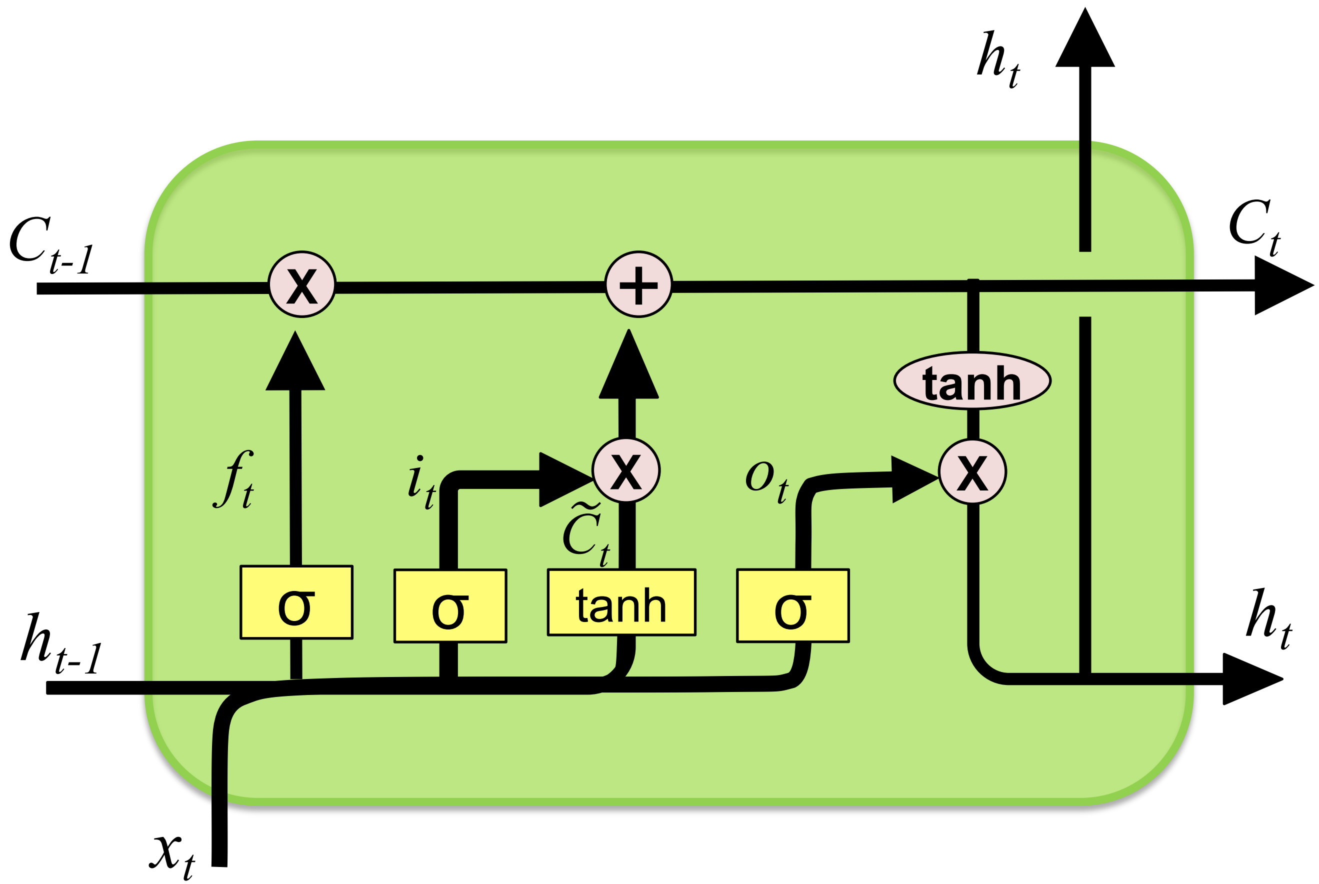


GRU

r and z are the reset and update gates, and h and h˜ are the activation and the candidate activation.

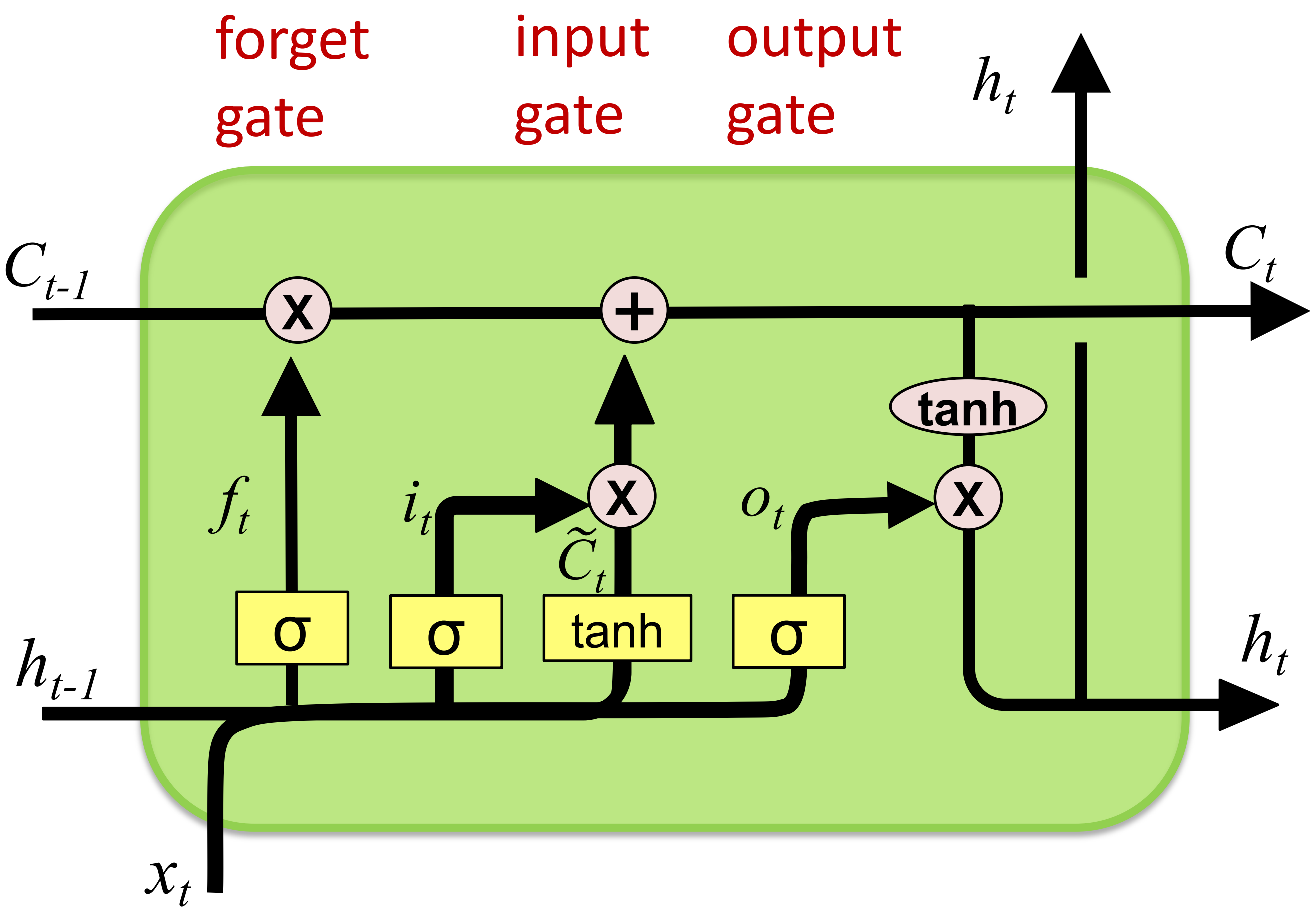
Source: Chung, Junyoung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. "Empirical evaluation of gated recurrent neural networks on sequence modeling." *arXiv preprint arXiv:1412.3555* (2014).

Long Short Term Memory (LSTM)



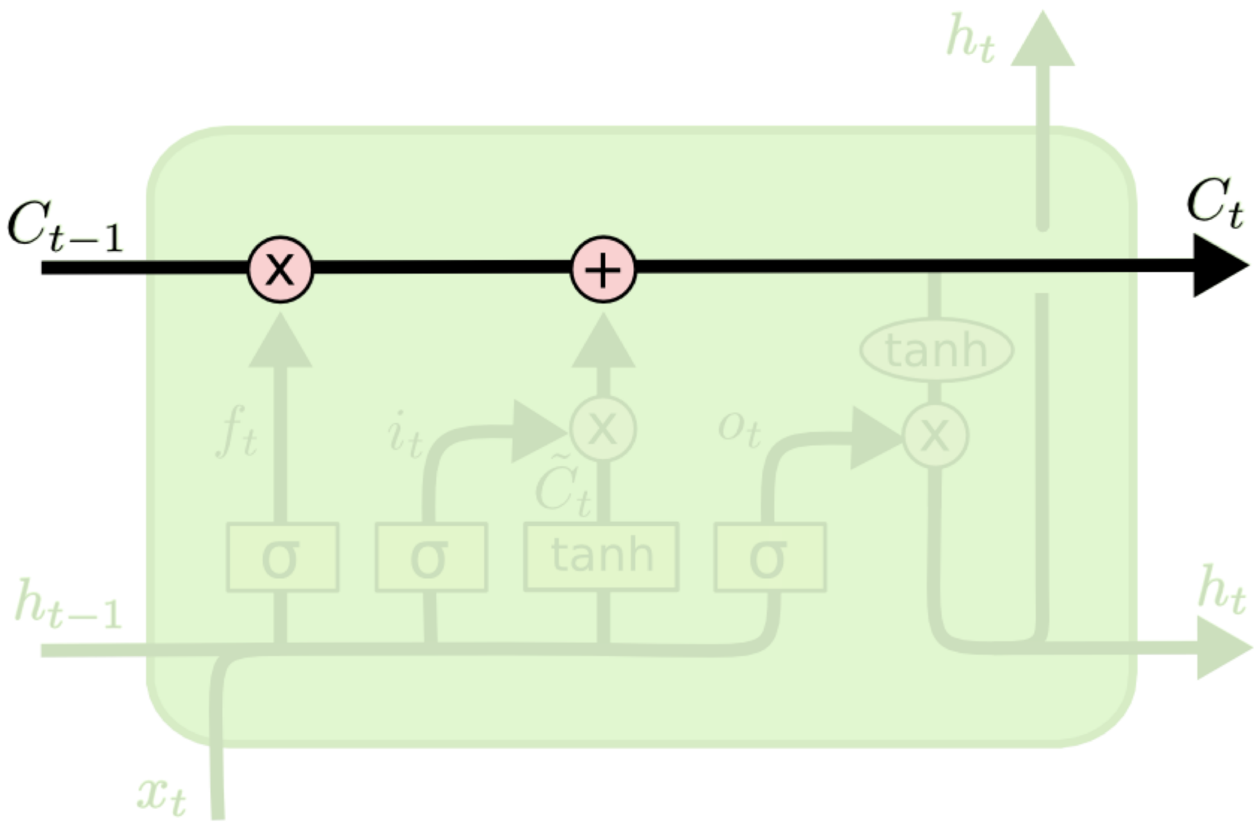
Source: Christopher Olah, (2015) Understanding LSTM Networks, http://colah.github.io/posts/2015-08-Understanding-LSTMs/

Long Short Term Memory (LSTM)



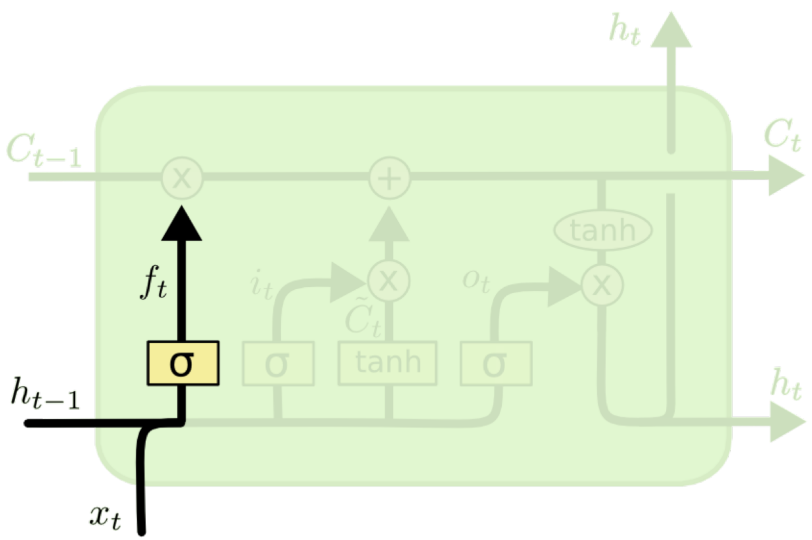
Source: Christopher Olah, (2015) Understanding LSTM Networks, http://colah.github.io/posts/2015-08-Understanding-LSTMs/

LSTM
Memory state (C)



Source: Christopher Olah, (2015) Understanding LSTM Networks, http://colah.github.io/posts/2015-08-Understanding-LSTMs/

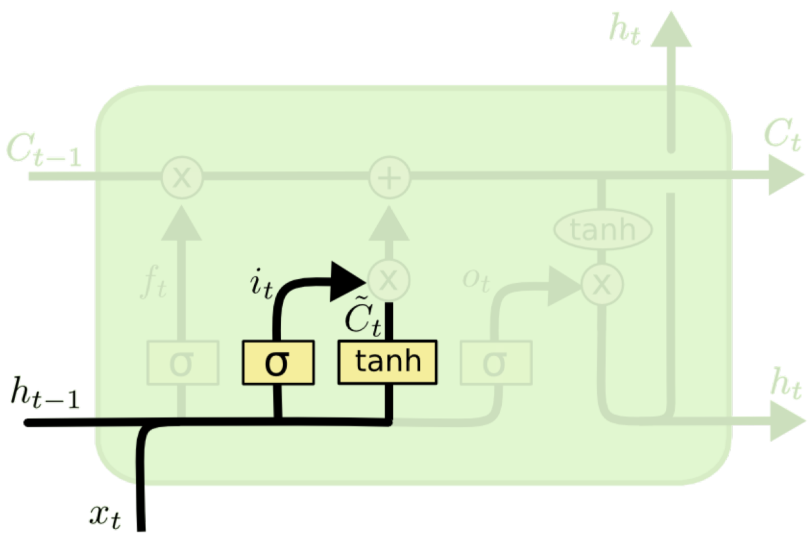
LSTM
forget gate (f)



$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \ + \ b_f\right)$$

Source: Christopher Olah, (2015) Understanding LSTM Networks, http://colah.github.io/posts/2015-08-Understanding-LSTMs/

LSTM
input gate (i)

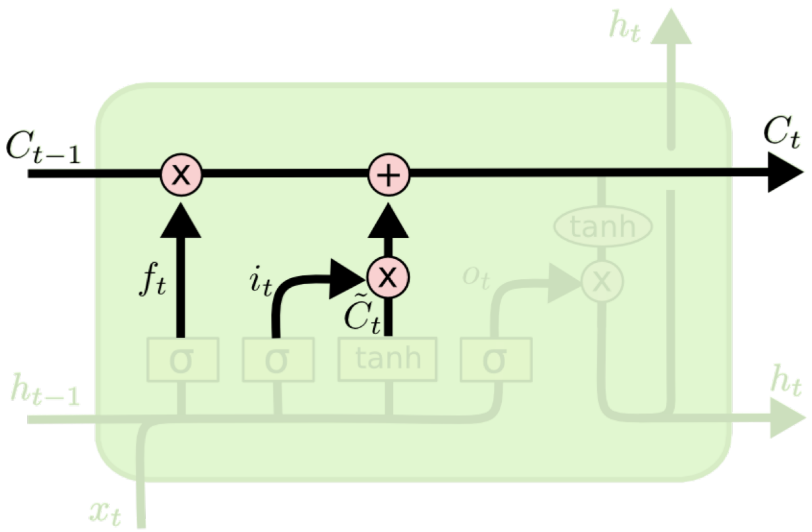


$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \ + \ b_i\right)$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \ + \ b_C)$$

Source: Christopher Olah, (2015) Understanding LSTM Networks, http://colah.github.io/posts/2015-08-Understanding-LSTMs/

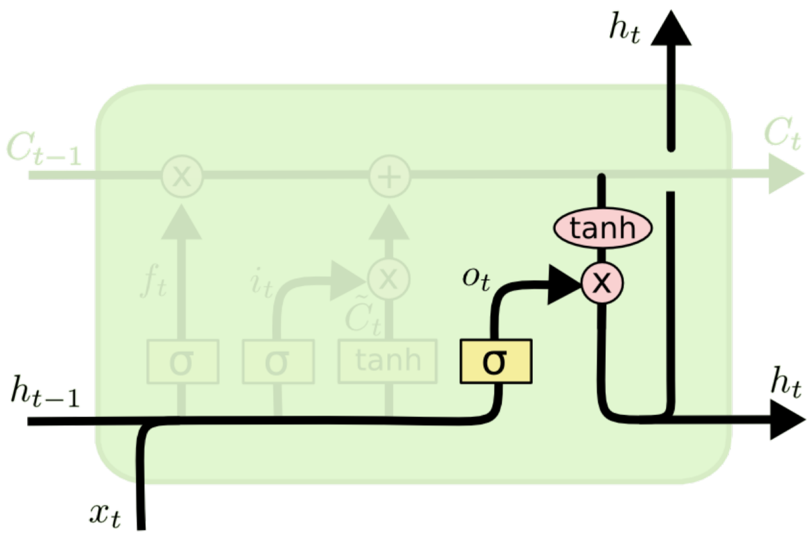
LSTM
Memory state (C)



$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

Source: Christopher Olah, (2015) Understanding LSTM Networks, http://colah.github.io/posts/2015-08-Understanding-LSTMs/

LSTM
output gate (o)

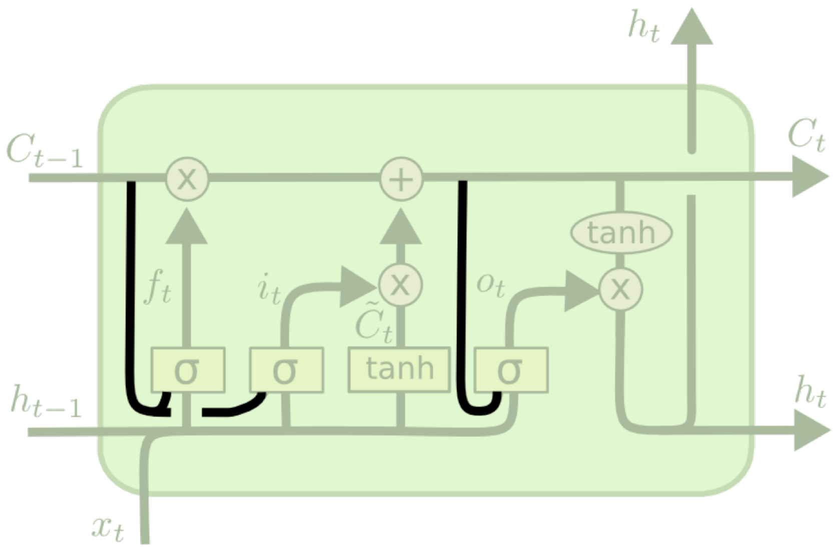


$$o_t = \sigma\left(W_o\ [h_{t-1}, x_t]\ +\ b_o\right)$$

$$h_t = o_t * \tanh\left(C_t\right)$$

Source: Christopher Olah, (2015) Understanding LSTM Networks, http://colah.github.io/posts/2015-08-Understanding-LSTMs/

LSTM
forget (f), input (i), output (o) gates

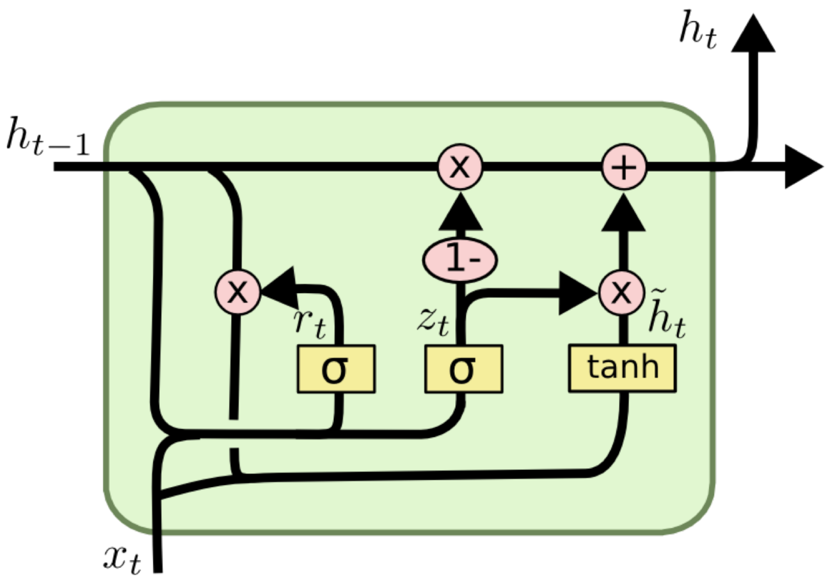


$$f_t = \sigma\left(W_f \cdot [\boldsymbol{C_{t-1}}, h_{t-1}, x_t] \ + \ b_f\right)$$
$$i_t = \sigma\left(W_i \cdot [\boldsymbol{C_{t-1}}, h_{t-1}, x_t] \ + \ b_i\right)$$
$$o_t = \sigma\left(W_o \cdot [\boldsymbol{C_t}, h_{t-1}, x_t] \ + \ b_o\right)$$

Source: Christopher Olah, (2015) Understanding LSTM Networks, http://colah.github.io/posts/2015-08-Understanding-LSTMs/

Gated Recurrent Unit (GRU)

update (z), reset (r) gates



$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$

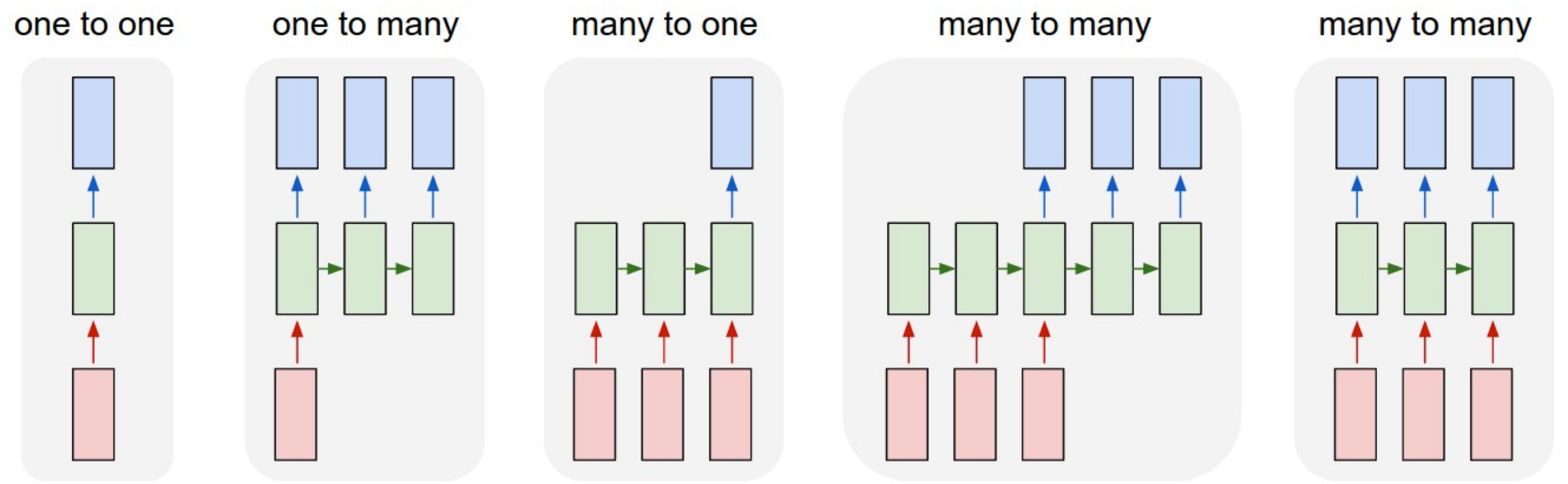
$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$

$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$

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

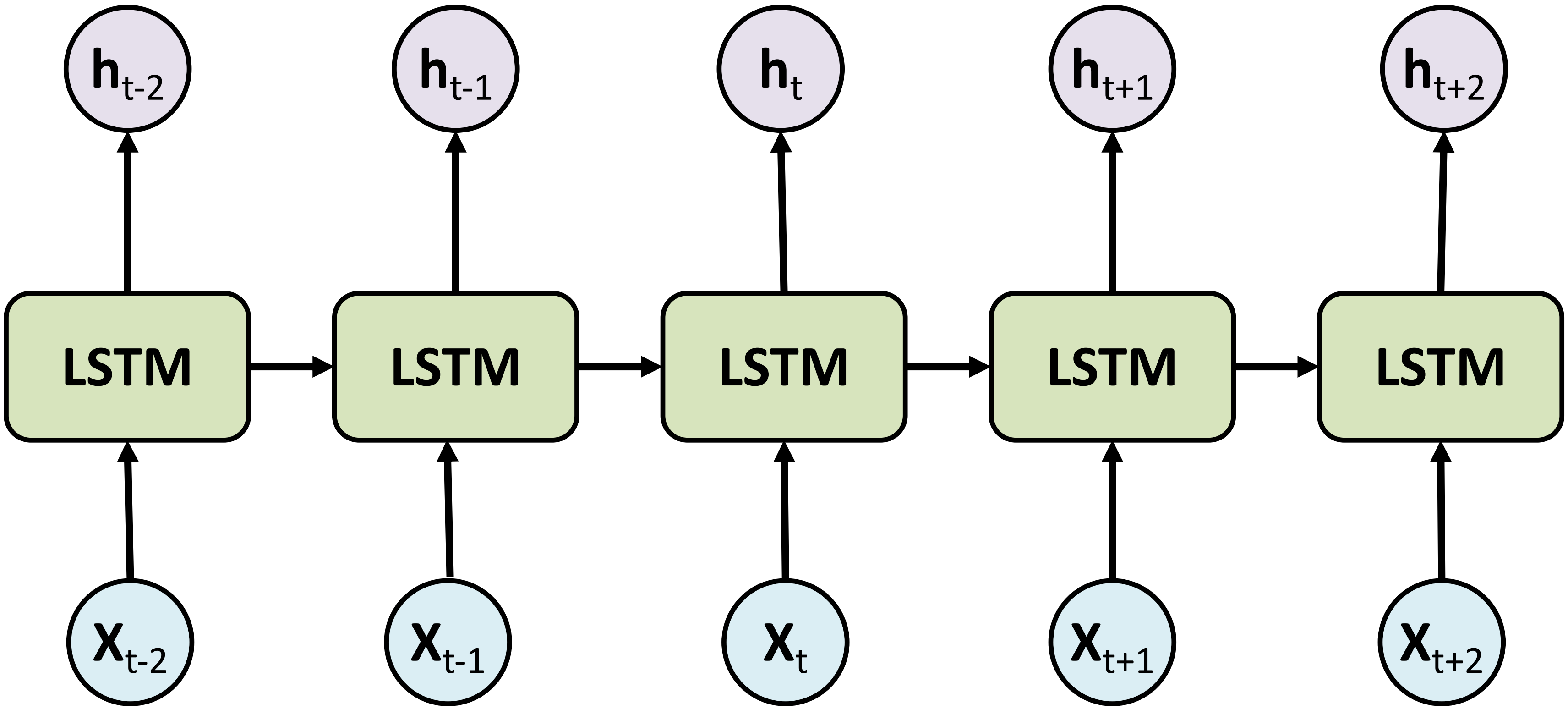
Source: Christopher Olah, (2015) Understanding LSTM Networks, http://colah.github.io/posts/2015-08-Understanding-LSTMs/

LSTM Recurrent Neural Network

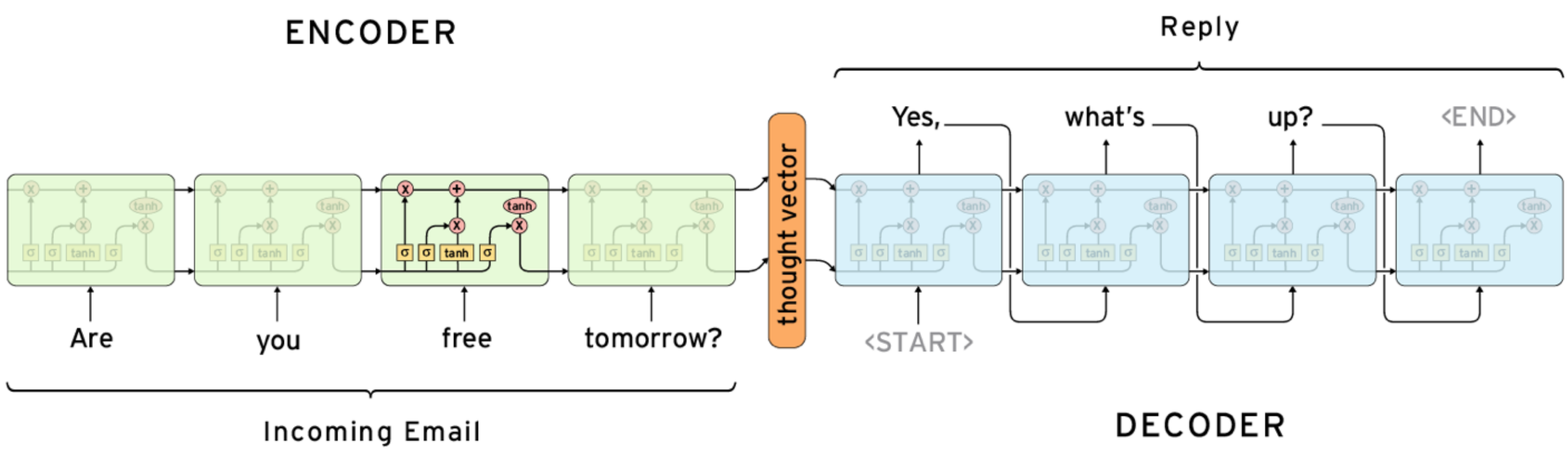


Source: https://github.com/Vict0rSch/deep_learning/tree/master/keras/recurrent

Long Short Term Memory (LSTM) for Time Series Forecasting

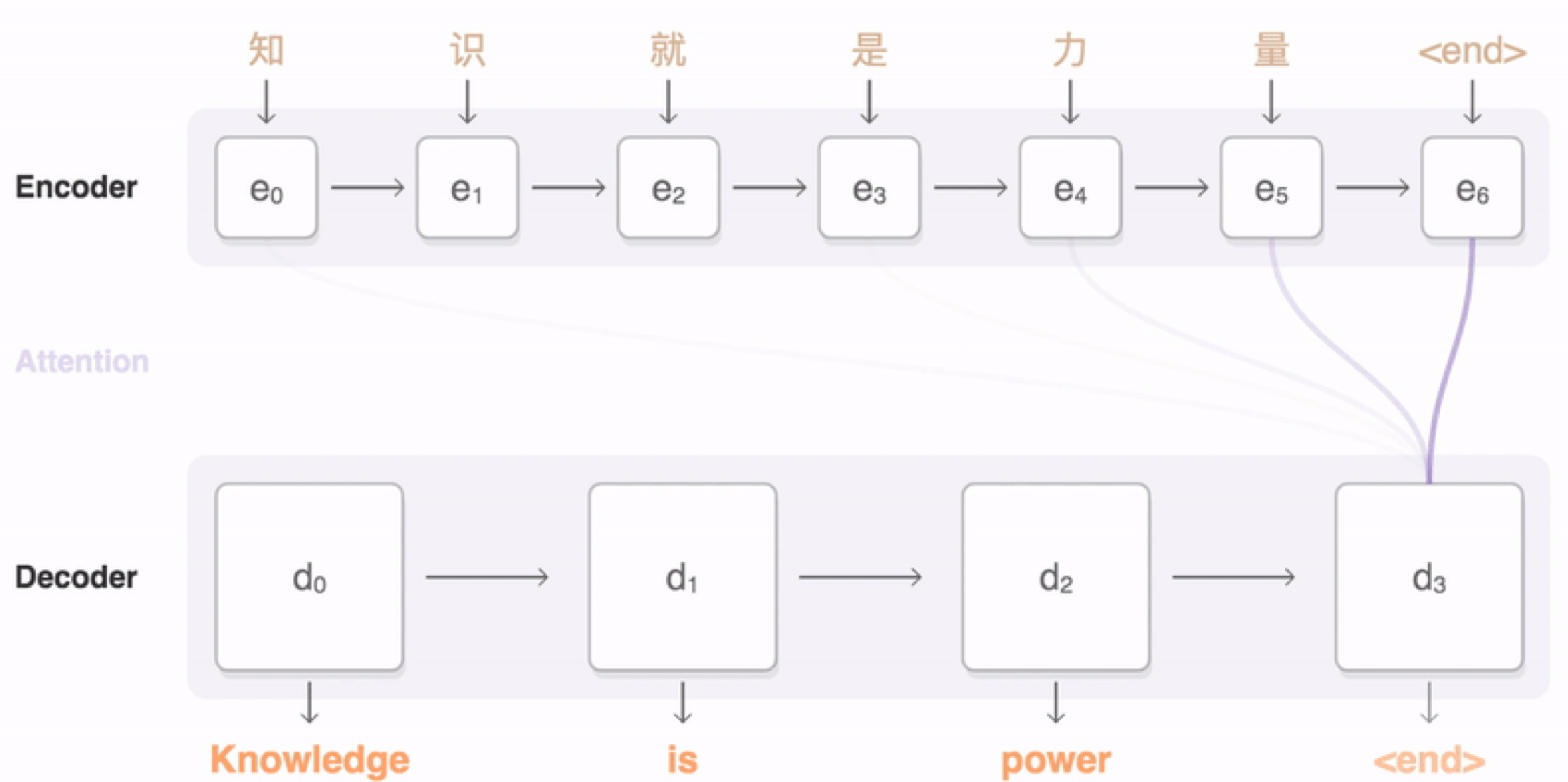


The Sequence to Sequence model (seq2seq)



Source: http://suriyadeepan.github.io/2016-12-31-practical-seq2seq/

Sequence to Sequence (Seq2Seq)



Source: https://google.github.io/seq2seq/

Natural Language Processing (NLP)

- Part-of-speech tagging
- Text segmentation
- Word sense disambiguation
- Syntactic ambiguity
- Imperfect or irregular input
- Speech acts

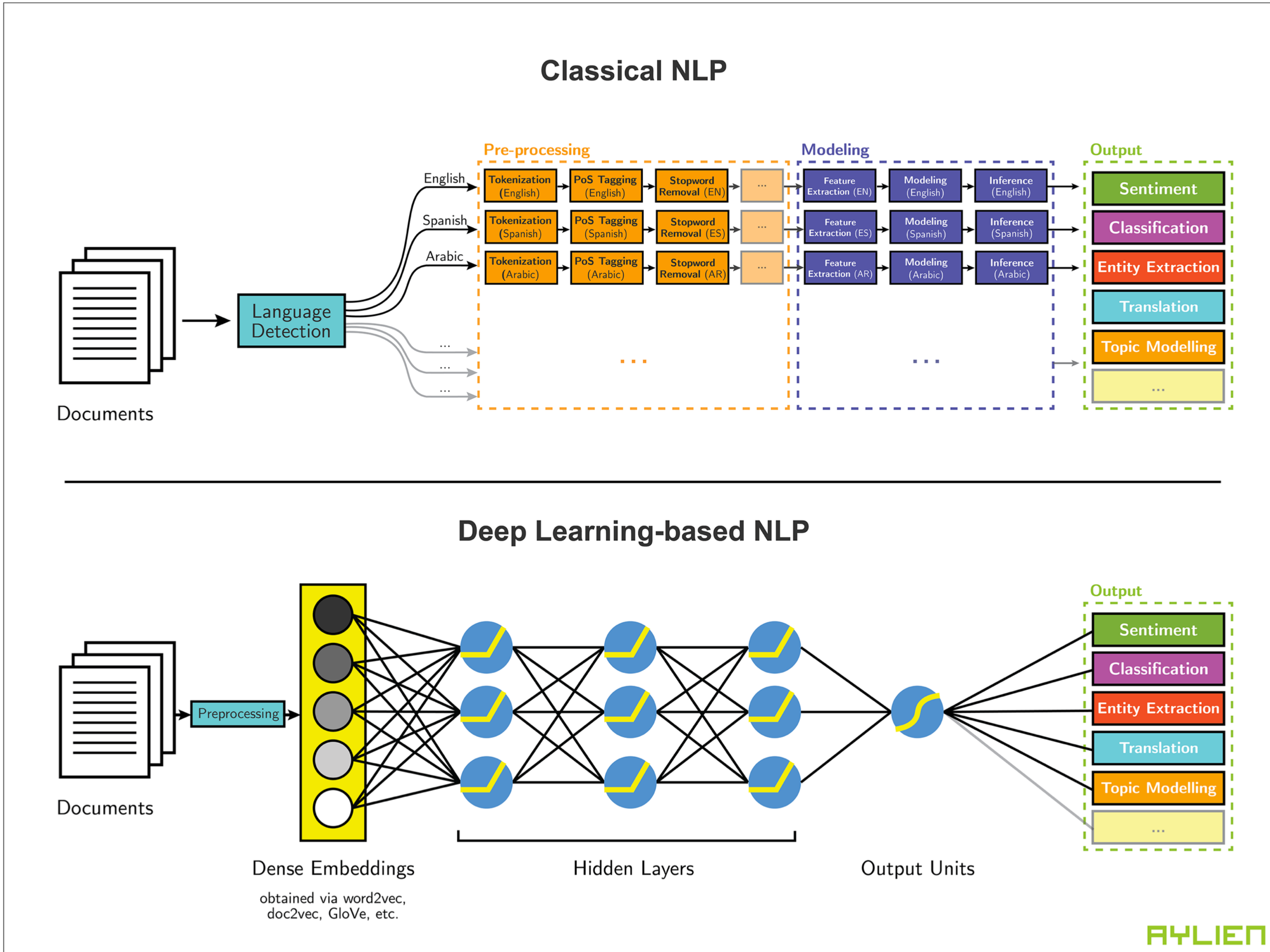
Source: Ramesh Sharda, Dursun Delen, and Efraim Turban (2017), Business Intelligence, Analytics, and Data Science: A Managerial Perspective, 4th Edition, Pearson

NLP Tasks

- Question answering
- Automatic summarization
- Natural language generation
- Natural language understanding
- Machine translation
- Foreign language reading
- Foreign language writing.
- Speech recognition
- Text-to-speech
- Text proofing
- Optical character recognition

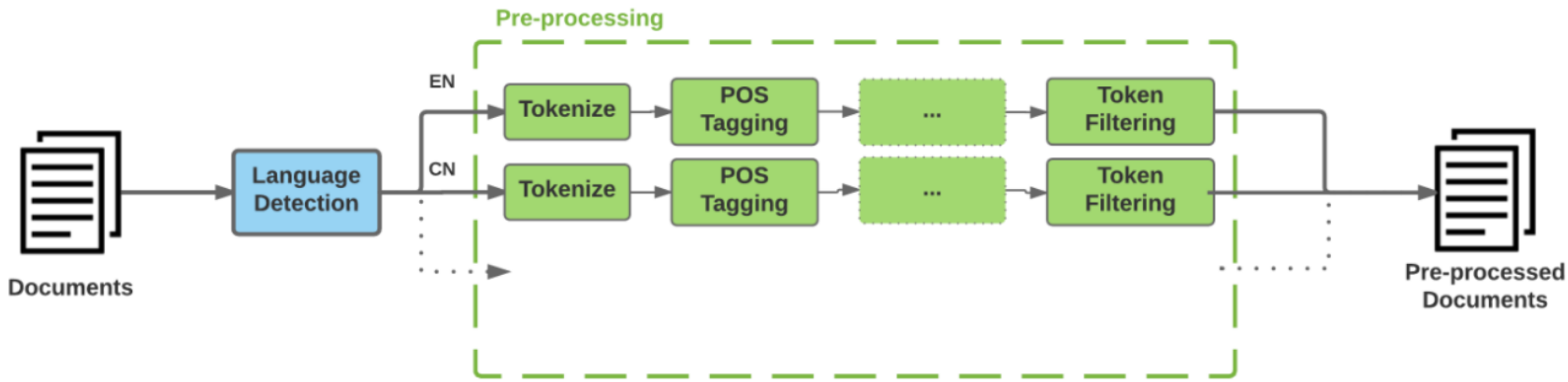
Source: Ramesh Sharda, Dursun Delen, and Efraim Turban (2017), Business Intelligence, Analytics, and Data Science: A Managerial Perspective, 4th Edition, Pearson

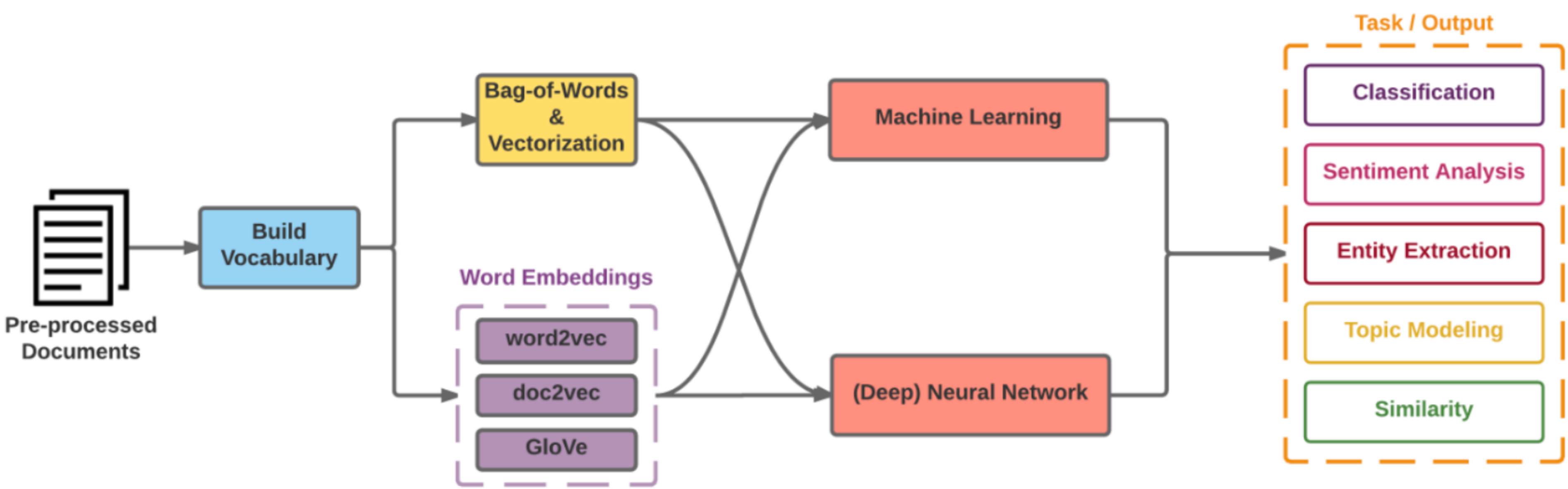
NLP



Source: http://blog.aylien.com/leveraging-deep-learning-for-multilingual/

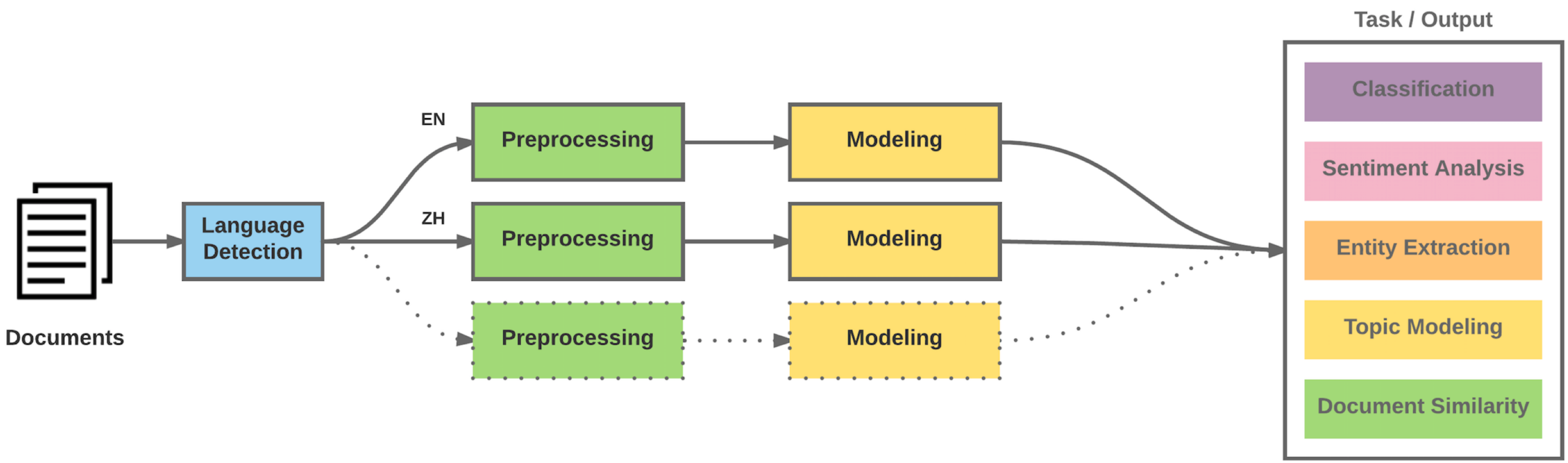
Modern NLP Pipeline





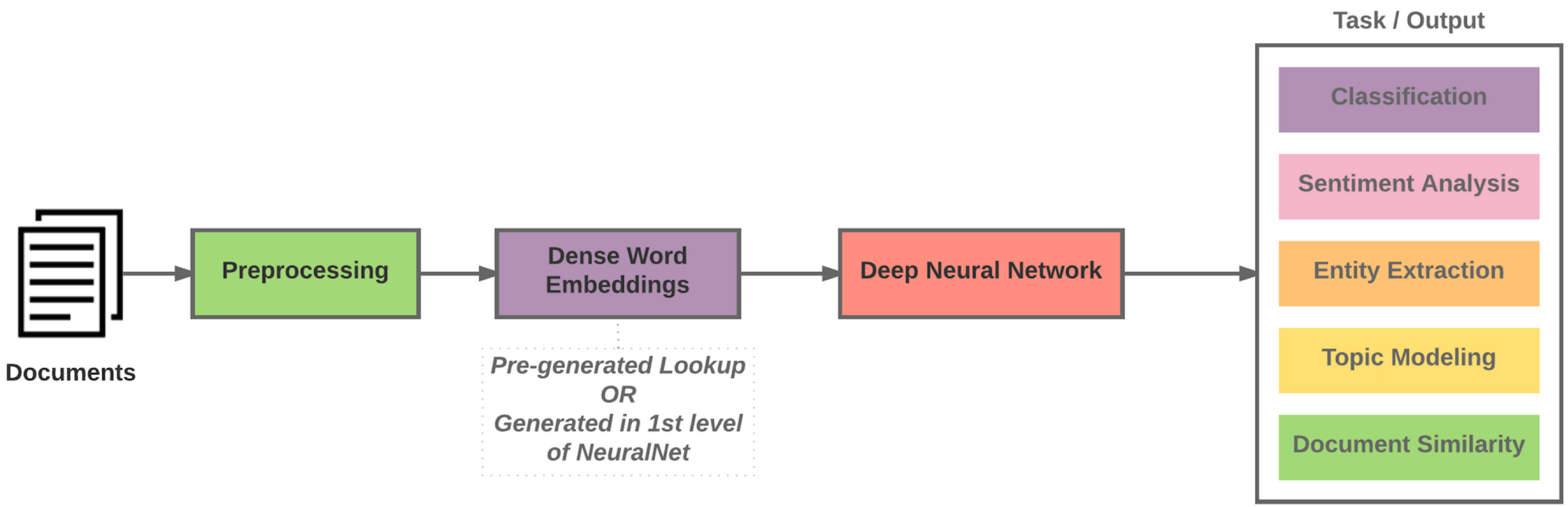
Source: https://github.com/fortiema/talks/blob/master/opendata2016sh/pragmatic-nlp-opendata2016sh.pdf

Modern NLP Pipeline



Source: http://mattfortier.me/2017/01/31/nlp-intro-pt-1-overview/

Deep Learning NLP



Source: http://mattfortier.me/2017/01/31/nlp-intro-pt-1-overview/

BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding

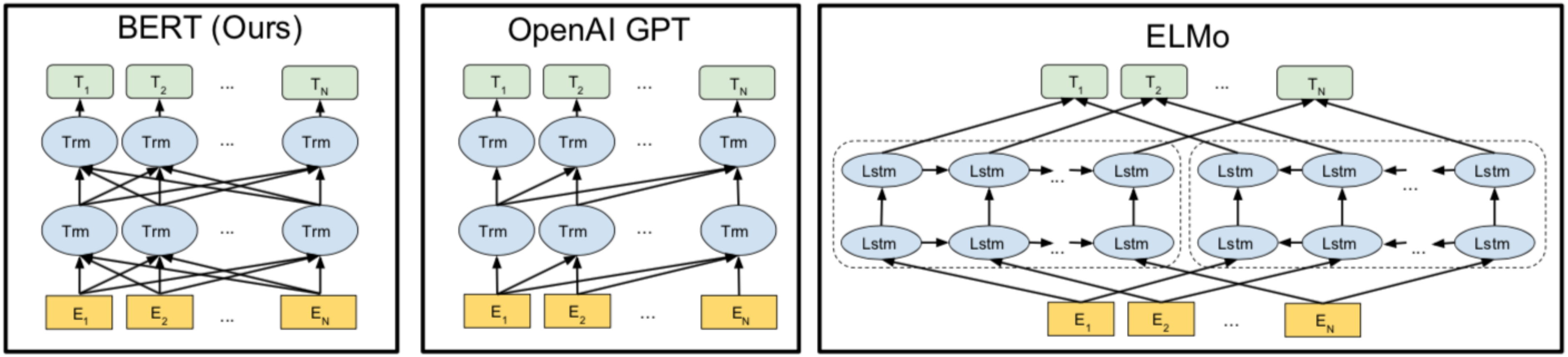
BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding

Jacob Devlin   Ming-Wei Chang   Kenton Lee   Kristina Toutanova
Google AI Language
{jacobdevlin,mingweichang,kentonl,kristout}@google.com

Source: Devlin, Jacob, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova (2018).
"BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding." arXiv preprint arXiv:1810.04805

BERT

Bidirectional Encoder Representations from Transformers

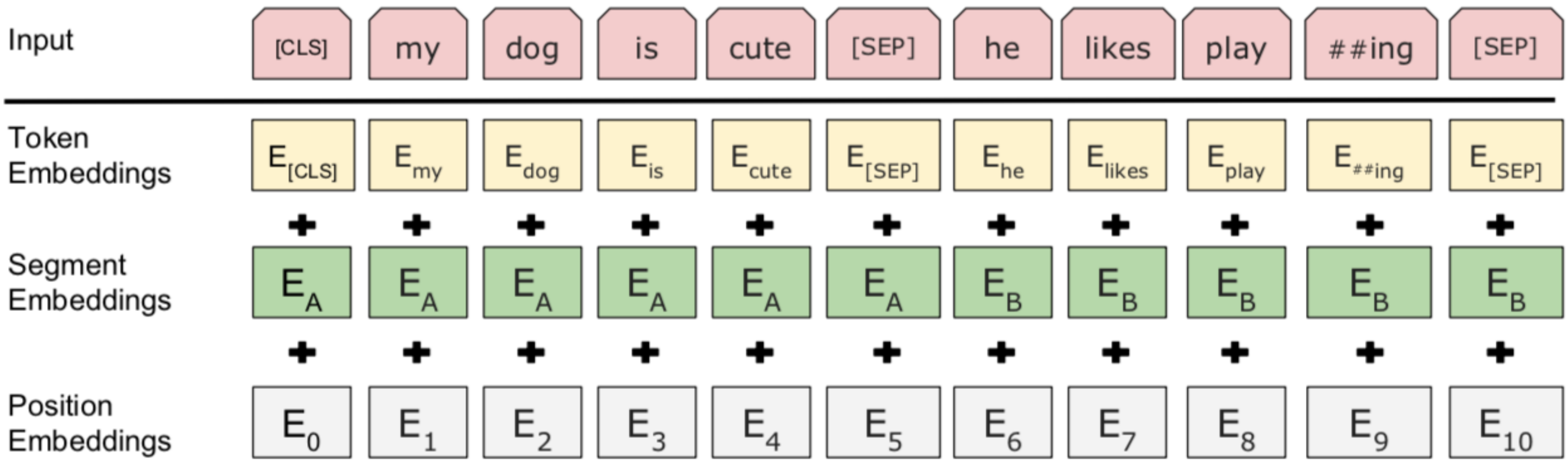


Pre-training model architectures

BERT uses a bidirectional Transformer.
OpenAI GPT uses a left-to-right Transformer.
ELMo uses the concatenation of independently trained left-to-right and right- to-left LSTM to generate features for downstream tasks.
Among three, only BERT representations are jointly conditioned on both left and right context in all layers.

Source: Devlin, Jacob, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova (2018).
"BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding." arXiv preprint arXiv:1810.04805

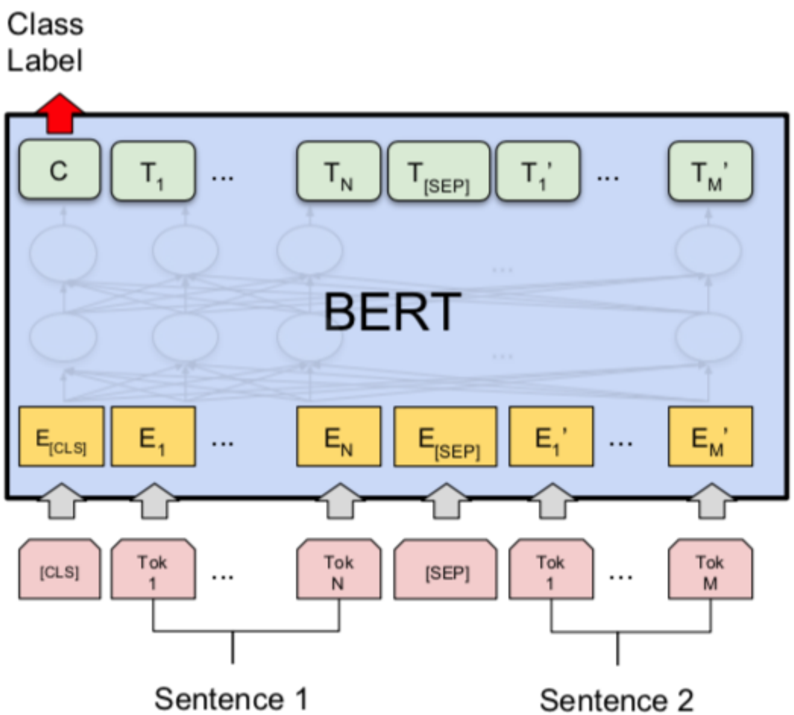
BERT input representation



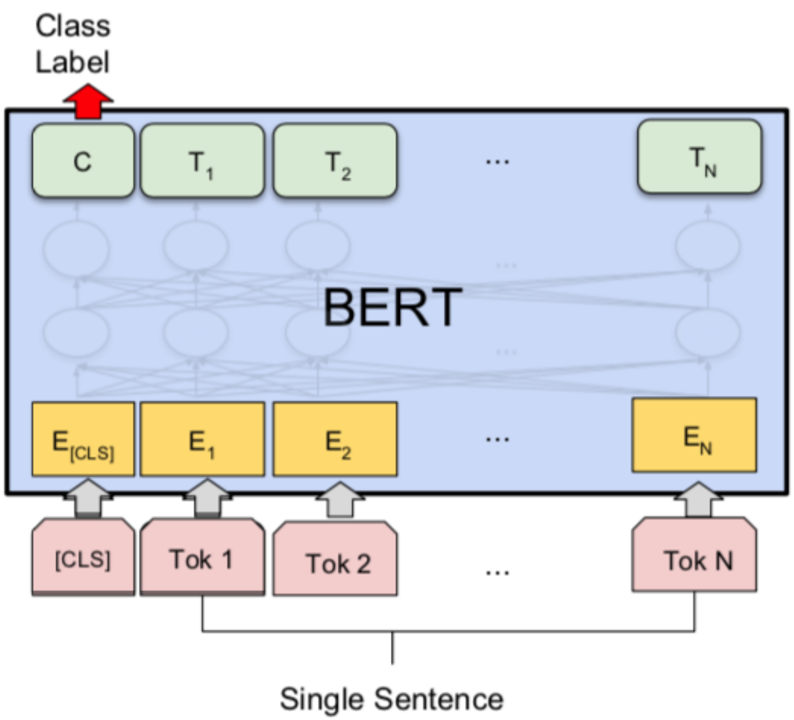
The input embeddings is the sum of the token embeddings, the segmentation embeddings and the position embeddings.

Source: Devlin, Jacob, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova (2018). "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding." arXiv preprint arXiv:1810.04805

BERT Sequence-level tasks



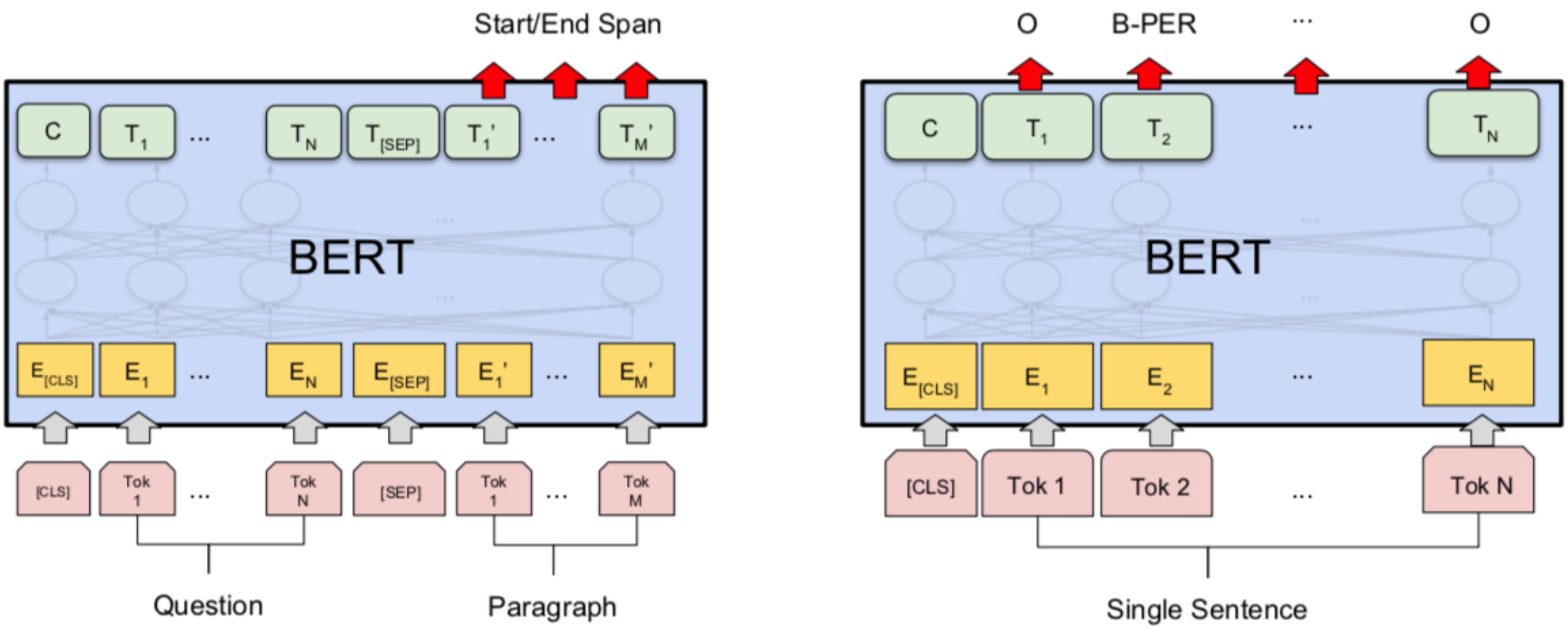
(a) Sentence Pair Classification Tasks: MNLI, QQP, QNLI, STS-B, MRPC, RTE, SWAG



(b) Single Sentence Classification Tasks: SST-2, CoLA

Source: Devlin, Jacob, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova (2018). "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding." arXiv preprint arXiv:1810.04805

BERT Token-level tasks



(c) Question Answering Tasks:
SQuAD v1.1

(d) Single Sentence Tagging Tasks:
CoNLL-2003 NER

Source: Devlin, Jacob, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova (2018).
"BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding." arXiv preprint arXiv:1810.04805

General Language Understanding Evaluation (GLUE) benchmark

GLUE Test results

System	MNLI-(m/mm)	QQP	QNLI	SST-2	CoLA	STS-B	MRPC	RTE	**Average**
	392k	363k	108k	67k	8.5k	5.7k	3.5k	2.5k	-
Pre-OpenAI SOTA	80.6/80.1	66.1	82.3	93.2	35.0	81.0	86.0	61.7	74.0
BiLSTM+ELMo+Attn	76.4/76.1	64.8	79.9	90.4	36.0	73.3	84.9	56.8	71.0
OpenAI GPT	82.1/81.4	70.3	88.1	91.3	45.4	80.0	82.3	56.0	75.2
$\text{BERT}_{\text{BASE}}$	84.6/83.4	71.2	90.1	93.5	52.1	85.8	88.9	66.4	79.6
$\text{BERT}_{\text{LARGE}}$	**86.7/85.9**	**72.1**	**91.1**	**94.9**	**60.5**	**86.5**	**89.3**	**70.1**	**81.9**

MNLI: Multi-Genre Natural Language Inference
QQP: Quora Question Pairs
QNLI: Question Natural Language Inference
SST-2: The Stanford Sentiment Treebank
CoLA: The Corpus of Linguistic Acceptability
STS-B:The Semantic Textual Similarity Benchmark
MRPC: Microsoft Research Paraphrase Corpus
RTE: Recognizing Textual Entailment

Source: Devlin, Jacob, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova (2018).
"BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding." arXiv preprint arXiv:1810.04805

NLP Libraries and Tools

Natural Language Processing with Python
– Analyzing Text with the Natural Language Toolkit

www.nltk.org/book/

Natural Language Processing with Python

NLTK

– Analyzing Text with the Natural Language Toolkit

Steven Bird, Ewan Klein, and Edward Loper

This version of the NLTK book is updated for Python 3 and NLTK 3. The first edition of the book, published by O'Reilly, is available at http://nltk.org/book_1ed/. (There are currently no plans for a second edition of the book.)

Please post any questions about the materials to the nltk-users mailing list. Please report any errors on the issue tracker.

http://www.nltk.org/book/

spaCy

spaCy

HOME USAGE API DEMOS BLOG

Industrial-Strength Natural Language Processing

in Python

Fastest in the world

spaCy excels at large-scale information extraction tasks. It's written from the ground up in carefully memory-managed Cython. Independent research has confirmed that spaCy is the fastest in the world. If your application needs to process entire web dumps, spaCy is the library you want to be using.

Get things done

spaCy is designed to help you do real work — to build real products, or gather real insights. The library respects your time, and tries to avoid wasting it. It's easy to install, and its API is simple and productive. I like to think of spaCy as the Ruby on Rails of Natural Language Processing.

Deep learning

spaCy is the best way to prepare text for deep learning. It interoperates seamlessly with TensorFlow, Keras, Scikit-Learn, Gensim and the rest of Python's awesome AI ecosystem. spaCy helps you connect the statistical models trained by these libraries to the rest of your application.

https://spacy.io/

gensim

https://radimrehurek.com/gensim/

TextBlob

TextBlob

Star 3,777

TextBlob is a Python (2 and 3) library for processing textual data. It provides a consistent API for diving into common natural language processing (NLP) tasks such as part-of-speech tagging, noun phrase extraction, sentiment analysis, and more.

Useful Links

TextBlob @ PyPI
TextBlob @ GitHub
Issue Tracker

Stay Informed

Follow @sloria

Donate

If you find TextBlob useful,

TextBlob: Simplified Text Processing

Release v0.12.0. (Changelog)

TextBlob is a Python (2 and 3) library for processing textual data. It provides a simple API for diving into common natural language processing (NLP) tasks such as part-of-speech tagging, noun phrase extraction, sentiment analysis, classification, translation, and more.

```
from textblob import TextBlob

text = '''
The titular threat of The Blob has always struck me as the ultimate movie
monster: an insatiably hungry, amoeba-like mass able to penetrate
virtually any safeguard, capable of--as a doomed doctor chillingly
describes it--"assimilating flesh on contact.
Snide comparisons to gelatin be damned, it's a concept with the most
devastating of potential consequences, not unlike the grey goo scenario
proposed by technological theorists fearful of
artificial intelligence run rampant.
'''

blob = TextBlob(text)
blob.tags           # [('The', 'DT'), ('titular', 'JJ'),
                    #  ('threat', 'NN'), ('of', 'IN'), ...]

blob.noun_phrases   # WordList(['titular threat', 'blob',
                    #             'ultimate movie monster',
                    #             'amoeba-like mass', ...])

for sentence in blob.sentences:
    print(sentence.sentiment.polarity)
# 0.060
```

https://textblob.readthedocs.io

Polyglot

polyglot
latest

Search docs

- Installation
- Language Detection
- Tokenization
- Command Line Interface
- Downloading Models
- Word Embeddings
- Part of Speech Tagging
- Named Entity Extraction
- Morphological Analysis
- Transliteration
- Sentiment
- polyglot

Docs » Welcome to polyglot's documentation!

Edit on GitHub

Welcome to polyglot's documentation!

polyglot

downloads 17k/month | pypi package 16.7.4 | build passing | docs passing

Polyglot is a natural language pipeline that supports massive multilingual applications.

- Free software: GPLv3 license
- Documentation: http://polyglot.readthedocs.org.

Features

- Tokenization (165 Languages)
- Language detection (196 Languages)
- Named Entity Recognition (40 Languages)
- Part of Speech Tagging (16 Languages)
- Sentiment Analysis (136 Languages)
- Word Embeddings (137 Languages)
- Morphological analysis (135 Languages)
- Transliteration (69 Languages)

https://polyglot.readthedocs.io/

scikit-learn

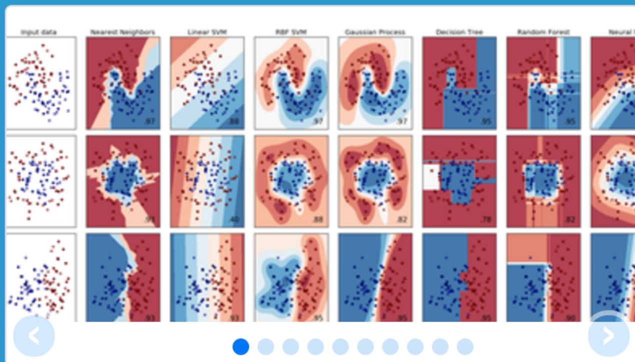
scikit learn

Home Installation Documentation Examples

powered by Google

Google Custom Search Search

Fork me on GitHub



scikit-learn

Machine Learning in Python

- Simple and efficient tools for data mining and data analysis
- Accessible to everybody, and reusable in various contexts
- Built on NumPy, SciPy, and matplotlib
- Open source, commercially usable - BSD license

Classification

Identifying to which category an object belongs to.

Applications: Spam detection, Image recognition.
Algorithms: SVM, nearest neighbors, random forest, ... — Examples

Regression

Predicting a continuous-valued attribute associated with an object.

Applications: Drug response, Stock prices.
Algorithms: SVR, ridge regression, Lasso, ... — Examples

Clustering

Automatic grouping of similar objects into sets.

Applications: Customer segmentation, Grouping experiment outcomes
Algorithms: k-Means, spectral clustering, mean-shift, ... — Examples

Dimensionality reduction

Reducing the number of random variables to consider.

Applications: Visualization, Increased efficiency

Model selection

Comparing, validating and choosing parameters and models.

Goal: Improved accuracy via parameter tuning

Preprocessing

Feature extraction and normalization.

Application: Transforming input data such as text for use with machine learning algorithms.
Modules: preprocessing, feature extraction.

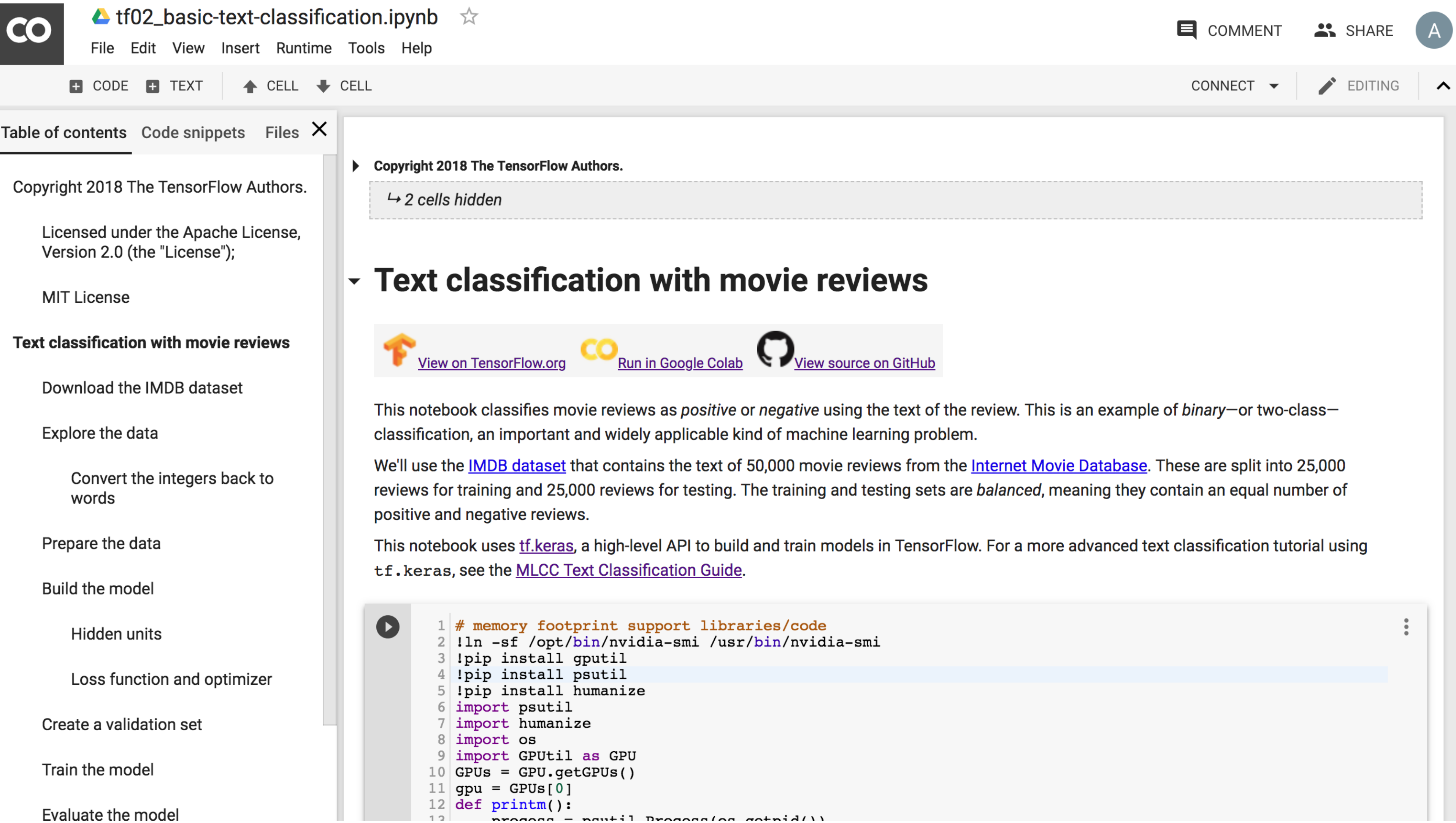
http://scikit-learn.org/

TensorFlow NLP Examples

- Basic Text Classification
  (Text Classification) (46 Seconds)
  - https://colab.research.google.com/github/tensorflow/docs/blob/master/site/en/tutorials/keras/basic_text_classification.ipynb
- NMT with Attention
  (20-30 minutes)
  - https://colab.research.google.com/github/tensorflow/tensorflow/blob/master/tensorflow/contrib/eager/python/examples/nmt_with_attention/nmt_with_attention.ipynb

Text Classification
IMDB Movie Reviews

https://colab.research.google.com/drive/1x16h1GhHsLIrLYtPCvCHaoO1W-i_gror



tf02_basic-text-classification.ipynb

File Edit View Insert Runtime Tools Help

COMMENT SHARE

CODE TEXT CELL CELL CONNECT EDITING

Table of contents Code snippets Files

Copyright 2018 The TensorFlow Authors.

2 cells hidden

Text classification with movie reviews

View on TensorFlow.org Run in Google Colab View source on GitHub

This notebook classifies movie reviews as *positive* or *negative* using the text of the review. This is an example of *binary*—or two-class—classification, an important and widely applicable kind of machine learning problem.

We'll use the IMDB dataset that contains the text of 50,000 movie reviews from the Internet Movie Database. These are split into 25,000 reviews for training and 25,000 reviews for testing. The training and testing sets are *balanced*, meaning they contain an equal number of positive and negative reviews.

This notebook uses tf.keras, a high-level API to build and train models in TensorFlow. For a more advanced text classification tutorial using `tf.keras`, see the MLCC Text Classification Guide.

```
# memory footprint support libraries/code
!ln -sf /opt/bin/nvidia-smi /usr/bin/nvidia-smi
!pip install gputil
!pip install psutil
!pip install humanize
import psutil
import humanize
import os
import GPUtil as GPU
GPUs = GPU.getGPUs()
gpu = GPUs[0]
def printm():
```

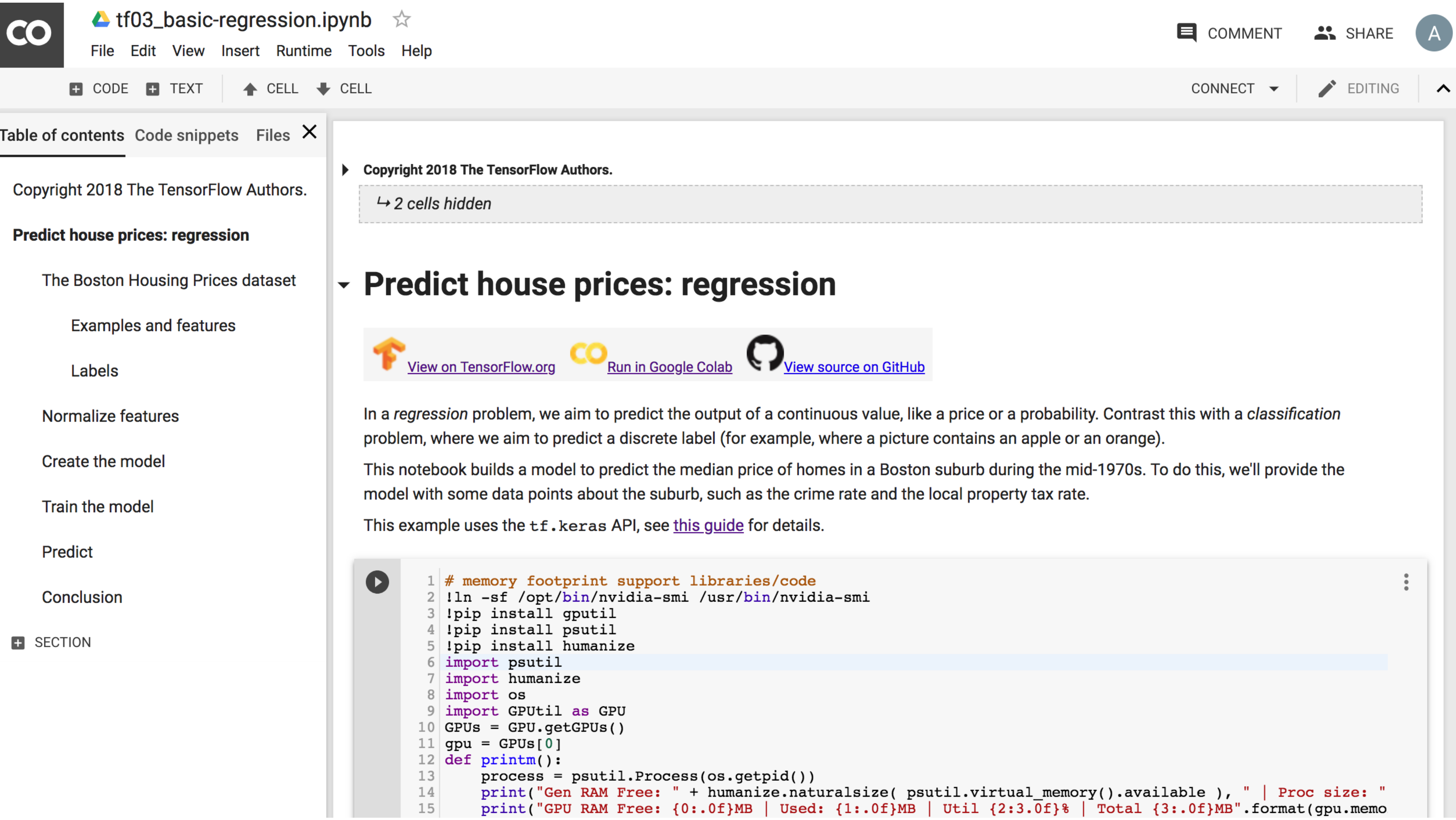
Source: https://colab.research.google.com/github/tensorflow/docs/blob/master/site/en/tutorials/keras/basic_text_classification.ipynb

AI + VDI POS TensorFlow Models

- M1: Basic Classification (Image Classification) (65 Seconds)
  - https://colab.research.google.com/github/tensorflow/docs/blob/master/site/en/tutorials/keras/basic_classification.ipynb
- M2: Basic Text Classification (Text Classification) (46 Seconds)
  - https://colab.research.google.com/github/tensorflow/docs/blob/master/site/en/tutorials/keras/basic_text_classification.ipynb
- M3: Basic Regression (Predict House Prices) (43 Seconds)
  - https://colab.research.google.com/github/tensorflow/docs/blob/master/site/en/tutorials/keras/basic_regression.ipynb
- M4: Pix2Pix Eager (Option) (7-8 Hours)
  - https://colab.research.google.com/github/tensorflow/tensorflow/blob/master/tensorflow/contrib/eager/python/examples/pix2pix/pix2pix_eager.ipynb
- M5. NMT with Attention (Option) (20-30 minutes)
  - https://colab.research.google.com/github/tensorflow/tensorflow/blob/master/tensorflow/contrib/eager/python/examples/nmt_with_attention/nmt_with_attention.ipynb

Basic Regression
Predict House Prices

https://colab.research.google.com/drive/1v4c8ZHTnRtgd2_25K_AURjR6SCVBRdlj



Source: https://colab.research.google.com/github/tensorflow/docs/blob/master/site/en/tutorials/keras/basic_regression.ipynb

Deep Learning for Financial Market Prediction

Deep Learning for Financial Market Prediction

Stock Market Prediction

Stock Price Prediction

Time Series Prediction

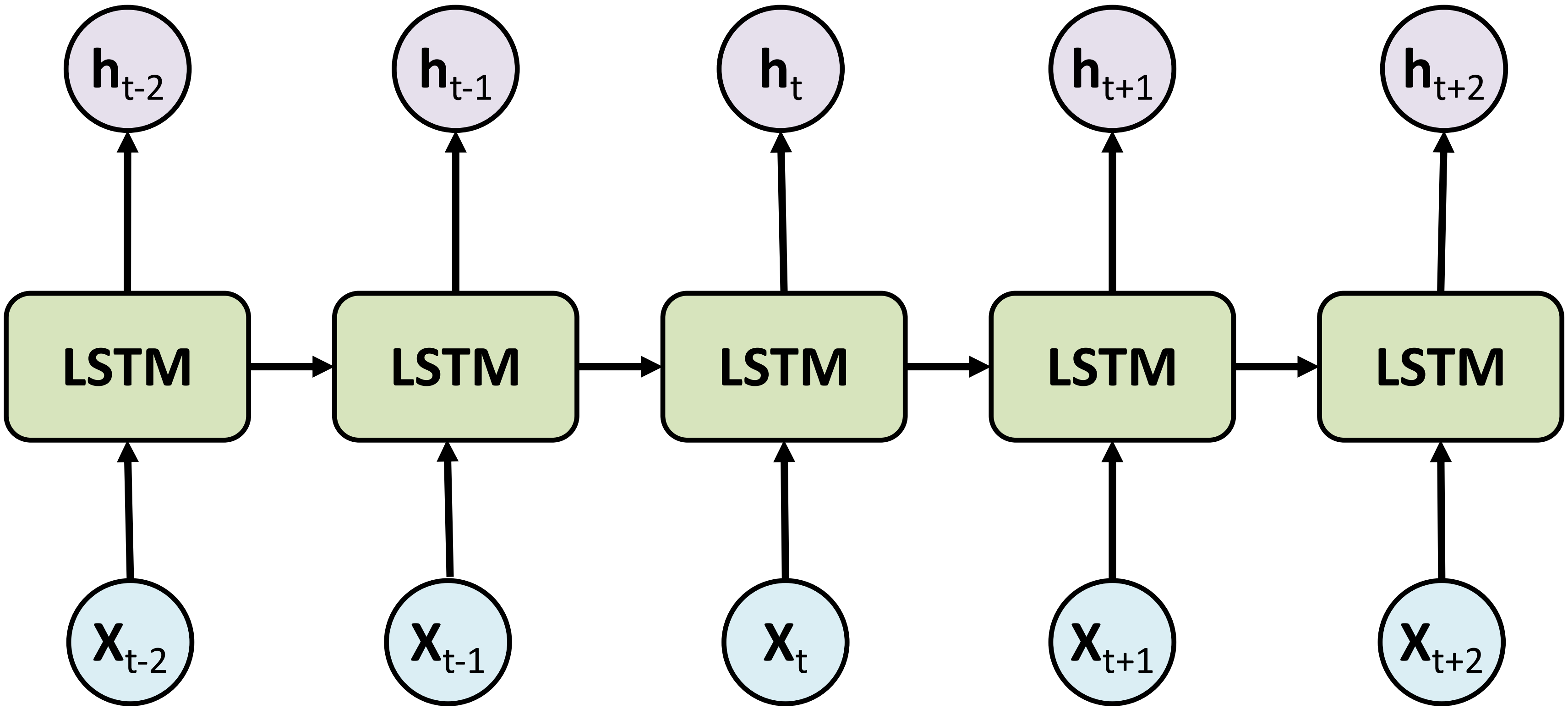
Time Series Data

```
df['Adj Close'].plot(legend=True, figsize=(12, 8), title='AAPL', label='Adj Close')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x1150bac88>
```

Source: https://mapattack.wordpress.com/2017/02/12/using-python-for-stocks-1/

Long Short Term Memory (LSTM) for Time Series Forecasting



Time Series Data

[10, 20, 30, 40, 50, 60, 70, 80, 90]

X	Y
[10 20 30]	40
[20 30 40]	50
[30 40 50]	60
[40 50 60]	70
[50 60 70]	80
[60 70 80]	90

Deep Learning for Financial Time Series Forecasting

https://colab.research.google.com/drive/1aEK0eSev8Q-Y0nNY32geFk7CB8pVgSQM

Deep_Learning_for_Financial_Time_Series_Forecasting.ipynb

File Edit View Insert Runtime Tools Help

CODE TEXT CELL CELL

```python
# univariate data preparation
from numpy import array
# split a univariate sequence into samples
def split_sequence(sequence, n_steps):
    X, y = list(), list()
    for i in range(len(sequence)):
        # find the end of this pattern
        end_ix = i + n_steps
        # check if we are beyond the sequence
        if end_ix > len(sequence)-1:
            break
        # gather input and output parts of the pattern
        seq_x, seq_y = sequence[i:end_ix], sequence[end_ix]
        X.append(seq_x)
        y.append(seq_y)
    return array(X), array(y)
# define input sequence
raw_seq = [10, 20, 30, 40, 50, 60, 70, 80, 90]
# choose a number of time steps
n_steps = 3
# split into samples
X, y = split_sequence(raw_seq, n_steps)
# summarize the data
for i in range(len(X)):
    print(X[i], y[i])
```

```
[10 20 30] 40
[20 30 40] 50
[30 40 50] 60
[40 50 60] 70
[50 60 70] 80
[60 70 80] 90
```

Source: https://machinelearningmastery.com/how-to-develop-lstm-models-for-time-series-forecasting/

Deep Learning for Financial Time Series Forecasting

https://colab.research.google.com/drive/1aEK0eSev8Q-Y0nNY32geFk7CB8pVgSQM

Deep_Learning_for_Financial_Time_Series_Forecasting.ipynb

File Edit View Insert Runtime Tools Help

COMMENT SHARE

CODE TEXT CELL CELL CONNECTED EDITING

LSTM for Time Series Forecasting

```python
# univariate lstm example
from numpy import array
from keras.models import Sequential
from keras.layers import LSTM
from keras.layers import Dense
import matplotlib.pyplot as plt
%matplotlib inline

# define dataset
X = array([[100, 110, 120], [110, 120, 130], [120, 130, 140], [130, 140, 150], [140, 150, 160]])
y = array([130, 140, 150, 160, 170])
# reshape from [samples, timesteps] into [samples, timesteps, features]
X = X.reshape((X.shape[0], X.shape[1], 1))
# define model
model = Sequential()
model.add(LSTM(50, activation='relu', input_shape=(3, 1)))
model.add(Dense(1))
model.compile(optimizer='adam', loss='mse')
# fit model
history = model.fit(X, y, epochs=2000, verbose=0)
# demonstrate prediction
x_input = array([150, 160, 170])
x_input = x_input.reshape((1, 3, 1))
yhat = model.predict(x_input, verbose=0)
print('yhat', yhat)
print(model.summary())
# list all data in history
print(history.history.keys())
# summarize history for loss
print('loss:', '%f'%history.history['loss'][-1])
print('loss:', history.history['loss'][-1])
plt.plot(history.history['loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.show()
```

```
yhat [[181.34615]]
```

Source: https://machinelearningmastery.com/how-to-develop-lstm-models-for-time-series-forecasting/

Deep Learning for Financial Time Series Forecasting

https://colab.research.google.com/drive/1aEK0eSev8Q-Y0nNY32geFk7CB8pVgSQM

CO Deep_Learning_for_Financial_Time_Series_Forecasting.ipynb

File Edit View Insert Runtime Tools Help

COMMENT SHARE

CODE TEXT CELL CELL CONNECTED EDITING

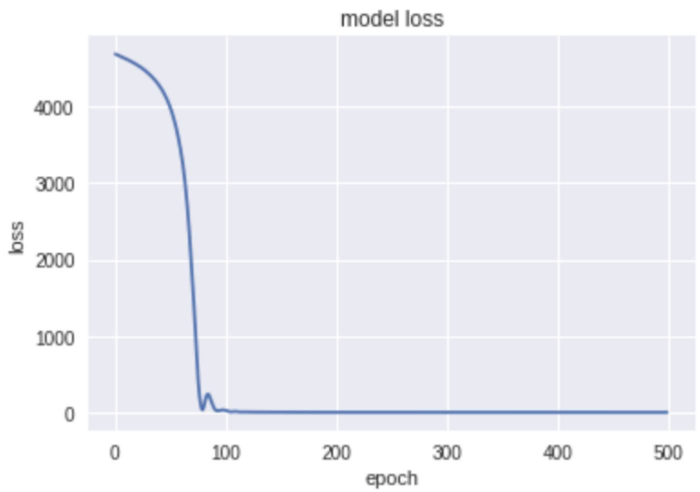
```python
# univariate lstm example
from numpy import array
from keras.models import Sequential
from keras.layers import LSTM
from keras.layers import Dense
import matplotlib.pyplot as plt
%matplotlib inline
# split a univariate sequence into samples
def split_sequence(sequence, n_steps):
    X, y = list(), list()
    for i in range(len(sequence)):
        # find the end of this pattern
        end_ix = i + n_steps
        # check if we are beyond the sequence
        if end_ix > len(sequence)-1:
            break
        # gather input and output parts of the pattern
        seq_x, seq_y = sequence[i:end_ix], sequence[end_ix]
        X.append(seq_x)
        y.append(seq_y)
    return array(X), array(y)
# define input sequence
raw_seq = [10, 20, 30, 40, 50, 60, 70, 80, 90]
# choose a number of time steps
n_steps = 3
# split into samples
X, y = split_sequence(raw_seq, n_steps)
# reshape from [samples, timesteps] into [samples, timesteps, features]
n_features = 1
X = X.reshape((X.shape[0], X.shape[1], n_features))
# define model
model = Sequential()
model.add(LSTM(50, activation='relu', input_shape=(n_steps, n_features)))
model.add(Dense(1))
model.compile(optimizer='adam', loss='mse')
# fit model
history = model.fit(X, y, epochs=500, verbose=0)
# demonstrate prediction
x_input = array([70, 80, 90])
x_input = x_input.reshape((1, n_steps, n_features))
yhat = model.predict(x_input, verbose=0)
print(yhat)
print('yhat', yhat)
print(model.summary())
```

Source: https://machinelearningmastery.com/how-to-develop-lstm-models-for-time-series-forecasting/

Deep Learning for
Financial Time Series Forecasting

https://colab.research.google.com/drive/1aEK0eSev8Q-Y0nNY32geFk7CB8pVgSQM

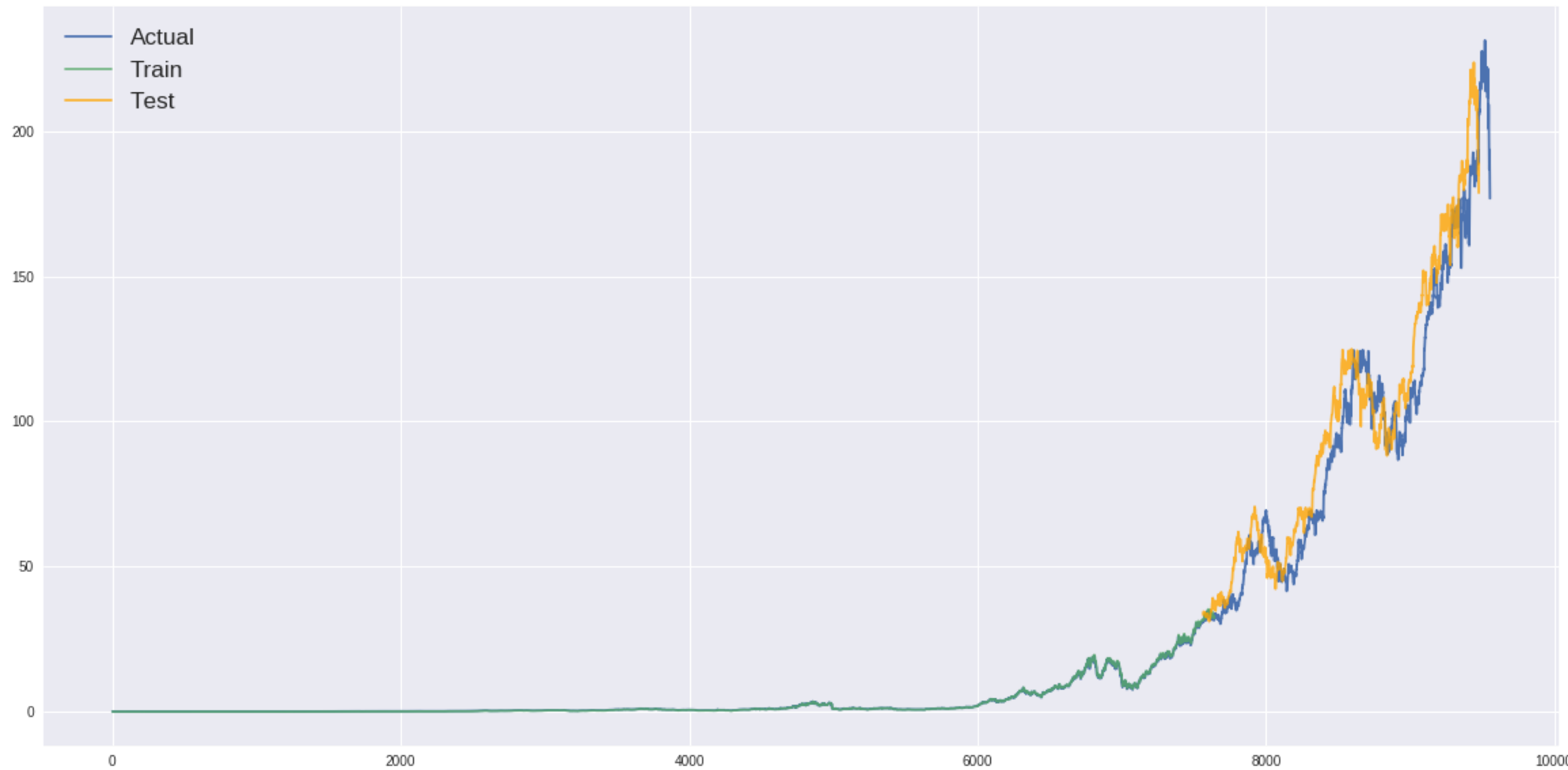
```
Using TensorFlow backend.
[[102.31296]]
yhat [[102.31296]]
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_1 (LSTM)                (None, 50)                10400
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 51
=================================================================
Total params: 10,451
Trainable params: 10,451
Non-trainable params: 0
_________________________________________________________________
None
dict_keys(['loss'])
loss: 0.000000
loss: 1.2578432517784677e-07
```



Source: https://machinelearningmastery.com/how-to-develop-lstm-models-for-time-series-forecasting/

Deep Learning for Financial Time Series Forecasting

https://colab.research.google.com/drive/1aEK0eSev8Q-Y0nNY32geFk7CB8pVgSQM



Source: https://github.com/yash-1337/AAPL_LSTM_Stock_Predictor/blob/master/AAPL_daily_LSTM_stock_predictor.ipynb

Summary

- Recurrent Neural Networks (RNN)
- Long Short Term Memory (LSTM)
- Gated Recurrent Unit (GRU)
- Deep Learning (RNN) for Text Analytics (NLP)
- Deep Learning (RNN) for Time Series Prediction

References

- Christopher Olah, (2015) Understanding LSTM Networks, http://colah.github.io/posts/2015-08-Understanding-LSTMs/
- Brandon Rohrer (2017), Recurrent Neural Networks (RNN) and Long Short-Term Memory (LSTM), https://www.youtube.com/watch?v=WCUNPb-5EYI
- Chris Manning and Richard Socher (2017), Lecture 10: Neural Machine Translation and Models with Attention, https://www.youtube.com/watch?v=IxQtK2SjWWM
- Martin Gorner (2017), Tensorflow, deep learning and modern RNN architectures, without a PhD, https://www.youtube.com/watch?v=pzOzmxCR37I
- Martin Gorner (2017), TensorFlow and Deep Learning without a PhD, Part 1 (Google Cloud Next '17), https://www.youtube.com/watch?v=u4alGiomYP4
- Martin Gorner (2017), TensorFlow and Deep Learning without a PhD, Part 2 (Google Cloud Next '17), https://www.youtube.com/watch?v=fTUwdXUFfI8
- Martin Gorner (2017), TensorFlow and Deep Learning without a PhD, https://goo.gl/pHeXe7, https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist
- Deep Learning Basics: Neural Networks Demystified, https://www.youtube.com/playlist?list=PLiaHhY2iBX9hdHaRr6b7XevZtgZRa1PoU
- 3Blue1Brown (2017), But what *is* a Neural Network? | Chapter 1, deep learning, https://www.youtube.com/watch?v=aircAruvnKk
- 3Blue1Brown (2017), Gradient descent, how neural networks learn | Chapter 2, deep learning, https://www.youtube.com/watch?v=IHZwWFHWa-w
- 3Blue1Brown (2017), What is backpropagation really doing? | Chapter 3, deep learning, https://www.youtube.com/watch?v=Ilg3gGewQ5U
- TensorFlow: https://www.tensorflow.org/
- Keras: http://keras.io/